Software Requirements Specification for MPIR: A Sparse Linear Solver

Xunzhou (Joe) Ye

March 26, 2025

Contents

Revision History

Date	Version	Notes
26 January 2025	1.0	Initial draft
26 March 2025	1.1	Updates regarding peer and instructor feedbacks

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Not applicable to this project because it does not interact with any physical system.

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document. The choice of symbols was made to be consistent with common numerical computing literatures.

symbol	**description**
n	The size of a vector/matrix
$\mathbf{A}$	An $n \times n$ matrix to be solved
$\mathbf{b}$	Some 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

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GMRES	General Minimal Residual Method
GS	Goal Statement
IM	Instance Model
IR	Iterative refinement
LC	Likely Change
MP	Mixed-precision
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
TM	Theoretical Model

1.4 Mathematical Notation

This document annotates variables of matrices or vectors in math bold face. For any matrix $\mathbf{A}$ or vector $\mathbf{b}$, one with subscript $\mathbf{A}_i$ or $\mathbf{b}_i$ always means "the ith matrix/vector". $a_{i,j}$ or b_i is used to reference "the element at row i column j in matrix $\mathbf{A}$" or "the ith element in vector $\mathbf{b}$".

2 Introduction

Solving linear systems is a key part of numerical computing and is widely used in many applications, creating a demand for fast and reliable solvers. Modern hardware and software have shown that using lower-precision calculations can be much faster and more memory-efficient than traditional double-precision (64-bit) methods. Mixed-precision (MP) algorithms combine the speed of lower-precision calculations with the accuracy of higher-precision ones to improve performance in various linear solvers. The document describes the solver called MPIR, which implements an iterative method in mixed-precisions and uses General Minimal RESidual (GMRES) in particular for find the error correction vector during the refinement process.

The following section provides an overview of the Software Requirements Specification (SRS) for MPIR. This section explains the purpose of this document, the scope of the requirements, the characteristics of the intended reader, and the organization of the document.

2.1 Purpose of Document

The primary purpose of this document is to record the requirements of MPIR. Goals, assumptions, theoretical models, definitions, and other model derivation information are specified, allowing the reader to fully understand and verify the purpose and scientific basis of MPIR.

2.2 Scope of Requirements

The scope of the software is limited to solving sparse real matrices. For completeness of the software suite, an example program for demonstrating the use of the core solver should be also considered. In terms of writing the requirement specifications itself, operational specifications will be given at a high level without expanding into details on each and every mathematical terms.

2.3 Characteristics of Intended Reader

The intended reader of this document is expected to have a strong foundation in linear algebra at the undergraduate level, with a preference for those who possess graduate-level knowledge in numerical analysis and linear solvers. At a minimum, readers should have completed an introductory undergraduate course in linear algebra and scientific computing, such as MATH 1ZC3 (Linear Algebra) and SFWRENG 4X03 (Scientific Computing) at McMaster University, or equivalent courses from another institution. This background ensures that readers can effectively understand and evaluate the mathematical formulations, algorithmic considerations, and numerical techniques described in this document.

2.4 Organization of Document

The organization of this document follows the template for an SRS for scientific computing software. The presentation follows the standard pattern of presenting goals, theories, definitions, and assumptions. For readers that would like a more bottom up approach, they can start reading the instance models and trace back to find any additional information they require.

The goal statements are refined to the theoretical models and the theoretical models to the instance models.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

3.1 System Context

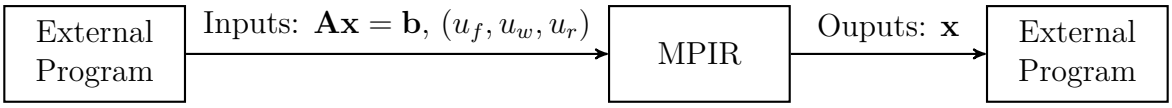

Figure 1: System Context

- User Responsibilities:
  - Provide a sparse matrix $\mathbf{A} : \mathbb{R}^{n \times n}$ in Compressed Sparse Column (CSC) format (*Compressed Sparse Column Format (CSC) — Scipy lecture notes* 2025) stored in memory with appropriate data type.
  - Provide an/multiple vector(s) $\mathbf{b} : \mathbb{R}^n$ to be solved for.
  - Specify 3 floating point arithmetics precisions (u_f, u_w, u_r) in which factorization, numeric solves, and residual computation will be performed respectively.
  - Ensure $\mathbf{A}$ satisfies the system assumption A1.
- MPIR Responsibilities:
  - Numerically solve $\mathbf{x}$ for $\mathbf{Ax} = \mathbf{b}$.

3.2 User Characteristics

The end user of MPIR is typically a technically proficient, domain-expert professional or researcher who is focused on solving large-scale, sparse linear systems efficiently. Users often come from fields such as computational physics, engineering, computer science, applied mathematics, or data science, where large sparse linear systems arise naturally (e.g., from discretized PDEs, network analysis, or optimization problems). They work with problems that are too large or complex for direct solvers, often involving millions or billions of unknowns. The user should also be proficient in programming languages commonly used for numeric analysis and high performance computing such as MATLAB, C/C++, or Fortran. They should have a strong understanding of floating point numbers representations as well as the pitfalls in floating point arithmetics in modern computers. This is especially needed to configure the inputs for the software. At a minimum, the user should have completed an introductory undergraduate course in scientific computing, such as SFWRENG 4X03 (Scientific Computing) at McMaster University, or equivalent courses from another institution.

3.3 System Constraints

This project is based on existing research with the intent to develop on top of the existing implementation of a solver with similar functionalities. This places constraints:

1. Internal matrix solves must use the Generalized Minimal RESidual (GMRES) method (Lindquist, Luszczek, and Dongarra 2020).

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

4.1 Problem Description

MPIR is a solver intended to find a numerical solution $\mathbf{x}$ to a sparse linear system characterized in the equation $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ matrix and $\mathbf{b}$ is an n vector.

4.1.1 Terminology and Definitions

Text descriptions are included along with their mathematical definitions as most of the terms used in this document is about math and linear algebra.

4.1.2 Physical System Description

Not applicable to this project because it does not interact with any physical system.

4.1.3 Goal Statements

Given an $n \times n$ matrix $\mathbf{A}$ and an n vector $\mathbf{b}$, where n is the size of the matrix, the goal statement is:

GS1: find a numerical solution $\mathbf{x}$ to the linear system characterized in the equation $\mathbf{Ax} = \mathbf{b}$.

4.2 Solution Characteristics Specification

This section specifies the information in the solution domain of MPIR. This section is intended to express what is required in such a way that analysts and stakeholders get a clear picture, and the latter will accept it. The purpose of this section is to reduce the problem into one expressed in mathematical terms.

This section presents the solution characteristics by successively refining models. It starts with the abstract/general Theoretical Models (TMs) and refines them to the concrete/specific Instance Models (IMs). The instance models that govern MPIRare presented in Subsection 4.2.9. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Types

Optionally omitted.

4.2.2 Scope Decisions

The scope of this project is inherently limited due to the need to build upon a prior research work (Wong 2024) rather than formulating a completely new approach to the problem described in Section 4.1. A signification portion of this project is dedicated to refactor the prior work, ensuring consistency with its methodology and results before introducing any new contributions. This project must adhere to the same assumptions established in the original study to maintain comparability and validity, as altering these assumptions would fundamentally change the research direction. These assumptions are listed in detail in Section 4.2.4.

4.2.3 Modelling Decisions

Optionally omitted.

4.2.4 Assumptions

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [TM], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: The input matrix $\mathbf{A}$ is non-singular and symmetric quasi-definite [1]. [TM1]

A2: The given precisions follow the order $u_f \leq u_w \leq u_r$, with u_r being the highest precision. [TM4]

4.2.5 Theoretical Models

This section focuses on the general equations and laws that MPIR is based on.

[1] Definitions of symmetric matrices and quasi-definite matrices are given in DD1 and DD3, respectively.

RefName: **TM:LDL**

Label: $\mathbf{LDL}^\intercal$ factorization

Equation: $\mathbf{PKP}^\intercal = \mathbf{LDL}^\intercal$

Description: Symmetric quasi-definite matrices are strongly factorizable. For any permutation $\mathbf{P}$ of some symmetric quasi-definite matrix $\mathbf{K}$, there is a diagonal $\mathbf{D}$ and a unit lower-triangular $\mathbf{L}$ such that the above equation holds.
Under Assumption A1, the input matrix $\mathbf{A}$ can always be decomposed into $\mathbf{LDL}^\intercal$ factors, regardless of whether it is permuted.

Notes: Definitions of symmetric matrices and quasi-definite matrices are given in DD1 and DD3, respectively. In general, if factors are obtained for some matrix $\mathbf{A}$, it is easy to repeatedly solve the problem $\mathbf{Ax} = \mathbf{b}$ for different vector $\mathbf{b}$'s using these factors.

Source: Vanderbei 1995, Gill, Saunders, and Shinnerl 1996

Ref. By: IM1

Preconditions for TM:LDL: Matrix $\mathbf{K}$ is symmetric quasi-definite.

Derivation for TM:LDL: Not Applicable

RefName: **TM:Krylov**

Label: Krylov subspace

Equation: $\mathbf{K}_m(\mathbf{A}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{m-1}\mathbf{r}_0\}$

Description: $\mathbf{A}$ is any given real, $n \times n$ matrix. In the context of solving $\mathbf{Ax} = \mathbf{b}$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$ is the initial residual. $\mathbf{K}_m(\mathbf{A}, \mathbf{r}_0)$ is said to be "the Krylov subspace of $\mathbf{A}$ of order m with respect to $\mathbf{r}_0$"

Notes: In numerical linear algebra, the Krylov subspace is a sequence of subspaces generated by repeatedly multiplying a matrix $\mathbf{A}$ with an initial vector $\mathbf{r}_0$. It is fundamental to many iterative methods for solving large linear systems and eigenvalue problems.

Source: Ascher and Greif 2011, p. 186

Ref. By: GD1

Preconditions for TM:Krylov: None

Derivation for TM:Krylov: Not Applicable

RefName: **TM:Precond**

Label: Matrix Preconditioning

Equation: $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$ or $\mathbf{A}\mathbf{M}^{-1}\mathbf{y} = \mathbf{b}, \mathbf{x} = \mathbf{M}^{-1}\mathbf{y}$

Description: $\mathbf{A}$ is any given real, $n \times n$ matrix. $\mathbf{M}$ is the preconditioner, chosen such that $\mathbf{M}^{-1}\mathbf{A}$ is better conditioned than $\mathbf{A}$. It is often the case that $\mathbf{M}^{-1} \approx \mathbf{A}^{-1}$, and computing $\mathbf{M}^{-1}$ is much easier than computing $\mathbf{A}^{-1}$ directly.

Notes: Intuitively, a matrix $\mathbf{A}$ describes some kind of linear transformation. If $\mathbf{A}$ is poorly conditioned, this transformation can be extreme—small differences between vectors may be greatly amplified, causing closely placed vectors to move far apart. This makes it difficult for iterative solvers to "backtrack" what happened before and after the transformation. By preconditioning $\mathbf{A}$ with an approximate inverse, we are effectively "undoing" some (but not all) of the transformations, taming $\mathbf{A}$ down to some extent. And then we can work with a tamed down, better conditioned version of $\mathbf{A}$ and solve the system from there.

Source: Ascher and Greif 2011, p. 187

Ref. By: GD1

Preconditions for TM:Precond: None

Derivation for TM:Precond: Not Applicable

RefName: **TM:IR**

Label: Iterative refinement in mixed-precision (MP)

Equation:

Algorithm 1 Iterative refinement

1: **for** $m \leftarrow 1, 2, \ldots,$ the mth iteration **do**
2: $\quad \mathbf{r}_m \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_m$ ▷ Compute the residuals
3: $\quad$ Solve $\mathbf{A}\mathbf{d}_m = \mathbf{r}_m$ for $\mathbf{d}_m$ ▷ Compute the correction
4: $\quad \mathbf{x}_{m+1} = \mathbf{x}_m + \mathbf{d}_m$ ▷ Add the correction
5: **end for**

Description: Iterative refinement is a process for reducing the round-off error in the computed solution $\mathbf{x}_0$ to an $n \times n$ system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. Any appropriate method with reasonable accuracy can be used at step 3. Only step 2 requires higher precision. Under the assumption A2 on given arithmetic precision configurations, that the precision used for computing the residual would be the highest, the final result produced by the result is shown to be accurate.

Notes: None

Source: Moler 1967

Ref. By: IM1

Preconditions for TM:IR: None

Derivation for TM:IR: Not Applicable

4.2.6 General Definitions

This section collects the laws and equations that will be used in building the instance models.

Number	GD1
Label	**Restarted GMRES with left preconditioning**
Algorithm	**Algorithm 2** Restarted GMRES with left preconditioning 1: $\mathbf{A} \in \mathbb{R}^{n\times n}, \quad \mathbf{x}_0, \mathbf{b} \in \mathbb{R}^n, \quad \mathbf{M}^{-1} \approx \mathbf{A}^{-1}$ 2: **for** $k \leftarrow 1, 2, \ldots,$ the kth restart **do** 3: $\quad \mathbf{z}_k \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_k$ ▷ Compute residual 4: $\quad \mathbf{r}_k \leftarrow \mathbf{M}^{-1}\mathbf{z}_k$ ▷ Apply preconditioning 5: $\quad \beta \leftarrow \|\mathbf{r}_k\|_2, \quad \mathbf{v}_1 = \mathbf{r}_k/\beta, \quad \mathbf{V}_1 \leftarrow [\mathbf{v}_1]$ ▷ Setup for Arnoldi process 6: $\quad$ Construct an orthogonal basis of preconditioned Krylov subspace $$\text{span}\{\mathbf{r}_k, \mathbf{M}^{-1}\mathbf{A}\mathbf{r}_k, \ldots, (\mathbf{M}^{-1}\mathbf{A})^{m-1}\mathbf{r}_k\}$$ 7: $\quad$ Solve the least square problem and compute the correction 8: $\quad \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$ ▷ Add the correction 9: **end for**
Description	Combination of TM2 and TM3.
Source	Saad 1993, Lindquist, Luszczek, and Dongarra 2020
Ref. By	IM1

4.2.7 Data Definitions

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given.

Number	DD1
Label	**Symmetric matrix**
Equation	$\mathbf{A}^\intercal = \mathbf{A}$
Description	A square matrix is symmetric if it satisfies the equation above.
Sources	Ascher and Greif 2011, p. 78
Ref. By	DD3, IM1

Number	DD2
Label	**Positive definite matrix**
Equation	$\mathbf{x}^\intercal \mathbf{A}\mathbf{x} > 0 \quad \forall \mathbf{x} \neq 0$
Description	A square matrix is positive definite if it satisfies the equation above. For any column vector $\mathbf{x} = (x_1, \ldots, x_n)^\intercal$, we require $\sum_{i,j=1}^{n} a_{i,j} x_i x_j > 0$, provided that at least one component $x_j \neq 0$.
Sources	Ascher and Greif 2011, p. 78
Ref. By	DD3

Number	DD3
Label	**Symmetric quasi-definite matrix**
Equation	$\mathbf{K} = \begin{bmatrix} -\mathbf{E} & \mathbf{A}^\intercal \\ \mathbf{A} & \mathbf{F} \end{bmatrix}$
Description	A symmetric matrix (defined in DD1) $\mathbf{K}$ is quasi-definite if it has the above form. $\mathbf{E}, \mathbf{F}$ are symmetric positive definite matrices (defined in DD2).
Sources	Gill, Saunders, and Shinnerl 1996
Ref. By	TM1, IM1

Number	DD4
Label	**Euclidean norm**
Equation	$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$
Description	The length of the vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ on the n-dimensional Euclidean space $\mathbb{R}^n$.
Sources	Weisstein 2025
Ref. By	GD1, IM1

4.2.8 Data Types

This section collects and defines all the data types needed to document the models.

<table>
<tr><td>Number</td><td>DT1</td></tr>
<tr><td>Type Name</td><td>u, meta type variable for any step that involves floating point arithmetics</td></tr>
<tr><td>Type Def</td><td>
<table>
<tr><th rowspan="2">Arithmetic</th><th rowspan="2">Symbol</th><th colspan="2">Bits</th><th rowspan="2">Unit Roundoff</th><th rowspan="2">Range</th></tr>
<tr><th>Significand</th><th>Exp.</th></tr>
<tr><td>bfloat16</td><td>B</td><td>8</td><td>8</td><td>3.91×10^{-3}</td><td>$10^{\pm 38}$</td></tr>
<tr><td>fp16</td><td>H</td><td>11</td><td>5</td><td>4.88×10^{-4}</td><td>$10^{\pm 5}$</td></tr>
<tr><td>fp32</td><td>S</td><td>24</td><td>8</td><td>5.96×10^{-8}</td><td>$10^{\pm 38}$</td></tr>
<tr><td>fp64</td><td>D</td><td>53</td><td>11</td><td>1.11×10^{-16}</td><td>$10^{\pm 308}$</td></tr>
<tr><td>fp128</td><td>Q</td><td>113</td><td>15</td><td>9.63×10^{-35}</td><td>$10^{\pm 4932}$</td></tr>
</table>
</td></tr>
<tr><td>Description</td><td>The IEEE 754-2019 standard specifies the formats of floating point numbers, including: half, single, double, and double extended (quadruple). Each format has its own representation for numbers, in the form of $2^{k+1-N}n$, with two integers n (signed significand) and k (unbiased signed exponent). Note that bfloat16 is a shorten IEEE single-precision 32-bit float. The value of meta type variable u could be any of the floating point types listed above.</td></tr>
<tr><td>Sources</td><td>"IEEE Standard for Floating-Point Arithmetic" 2019, Wang and Kanwar 2019</td></tr>
</table>

4.2.9 Instance Models

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.7 to replace the abstract symbols in the models identified in Sections 4.2.5 and 4.2.6.

The goal GS1 is solved by IM1 in that under Assumption A1, a) matrix $\mathbf{A}$ is factorizable (symmetric quasi-definite); b) a unique numerical solution exists (non-singular); and under Assumption A2, this solution can be obtained by iterative refinement in mixed-precisions, where the factors are used to speed up the refinement process.

Number	IM1
Label	**GMRES-IR with LDL$^\intercal$ factorization in MP**
Input	$\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{b} \in \mathbb{R}^n, \epsilon, n_{\text{iter}}, u_f, u_w, u_r$ $\mathbf{A}$ is symmetric and quasi-definite under assumption A1. Meta type parameters u_f, u_w, u_r are constrained to values listed in DT1
Output	$\mathbf{x}, \|\mathbf{r}\|_2$, such that, $\mathbf{x}$ is the numerical solution to the problem $\mathbf{Ax} = \mathbf{b}$ up to tolerance ϵ; $\|\mathbf{r}\|_2 = \|\mathbf{b} - \mathbf{Ax}\|_2 \leq \epsilon$.
Description	**Algorithm 3** GMRES-IR with **LDL**$^\intercal$ factorization in MP 1: Perform **LDL**$^\intercal$ factorization of $\mathbf{A}$ ▷ at u_f 2: Solve $\mathbf{LDL}^\intercal\mathbf{x}_0 = \mathbf{b}$ ▷ at u_f 3: **for** $i \leftarrow 0, 1, \ldots, n_{\text{iter}}$ and $\|r_i\|_2 \geq \epsilon$ **do** 4: $r_i \leftarrow \mathbf{b} - \mathbf{Ax}_i$ ▷ at u_r 5: Solve $(\mathbf{LDL}^\intercal)^{-1}\mathbf{Ad}_i = (\mathbf{LDL}^\intercal)^{-1}\mathbf{r_i}$ with GMRES ▷ at u_w 6: $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{d}_i$ ▷ at u_w 7: **end for**
Sources	N/A
Ref. By	GS1

4.2.10 Input Data Constraints

Table 2 shows the data constraints on the input output variables. The column for software constraints restricts the range of inputs to reasonable values. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario.

Table 2: Input Variables

Var	Software Constraints
$\mathbf{A}$	$a_{i,j}$ fits in one of ranges specified in DT1
$\mathbf{b}$	b_i fits in one of ranges specified in DT1
u_f, u_w, u_r	one of the data types in DT1, such that none of the elements in $\mathbf{A}$ or $\mathbf{b}$ overflows the lowest precision u_f
ϵ	fits in the range of u_r
n_{iter}	positive integer value

4.2.11 Example Input Data

Table 3 shows a scenario where all floating point arithmetics are performed at the commonly used double (D) precision in modern 64-bit architectures. ϵ is some multiples of machine epsilon (unit round-off in DT1) in double precision. n_{iter} is an arbitrarily chosen positive integer.

Table 3: Input Example DD

Var	Value
$\mathbf{A}$	$a_{i,j}$: D
$\mathbf{b}$	b_i: D
u_f	D
u_w	D
u_r	D
ϵ	1×10^{-12}
n_{iter}	1000

On the other hand, Table 4 shows a similar scenario with the exception that matrix factorization is performed in single (S) precision while both refinements and residual evaluations are performed in double (D) precision. This combination of precisions is chosen with the intend to save on memory footprint in the matrix factorization step while keeping ϵ (effectively, the accuracy of the solve) the same.

Table 4: Input Example SD

Var	Value
$\mathbf{A}$	$a_{i,j}$: D
$\mathbf{b}$	b_i: D
u_f	S
u_w	D
u_r	D
ϵ	1×10^{-12}
n_{iter}	1000

4.2.12 Properties of a Correct Solution

No addition to the requirements specification.

5 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: 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 norm of 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. This iterative process should strictly follow the operational specifications of IM1.

R2: The solver should allow a configuration of three floating point arithmetic precisions to be used for matrix factorization, iterative refinement, and residual evaluation respectively.

R3: The software package accompanying the solver should provide one example program to demonstrate the process of preparing the inputs for the solver, invoking the solver APIs, and observing the outputs from the solver.

5.2 Nonfunctional Requirements

NFR1: **Accuracy** The solver should converge to a solution within a user-defined tolerance ϵ, with a default threshold of *DEFAULT_TOL* for residual norm reduction. If the solves fails to converge after n_{iter} iterations, the solver should warn about such failure.

NFR2: **Usability** The solver 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 for the intended user characterized in Section 3.2. See *MPIR/docs/VnVPlan/VnVPlan.pdf at main · yex33/MPIR* 2025 for details in verifying the usability of the software through a usability survey.

NFR3: **Maintainability** The effort required to make any of the likely changes listed for MPIR should be less than *FRACTION* of the original development time.

NFR4: **Portability** The library should run on all actively maintained operating systems including Windows 10, Windows 11, Linux, Mac OS.

5.3 Rationale

The scope, modeling choices, and assumptions outlined in this document are primarily informed by Dr. N. Nedialkov's research on mixed-precision techniques in sparse linear solvers. In particular, Kim Ying Wong's recent work establishes the theoretical foundation for combining preconditioned GMRES with iterative refinement in a mixed-precision setting and proposes an implementation based on this approach (Wong 2024). This project builds upon Wong's implementation, inheriting and refining the models and assumptions established in prior research.

The decision to use the GMRES method with iterative refinement is motivated by its relevance to ongoing research in mixed-precision solvers (Bassi, Derakhshan, and Nedialkov 2022). The iterative refinement process specifically requires a solver that: a) Supports factorization at arbitrary floating-point precision, and b) Exposes computed factors for further processing. To meet these requirements, the research team has adopted the QDLDL solver in the mixed-precision (Shahrooz Derakhshan et al. 2023). However, this choice introduces an additional constraint: the input matrix must be symmetric and quasi-definite. Consequently, Assumption A1 ensures compatibility with QDLDL, enabling factorization to be efficiently offloaded. Additionally, Assumption A2 is made to guarantee the progress of iterative refinement, as each iteration depends on obtaining an accurate residual estimation.

6 Likely Changes

This section lists the likely changes to be made to the software.

LC1: The solver accepts generic matrix as the input **A**. The assumption A1 on the inputs being symmetric quasi-definite may be relaxed.

7 Unlikely Changes

This section lists the unlikely changes to be made to the software.

LC2: The goal of the library is to solve sparse matrices.

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 6 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 7 shows the dependencies of instance models, requirements, and data constraints on each other. Table 5 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

	A1	A2
GS1		
TM1	X	
TM2		
TM3		
TM4		X
GD1		
DD1		
DD2		
DD3	X	
DD4		
DT1		
R1	X	
R2		X
R3		
NFR1		
NFR2		
NFR3		
NFR4		
LC1	X	
LC2		

Table 5: Traceability matrix showing the connections between assumptions and other items

	TM1	TM2	TM3	TM4	GD1	DD1	DD2	DD3	DD4	DT1	IM1
TM1								X			
TM2											
TM3											
TM4											
GD1		X	X						X		
DD1											
DD2											
DD3						X	X				
DD4											
DT1											
IM1	X			X	X	X		X	X	X	

Table 6: Traceability matrix showing the connections between models of different sections

	IM1	R1	R2	R3
IM1				
R1	X			
R2	X			
R3	X			

Table 7: Traceability matrix showing the connections between requirements and instance models

9 Development Plan

Optional contents omitted for now.

10 Values of Auxiliary Constants

symbol	**value**
DEFAULT_TOL	1×10^{-6}
FRACTION	30 %

References

Ascher, U. M. and Chen Greif (2011). *A first course in numerical methods*. Computational science & engineering. OCLC: ocn712930845. Philadelphia: Society for Industrial and Applied Mathematics. 552 pp. ISBN: 978-0-89871-997-0.

Bassi, Dylan, Shahrooz Derakhshan, and Ned Nedialkov (Dec. 10, 2022). *Investigating Mixed-Precision in Sparse Linear Solvers and Linear Programming*. 1. Hamilton, Ontario, Canada: McMaster University, p. 18.

Compressed Sparse Column Format (CSC) — Scipy lecture notes (2025). URL: https://scipy-lectures.org/advanced/scipy_sparse/csc_matrix.html (visited on 03/26/2025).

Gill, Philip E., Michael A. Saunders, and Joseph R. Shinnerl (Jan. 1996). "On the Stability of Cholesky Factorization for Symmetric Quasidefinite Systems". In: *SIAM Journal on Matrix Analysis and Applications* 17.1, pp. 35–46. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/S0895479893252623. URL: http://epubs.siam.org/doi/10.1137/S0895479893252623 (visited on 02/05/2025).

"IEEE Standard for Floating-Point Arithmetic" (July 2019). In: *IEEE Std 754-2019 (Revision of IEEE 754-2008)*. Conference Name: IEEE Std 754-2019 (Revision of IEEE 754-2008), pp. 1–84. DOI: 10.1109/IEEESTD.2019.8766229. URL: https://ieeexplore.ieee.org/document/8766229 (visited on 02/05/2025).

Lindquist, Neil, Piotr Luszczek, and Jack Dongarra (Nov. 3, 2020). *Improving the Performance of the GMRES Method using Mixed-Precision Techniques*. DOI: 10.48550/arXiv.2011.01850. arXiv: 2011.01850[math]. URL: http://arxiv.org/abs/2011.01850 (visited on 02/05/2025).

Moler, Cleve B. (Apr. 1967). "Iterative Refinement in Floating Point". In: *Journal of the ACM* 14.2, pp. 316–321. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/321386.321394. URL: https://dl.acm.org/doi/10.1145/321386.321394 (visited on 02/05/2025).

MPIR/docs/VnVPlan/VnVPlan.pdf at main · yex33/MPIR (2025). GitHub. URL: https://github.com/yex33/MPIR/blob/main/docs/VnVPlan/VnVPlan.pdf.

Saad, Youcef (Mar. 1993). "A Flexible Inner-Outer Preconditioned GMRES Algorithm". In: *SIAM Journal on Scientific Computing* 14.2. Publisher: Society for Industrial and Applied Mathematics, pp. 461–469. ISSN: 1064-8275. DOI: 10.1137/0914028. URL: https://epubs.siam.org/doi/10.1137/0914028 (visited on 02/05/2025).

Shahrooz Derakhshan et al. (Dec. 27, 2023). *Using Mixed-Precision in the Linear Solver of Ipopt and in QDLDL*. 2. Hamilton, Ontario, Canada: McMaster University, p. 19.

Vanderbei, Robert J. (Feb. 1995). "Symmetric Quasidefinite Matrices". In: *SIAM Journal on Optimization* 5.1, pp. 100–113. ISSN: 1052-6234, 1095-7189. DOI: 10.1137/0805005. URL: http://epubs.siam.org/doi/10.1137/0805005 (visited on 02/05/2025).

Wang, Shibo and Pankaj Kanwar (Aug. 23, 2019). *BFloat16: The secret to high performance on Cloud TPUs*. Google Cloud Blog. URL: https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus (visited on 02/05/2025).

Weisstein, Eric W. (2025). *Vector Norm*. Publisher: Wolfram Research, Inc. URL: https://mathworld.wolfram.com/VectorNorm.html (visited on 02/06/2025).

Wong, Kim Ying (Dec. 27, 2024). “Exploring a mixed precision GMRES-based iterative refinement sparse linear solver with QDLDL”. MA thesis. Hamilton, Ontario: McMaster University. 41 pp.