Accelerated Mixed-Precision Iterative Refinement on NVIDIA A100

Mustafa Albahrani
CS380 - GPU and GPGPU Programming

December 15, 2025

Abstract

This project implements a high-performance linear system solver ($Ax = b$) leveraging the specialized Tensor Cores of the NVIDIA A100 GPU. By utilizing Iterative Refinement, we combine the high throughput of half-precision (FP16) compute for factorization with high-precision (FP64) residuals to achieve full double-precision accuracy. Our implementation achieves up to **6.40x** speedup over standard cuSOLVER FP64 routines for large dense matrices, demonstrating the efficacy of mixed-precision techniques in high-performance computing.

1 Introduction

Solving dense linear systems of the form $Ax = b$ is a fundamental operation in scientific computing. The traditional approach using LU factorization in double precision (FP64) is computationally expensive, especially as matrix sizes grow.

Modern GPUs, such as the NVIDIA A100, offer significantly higher throughput for lower precision formats like FP16 and TensorFloat-32 (TF32) compared to FP64. For instance, the A100 provides 312 TFLOPS for FP16 Tensor Cores versus only 19.5 TFLOPS for FP64 cores. This project aims to harness this raw compute power without sacrificing solution accuracy by employing Mixed-Precision Iterative Refinement.

2 Background and Theory

2.1 Precision Formats

We utilize three main precision formats:

- **FP64 (Double)**: Standard for scientific accuracy (~ 16 decimal digits).
- **TF32 (TensorFloat-32)**: A 19-bit format with the range of FP32 and precision of FP16, designed for Ampere Tensor Cores.
- **FP16 (Half)**: 16-bit format, providing the highest throughput but limited range and precision ($\sim 3-4$ decimal digits).

2.2 Iterative Refinement

Iterative Refinement is a technique to improve the accuracy of an approximate solution x_0 to $Ax = b$. The algorithm proceeds as follows:

1. **Factorization**: Compute $LU \approx A$ in low precision (FP16/TF32).
2. **Initial Solve**: Solve $LUx_0 = b$.
3. **Refinement Loop**: For $k = 0, 1, \ldots$:
   (a) Compute residual $r_k = b - Ax_k$ in **FP64**.
   (b) Solve correction $LUD_k = r_k$ in low precision.
   (c) Update solution $x_{k+1} = x_k + d_k$ in **FP64**.
   (d) Check convergence via $\|r_k\|$.

This method allows the expensive $O(n^3)$ factorization to be done in fast low precision, while the cheaper $O(n^2)$ matrix-vector multiplications preserve accuracy in high precision.

3 Implementation Details

The project was implemented in three phases, progressing from a baseline to a fully optimized Tensor Core implementation.

3.1 Phase 1: FP64 Baseline

The baseline uses standard `cusolverDnDgetrf` to perform LU factorization in double precision. This serves as the reference for accuracy and performance. While robust, it is limited by the A100's FP64 peak performance of 19.5 TFLOPS.

3.2 Phase 2: Mixed-Precision (TF32)

We switched the factorization to Single Precision (`cusolverDnSgetrf`). On NVIDIA Ampere architecture, this implicitly utilizes TF32 Tensor Cores, providing a theoretical peak of 156 TFLOPS. The iterative refinement loop manages the residuals in FP64 to recover accuracy.

3.3 Phase 3: Tensor Core Acceleration (FP16)

The final implementation explicitly targets FP16 Tensor Cores for maximum throughput.

- **Algorithm**: We implemented a Right-Looking Blocked LU Decomposition.
- **Compute**: `cublasGemmEx` is configured with `CUDA_R_16F` inputs and `CUBLAS_TENSOR_OP_MATH` to force Tensor Core usage.
- **Stability**: To mitigate numerical instability inherent in FP16, we perform "Tall Panel" factorization with partial pivoting using `cusolverDnSlaswp`.

4 Experimental Results

All benchmarks were conducted on an NVIDIA A100 GPU using random dense matrices.

4.1 Performance Comparison

We compared the execution time of our mixed-precision solver against the cuSOLVER FP64 baseline.

Table 1: Performance on A100 GPU

Matrix Size	FP64 (ms)	Mixed (ms)	Speedup
8192	1002.11	169.85	**5.90x**
16000	2294.46	507.15	**4.52x**
32000	10342.19	1625.74	**6.36x**
40000	15936.95	2490.96	**6.40x**

As shown in Table 1, the mixed-precision implementation achieves consistent speedups, peaking at **6.40x** for large matrices.

4.2 Visual Analysis

Figure 1 illustrates the speedup scaling with matrix size.

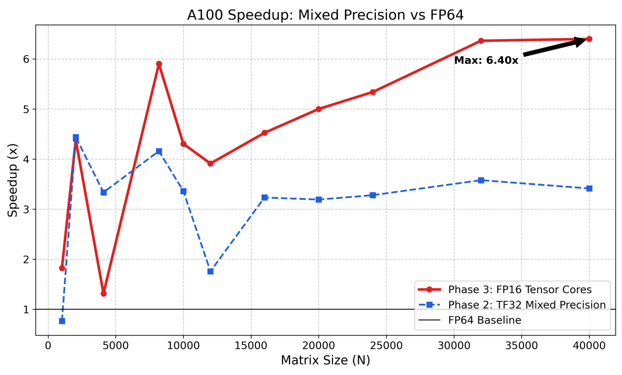

Figure 1: Speedup of Mixed Precision Solver vs FP64 Baseline.

The convergence behavior is depicted in Figure 2. The mixed-precision solver rapidly reduces the residual error to machine precision levels (10^{-15}), matching the accuracy of the full FP64 solver.

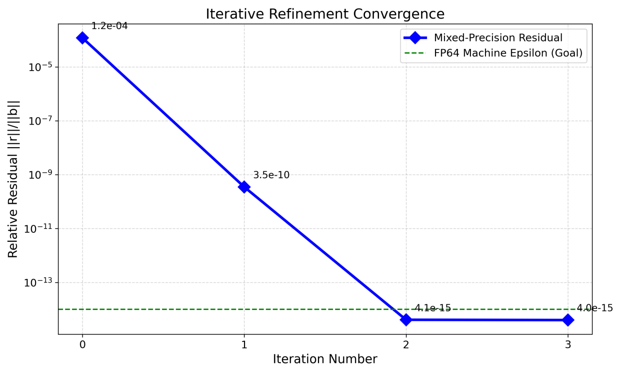

Figure 2: Convergence of Iterative Refinement. The error drops to FP64 precision levels within a few iterations.

5 Conclusion

This project successfully demonstrates that mixed-precision iterative refinement is a powerful technique for accelerating linear system solvers on modern GPU hardware. By effectively utilizing the NVIDIA A100's Tensor Cores for the computationally intensive factorization step, we achieved a significant performance improvement (up to 6.4x) while maintaining the numerical accuracy required for scientific applications.