Problem Statement and Goals
MPIR

Xunzhou (Joe) Ye

Table 1: Revision History

Date	Developer(s)	Change
16 January 2025	Xunzhou	Initial draft

1 Problem Statement

1.1 Problem

In numerical computing, mixed precision *iterative refinement (IR)* is a technique to solve systems of linear equations efficiently while maintaining high accuracy [**lindquist2020improvingperformancegmresmethod**]. It combines computations in lower precision (e.g., single or half precision) for speed and resource efficiency, with higher precision (e.g., double precision) for key operations to ensure numerical stability and accuracy. In sparse matrix solvers, where computational costs and memory access patterns are critical, mixed precision IR balances speed and precision, making it ideal for large-scale simulations, finite element methods, or optimization problems. Lower precision calculations performed in the refinement process require less memory bandwidth and storage, enabling better utilization of memory hierarchies and allowing larger problems to fit into limited memory. These calculations also run faster on modern hardware, particularly GPUs and tensor cores.

Dr. N. Nedialkov and his student Yiqi Huang have done some preliminary works in developing a linear solver for quasi-definite matrices using mixed precision IR, where the internal refinement steps utilizes the *Generalized Minimal Residual (GMRES)* iterative method. The scope of this project is to refactor and improve on the existing implementations.

1.2 Inputs and Outputs

Inputs Like any general purposed numeric linear solver, this solver attempts to solve x satisfying the equation $\mathbf{Ax} = \mathbf{b}$ for some given $n \times n$ real matrix $\mathbf{A}$ and

n real vector $\mathbf{b}$ within certain tolerance ϵ. With the added feature of employing a mixed precision technique, the user can customize on the precision on which they wish to perform matrix factorizations, triangular solves, and computing residuals. The list of inputs is summarized in table 2.

Table 2: List of inputs and parameters

Variable	**Description**
$\mathbf{A}$	$n \times n$ matrix
$\mathbf{b}$	n-vector
ϵ	a solution is found if the norm of the residual is less than ϵ
n_{iter}	the maximum number of iterations to perform
u_f	factorization precision
u_w	working precision
u_r	precision in which the residuals are computed

outputs This solver outputs a numerical approximation of x to the problem $\mathbf{Ax} = \mathbf{b}$ along with some diagnostic information such as the resulting residual of the solve, the number of iterations performed.

table

1.3 Stakeholders

- Supervisor of the project, Dr. N. Nedialkov.
- As the library will be publicly released under an open source license, it is relevant to any individual who is interested in and can make use of high-performance sparse linear solvers for either academic or commercial purposes.

1.4 Environment

Software Any general purpose operating system (OS) with compatible toolchain for building the library, optionally running a few example programs.

Hardware Computer with modern processors.

2 Goals

Basic linear solver function Given some matrix $\mathbf{A}$ and column vector $\mathbf{b}$, the solver should iteratively find $\mathbf{x}$ satisfying the equation $\mathbf{Ax} = \mathbf{b}$ until the residual $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$ is smaller than some tolerance ϵ or the maximum number of iterations n_{iter} is exhausted, whichever comes first.

Mixed precision Given some combinations of floating point precision configuration, the solver should perform internal steps such as matrix factorizations, triangular solves, and residual computation in these configured precisions.

High performance and high efficiency The solver should offer a quantifiable performance or resource utilization advantage over other competing sparse linear solvers.

Ease of use The library should offer a set of streamlined public application programming interfaces (APIs), such that when integrated into other software as a dependent library, the interfaces are self-contained, readable and easy to consume.

Implementation improvement, educational Based on existing implementations, the project should result in an out-of-box, production grade codebase with cross-platform support. As the codebase is developed in an academic setting, the codebase should capture domain knowledge in various forms including documentations, code comments, and academic reports. It should also demonstrate best practices to develop high quality software, including code modularity, traceability, maintainability, and so on.

3 Stretch Goals

Accuracy improvement With the existing solver implementation as being the baseline, the refactored solver should produce more accurate results (residuals lower by at least 1 order of magnitude).

Algorithm optimization The solver should optimize existing algorithms such that given the same set of inputs, it produces results with the same accuracy using notably less time.

4 Challenge Level and Extras

The challenge level of this project is expected to be general, leaning toward the advanced level for the following reasons:

1. This is a continuation of a Master's level research project. The associated thesis is still working in progress.
2. The choice of algorithm and mathematical technique is predefined as part of the project. The mathematical knowledge required for understanding and implementing these algorithms (e.g. GMRES) is graduate level.
3. Working with multiple floating point precisions requires knowledge on computer architectures, low-level floating point representations in computer, as well as advance implementation techniques to support manipulations of precisions.

4. Testing, benchmarking, and evaluating the performance gain on applying these advanced algorithms and techniques are also very involved.