POLITECNICO
MILANO 1863

Hybrid Thread-MPI Parallelization for the ADR Equation

Advection-Diffusion-Reaction Equation via FEM

Peng Rao **Jiali Claudio Huang** **Ruiying Jiao**
M.Sc. High Performance Computing — Politecnico di Milano

Outline

Strong Formulation

Find $u : \Omega \to \mathbb{R}$ such that:

ADR Equation with Mixed BCs

$$\begin{cases} -\nabla \cdot (\mu \nabla u) + \nabla \cdot (\boldsymbol{\beta} u) + \gamma u = f & \text{in } \Omega \\ u = g & \text{on } \Gamma_D \\ \nabla u \cdot \boldsymbol{n} = h & \text{on } \Gamma_N \end{cases}$$

- $\Omega \subset \mathbb{R}^d$, $d = 2, 3$
- $\mu > 0$: diffusion coefficient
- $\boldsymbol{\beta} \in [L^\infty(\Omega)]^d$: advection field
- $\gamma \geq 0$: reaction coefficient
- $f \in L^2(\Omega)$: source term
- $\Gamma_D \cup \Gamma_N = \partial\Omega$

Weak Formulation

Define $V_0 = \{v \in H^1(\Omega) : v = 0 \text{ on } \Gamma_D\}$.

Variational Problem

Find $u_0 \in V_0$ such that $a(u_0, v) = F(v)$ for all $v \in V_0$, where:

$$a(u, v) = \int_\Omega \mu \nabla u \cdot \nabla v \, dx + \int_\Omega (\boldsymbol{\beta} \cdot \nabla u) \, v \, dx + \int_\Omega \gamma u v \, dx$$

$$F(v) = \int_\Omega f v \, dx + \int_{\Gamma_N} \mu h v \, ds - a(u_g, v)$$

Manufactured Solution

Domain: $\Omega = [0,1]^d$, $d = 2, 3$.

Exact Solution

$$u_{\mathrm{ex}}(\mathbf{x}) = \prod_{i=1}^{d} \sin(\pi x_i)$$

Physical parameters:

- $\mu = 1.0$ (isotropic diffusion), $\gamma = 0.1$ (reaction)
- Rotational advection: $\boldsymbol{\beta} = (-x_2,\, x_1,\, 0.1)^\top$ — divergence-free
- **Neumann BC** on right face $x_1 = 1$: $h = -\pi \prod_{j\geq 2} \sin(\pi x_j)$
- **Dirichlet BC** $u = 0$ on all other faces

Galerkin Approximation

Triangulation $\mathcal{T}_h$: hexahedral cells, mesh size $h = \max_K \operatorname{diam}(K)$.
Space V_h^k: Q_k Lagrangian elements (tensor-product degree k).

Discrete Problem

Find $u_h \in V_{h,g}$ such that $a(u_h, v_h) = L(v_h)$ for all $v_h \in V_{h,0}$.

Expanding in nodal basis $\{\varphi_j\}$ gives the **linear system**:

$$\mathbf{A}\,\mathbf{U} = \mathbf{F}, \qquad A_{ij} = \int_\Omega \big(\mu\,\nabla\varphi_j \cdot \nabla\varphi_i + (\boldsymbol{\beta} \cdot \nabla\varphi_j)\,\varphi_i + \gamma\,\varphi_j\,\varphi_i\big)\,dx$$

A Priori Error Estimates

For $u \in H^{k+1}(\Omega)$ and Q_k Lagrangian elements:

H^1 Seminorm

$$|u - u_h|_{H^1} \leq C\, h^k\, |u|_{H^{k+1}}$$

Rate: $\mathcal{O}(h^k)$

L^2 Norm

$$\|u - u_h\|_{L^2} \leq C\, h^{k+1}\, |u|_{H^{k+1}}$$

Rate: $\mathcal{O}(h^{k+1})$

- H^1 bound via **Céa's Lemma** + interpolation theory
- L^2 improvement via **Aubin-Nitsche** duality argument

Matrix-Based Solver (AMG)

Assembly — `WorkStream`:

$$A = \sum_{\text{cell}} P_{\text{cell}}^T A_{\text{cell}} P_{\text{cell}}$$

- **Worker:** local cell matrix (embarrassingly parallel)
- **Copier:** scatter to global (sequential)
- Sparse AIJ storage via **PETSc**

Solver:

- GMRES iterative solver
- **AMG** preconditioner (`hypre`)
- Memory: $\mathcal{O}(N_{\text{dof}} \cdot (2k+1)^d)$

Matrix-Free Solver (GMG)

Matrix-vector product computed *on the fly*:

$$\mathbf{v} = \sum_{e=1}^{N_{\text{el}}} P_e^T A_e (P_e\, \mathbf{u}), \quad \text{(no explicit } A \text{ stored)}$$

Optimizations:

- **Sum factorization**: $\mathcal{O}(d(k+1)^{d+1})$ per cell
- **SIMD**: AVX over cell batches
- Memory: $\mathcal{O}(N_{\text{dof}})$

GMG Preconditioner:

- Chebyshev smoothers (deg. 5, range $\times 15$–20)
- Matrix-free transfer operators
- Coarse-grid Chebyshev solve

MPI + Threading Hybrid

MPI — Distributed Memory

- Mesh via `p4est` (Morton SFC)
- Each rank: subset of cells & DoFs
- Ghost cell exchange
- Parallel matrix/vector assembly
- Multigrid transfers across ranks

TBB — Shared Memory

- Within each MPI rank
- MB: `WorkStream` parallel assembly
- MF: `partition_partition` cell loop
- Parallel vector operations

Key Finding

Pure MPI (28M$\times$1T) outperforms pure threading (1M$\times$28T) by **39**$\times$ at 28 cores.

Why MPI Dominates

- **Memory bandwidth**: 28 threads share 1 controller; MPI uses multiple
- **NUMA effects**: remote-NUMA threads suffer 3–$4\times$ latency penalty; MPI ranks can be pinned locally
- **Cache efficiency**: threading shares 16.8M DoFs in one space $\rightarrow$ cache thrash; MPI distributes $\approx 600\mathrm{K}$ DoFs/rank $\rightarrow$ fits in LLC
- **Library optimization**: `deal.II` & PETSc are MPI-first; threading is secondary

Recommendation

Use **pure MPI** (1 process/core). If hybrid is required, maximize MPI ranks and minimize threads per rank.

Convergence Verification ($k = 2,\ d = 2$)

Result

Both implementations confirm the expected L^2-error rate $\mathcal{O}(h^3)$.

Theoretical:

$$\|u - u_h\|_{L^2} \leq C\,h^{k+1} = \mathcal{O}(h^3)$$

- Céa's Lemma + Aubin-Nitsche

Observed:

✓ Matrix-based: $\mathcal{O}(h^3)$

✓ Matrix-free: $\mathcal{O}(h^3)$

- Both solvers are **correct**

Time Complexity — Config: Hybrid 7×4 (28 cores)

	AMG Matrix-Based			**GMG Matrix-Free**		
N_{dof}	Setup	Assem.	Solve	Setup	Assem.	Solve
1K	0.009s	0.001s	0.009s	0.011s	0.0001s	0.013s
17K	0.018s	0.010s	0.025s	0.021s	0.0004s	0.044s
263K	0.159s	0.047s	0.508s	0.189s	0.006s	0.556s
4.2M	2.55s	1.61s	8.03s	2.26s	0.094s	6.98s
67M	40.7s	22.6s	155.5s	37.1s	1.59s	125.3s
268M	167.7s	92.7s	721.8s	147.8s	6.10s	573.3s

- **Crossover** at ≈ 263K DoFs: MF faster beyond this point
- MF assembly: $92.7\text{s} \to 6.1\text{s}$ (on-the-fly vs. explicit storage)
- GMG: **constant 6 iterations** vs. AMG 19–48 (growing with N)

Memory Consumption

Table: Memory (MB) — Hybrid 7×4

N_{dof}	MB	MF
1K	0.37	0.03
17K	2.41	0.28
263K	28.2	4.1
4.2M	416	64
67M	6517	1026
268M	25979	4099

$6.3\times$ reduction at 268M DoFs

- Matrix-based: **26 GB**
- Matrix-free: **4.1 GB**

MB storage: $\mathcal{O}(N \cdot (2k+1)^d)$
MF storage: $\mathcal{O}(N)$ (vectors only)

MF enables solving $6\times$ larger problems on the same hardware.

Strong Scaling & Parallel Efficiency

At 28 cores, 16.8M DoFs:

- MF total: 7.2s MB total: 13.5s
- MF speedup: $\mathbf{1.9}\times$ over MB

Parallel efficiency:

- Matrix-free: $\approx \mathbf{74}\%$ at 28 cores
- Matrix-based: $\approx \mathbf{50}\%$ at 28 cores
- Larger problems $\rightarrow$ better efficiency

MPI vs. Threading (28 cores):

Config	Time	Eff.
28M×1T	7.2s	74%
14M×2T	9.1s	58%
4M×7T	42.3s	10%
1M×28T	281.8s	<2%

Key Findings

1. **Correctness:** Both solvers achieve $\mathcal{O}(h^{k+1})$ L^2 convergence — confirmed against manufactured solution.
2. **Memory ($6.3\times$):** Matrix-free needs 4.1 GB vs. 26 GB at 268M DoFs.
3. **Algorithmic scalability:** GMG-MF maintains **constant 6 iterations** across 10^3–2.68×10^8 DoFs; AMG-MB grows from 19 to 48.
4. **Parallel efficiency:** MF achieves 74% at 28 cores vs. 50% for MB.
5. **MPI $\gg$ Threading:** Pure MPI outperforms pure threading by $\mathbf{39}\times$ at 28 cores due to memory bandwidth, NUMA, and cache effects.

Recommendations & Future Work

Recommendations

- $N_{\text{dof}} > 250\text{K}$: prefer **MF + GMG**
- Small problems / prototyping: MB is competitive
- Production: **pure MPI**, 1 process/core
- Memory-limited: MF solves $6\times$ larger problems

Future Work

- 3D problems & higher *k*
- GPU acceleration for MF operator
- Adaptive mesh refinement
- Time-dependent & nonlinear ADR

Repository

`https://github.com/Peng-Rao/HybridADRSolver`