CAS 741 (Development of Scientific Computing Software)

Winter 2025

MPIR Implementations

Xunzhou (Joe) Ye

Faculty of Engineering, McMaster University

April 1, 2025

General Information

- MPIR is a sparse linear solver designed to solve large, sparse real matrices efficiently.
- It uses the General Minimal Residual (GMRES) method for internal matrix solves and iterative refinement techniques to improve both speed and accuracy.
- Intended for use in computational science, engineering, and numerical analysis applications.
- As a complete library suite, the software also includes example programs to demonstrate the solver interfaces and practical use cases of the solver.

Inputs

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

Assumptions and Constraints

A1 Matrix **A** is symmetric quasi-definite.

A2 **A** is stored in *Compressed Sparse Column Format (CSC) — Scipy lecture notes* 2025 format.

A3 Only the upper triangular part of **A** is stored.

A4 The precisions follow the order $u_f \leq u_w \leq u_r$, with u_r being the highest precision.

C1 Use preconditioned GMRES for solving the error correction vector.

Core Algorithms

Algorithm Iterative refinement

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

Core Algorithms

Algorithm GMRES-IR with $\mathbf{LDL}^\intercal$ factorization in MP

1: Perform $\mathbf{LDL}^\intercal$ 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: $\quad r_i \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_i$ ▷ at u_r
5: $\quad$ GMRES Solve $(\mathbf{LDL}^\intercal)^{-1}\mathbf{A}\mathbf{d}_i = (\mathbf{LDL}^\intercal)^{-1}\mathbf{r_i}$ ▷ at u_w
6: $\quad \mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{d}_i$ ▷ at u_r
7: **end for**

Floating Point Precisions

Arithmetic	Sym.	Bits		Type
		Sig.	Exp.	
bfloat16	B	8	8	`std::bfloat16_t`
fp16	H	11	5	`std::float16_t`
fp32	S	24	8	`std::float32_t`, `float`
fp64	D	53	11	`std::float64_t`, `double`
fp128	Q	113	15	`std::float128_t`

Demo

- C++ templates: a bit of metaprogramming to ensure $u_f \leq u_w \leq u_r$.
- CMake/CTest: portability, project dependencies, unit test driver, CI/CD.
- clang-format: also with CI/CD.

Future Work

1. Performance testing
2. Hardly converges in low precisions

References

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).

Questions