Module Interface Specification for MPIR

Xunzhou (Joe) Ye

March 19, 2025

1 Revision History

Date	Version	Notes
19 March 2025	1.0	Initial draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at *MPIR/docs/SRS.pdf at main · yex33/MPIR* 2025

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Introduction **1**

4 Notation **1**

5 Module Decomposition **1**

6 MIS of Matrix Factorization Module **3**
6.1 Module . . . 3
6.2 Uses . . . 3
6.3 Syntax . . . 3
6.3.1 Exported Access Programs . . . 3
6.4 Semantics . . . 3
6.4.1 State Variables . . . 3
6.4.2 Assumptions . . . 3
6.4.3 Access Routine Semantics . . . 3

7 MIS of Solver Output Module **5**
7.1 Module . . . 5
7.2 Uses . . . 5
7.3 Semantics . . . 5
7.3.1 State Variables . . . 5

8 MIS of Iterative Solves Module **6**
8.1 Module . . . 6
8.2 Uses . . . 6
8.3 Syntax . . . 6
8.3.1 Exported Access Programs . . . 6
8.4 Semantics . . . 6
8.4.1 Assumptions . . . 6
8.4.2 Access Routine Semantics . . . 6

9 MIS of Read Matrix Module **7**
9.1 Module . . . 7
9.2 Uses . . . 7
9.3 Syntax . . . 7
9.3.1 Exported Access Programs . . . 7
9.4 Semantics . . . 7
9.4.1 Assumptions . . . 7

9.4.2 Access Routine Semantics 7

10 MIS of Example Program Module **8**
10.1 Module 8
10.2 Uses 8
10.3 Syntax 8
10.3.1 Exported Access Programs 8
10.4 Semantics 8
10.4.1 Access Routine Semantics 8

3 Introduction

The following document details the Module Interface Specifications for MPIR. It is intended to solve a sparse linear system with an iterative method in mixed-precisions.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at *MPIR/docs/SRS.pdf at main · yex33/MPIR* 2025 and *MPIR/docs/Design/SoftArchitecture/MG.pdf at main · yex33/MPIR* 2025.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper 1995, with the addition that template modules have been adapted from Ghezzi, Jazayeri, and Mandrioli 2003. The mathematical notation comes from Chapter 3 of Hoffman and Strooper 1995. For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by MPIR.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of MPIR uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, MPIR uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	–
Behaviour-Hiding Module	Read Matrix Module Solver Output Module Matrix Factorization Module Iterative Solves Module
Software Decision Module	Example Program Module

Table 1: Module Hierarchy

6 MIS of Matrix Factorization Module

6.1 Module

Factorize

6.2 Uses

None

6.3 Syntax

6.3.1 Exported Access Programs

Name	In	Out	Exceptions
QDLDL_etree	$\mathbf{A} : \mathbb{R}^{n \times n}$	$L_{\text{nz}} : \mathbb{N}, \mathbf{E} : \mathbb{R}^n$	NOT_UPPER
QDLDL_factor	$\mathbf{A} : \mathbb{R}^{n \times n}, L_{\text{nz}} : \mathbb{N}, \mathbf{E} : \mathbb{R}^n, u_f$	$\mathbf{L} : \mathbb{R}^{n \times n}, \mathbf{d} : \mathbb{R}^n$	FAC_FAILED
QDLDL_solve	$\mathbf{L} : \mathbb{R}^{n \times n}, \mathbf{d} : \mathbb{R}^n, \mathbf{b} : \mathbb{R}^n, u_w$	$\mathbf{x} : \mathbb{R}^n$	–

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Assumptions

The (sparse) matrix used or returned by this module is stored in Compressed Sparse Column (CSC) format. The input matrix $\mathbf{A}$ only contains data for the upper triangular part.

6.4.3 Access Routine Semantics

QDLDL_etree($\mathbf{A}$):

- output: $\mathbf{E} \coloneqq$ elimination tree for the factorization $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^\intercal$, $L_{\text{nz}} \coloneqq$ the number of non-zeros in the $\mathbf{L}$ factor.
- exception: $err \coloneqq$ (entries found in lower triangle $\implies$ NOT_UPPER)

QDLDL_factor<u_f>($\mathbf{A}, L_{\text{nz}}, \mathbf{E}$):

- output: $\mathbf{L}, \mathbf{D} \coloneqq$ factors of $\mathbf{A}$ in u_f precision, where diagonal matrix $\mathbf{D}$ is simply represented by a vector $\mathbf{d}$ as there's only non-zero elements along the diagonal.
- exception: $err \coloneqq ((\exists\, i : \mathbf{d}_i = 0) \implies$ FAC_FAILED)

QDLDL_solve<u_w>($\mathbf{L}, \mathbf{d}, \mathbf{b}$):

- output: solves $\mathbf{Ax} = \mathbf{LDL}^\intercal\mathbf{x} = \mathbf{b}$ in u_w precision, where $\mathbf{D}$ is reinterpreted from $\mathbf{d}$. Let $\mathbf{y} = \mathbf{DL}^\intercal\mathbf{x}$, we have $\mathbf{LDL}^\intercal\mathbf{x} = \mathbf{L}(\mathbf{DL}^\intercal\mathbf{x}) = \mathbf{Ly}$. Solve $\mathbf{Ly} = \mathbf{b}$ by back substitution (triangular solve). Solve $\mathbf{DL}^\intercal\mathbf{x}$ by multiplying $\mathbf{y}$ with $\frac{1}{d_i}$ followed by another triangular solve.

7 MIS of Solver Output Module

7.1 Module

Stats

7.2 Uses

None

7.3 Semantics

7.3.1 State Variables

Field	Description
$n : \mathbb{N}$	size of the matrix being solved
$n_{\text{nz}} : \mathbb{N}$	number of non-zeros in the matrix
$L_{\text{nz}} : \mathbb{N}$	number of non-zeros in the **L** factor
$t_{\text{fact}} : \mathbb{R}$	time spent in numeric factorization
$t_{\text{solve}} : \mathbb{R}$	time spent in iterative solves
$n_{\text{refine}} : \mathbb{N}$	number of refinement steps

8 MIS of Iterative Solves Module

8.1 Module

Solve

8.2 Uses

Factorize (Section 6), Stats (Section 7)

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
solve	$\mathbf{A} : \mathbb{R}^{n\times n}, \mathbf{b} : \mathbb{R}^n, \epsilon : \mathbb{R}, n_{\text{iter}} : \mathbb{N}, u_f, u_w, u_r$	$\mathbf{x} : \mathbb{R}^n$, s: Stats	–

8.4 Semantics

8.4.1 Assumptions

The input matrix $\mathbf{A}$ is non-singular, quasi-definite, and is stored in CSC format.

8.4.2 Access Routine Semantics

solve$<u_f, u_w, u_r>(\mathbf{A}, \mathbf{b}, \epsilon, n_{\text{iter}})$:

- transition: fill all fields in an instance of Stats record.
- output: solves $\mathbf{Ax} = \mathbf{b}$ in u_w precision following the algorithm specified in IM1 in *MPIR/docs/SRS.pdf at main · yex33/MPIR* 2025.

9 MIS of Read Matrix Module

9.1 Module

ReadMatrix

9.2 Uses

None

9.3 Syntax

9.3.1 Exported Access Programs

Name	In	Out	Exceptions
ReadMMtoCSC	filename: string	$\mathbf{A} : \mathbb{R}^{n \times n}$	–

9.4 Semantics

9.4.1 Assumptions

The file (filename) contains a sparse matrix in Matrix Market Exchange format (*Matrix Market: File Formats* 2013).

9.4.2 Access Routine Semantics

ReadMMtoCSC(filename):

- output: $\mathbf{A} :=$ the sparse matrix parsed and compressed in CSC.

10 MIS of Example Program Module

10.1 Module

Example

10.2 Uses

ReadMatrix (Section 9), Solve (Section 8), Stats (Section 9)

10.3 Syntax

10.3.1 Exported Access Programs

Name	In	Out	Exceptions
demo	filename: string	-	-

10.4 Semantics

10.4.1 Access Routine Semantics

demo():

- transition: read matrix from file (filename), invoke the solver with a predefined set of $\mathbf{b}, \epsilon, n_{\text{iter}}, u_f, u_w, u_r$, and finally print the outputs and stats returned by the solver.

References

Ghezzi, Carlo, Mehdi Jazayeri, and Dino Mandrioli (2003). *Fundamentals of Software Engineering*. 2nd. Upper Saddle River, NJ, USA: Prentice Hall.

Hoffman, Daniel M. and Paul A. Strooper (1995). *Software Design, Automated Testing, and Maintenance: A Practical Approach*. New York, NY, USA: International Thomson Computer Press. URL: http://%20citeseer.ist.psu.edu/428727.html.

Matrix Market: File Formats (Aug. 14, 2013). The Matrix Market. URL: https://math.nist.gov/MatrixMarket/formats.html (visited on 02/14/2025).

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

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