POLITECNICO
MILANO 1863

23.02.2026

Deal.II Matrix Free Solver for Advection-Diffusion-Reaction

Samuele Allegranza, Vale Turco, Bianca Michielan, Valeriia Potrebiina

Mathematical Background

Advection-Diffusion-Reaction

Problem

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

Weak Formulation

Find $u \in V$ $\quad a(u, v) = f(v) - a(R_g, v) \quad \forall v \in V$

where: $$a(u, v) = \underbrace{\int_\Omega \mu \nabla u \cdot \nabla v}_{\text{diffusion}} + \underbrace{\int_\Omega (\beta \cdot \nabla u) v}_{\text{advection}} + \underbrace{\int_\Omega \gamma u v}_{\text{reaction}}$$

$$f(v) = \int_\Omega f v + \int_{\Gamma_N} h v$$

Mathematical Background

Matrix-Free discretization

$$(A_k)_{ij} = \int_k [\mu \nabla \varphi_j \nabla \varphi_i] + [\beta \cdot \nabla \varphi_j \varphi_i] + [\gamma \varphi_j \varphi_i]$$

$$(A_k)_{ij} \approx \sum_{q=1}^{N_q} [\mu(\nabla \varphi_j(x_q) \cdot \nabla \varphi_i(x_q)) + \beta \cdot \nabla \varphi_j(x_q) \varphi_i(x_q) + \gamma \varphi_j(x_q) \varphi_i(x_q)] \, |J_q| w_q$$

$$(A_k)_{ij} \approx \sum_{q=1}^{N_q} \nabla \varphi_i(x_q) \underbrace{[\mu J_q w_q]}_{D_\mu} \nabla \varphi_j(x_q) + \sum_{q=1}^{N_q} \varphi_i(x_q) \underbrace{[\beta J_q w_q]}_{D_\beta} \nabla \varphi_j(x_q) + \sum_{q=1}^{N_q} \varphi_i(x_q) \underbrace{[\gamma J_q w_q]}_{D_\gamma} \varphi_j(x_q)$$

$$A_k = \underbrace{B^T D_\mu B}_{\text{Diffusion}} + \underbrace{B^T D_\beta K}_{\text{Advection}} + \underbrace{K^T D_\gamma K}_{\text{Reaction}}$$

$$B_{qj} = \nabla \phi_j(x_q) \qquad B, K \in \mathbb{R}^{N_q \times N_{\text{DoF}}}$$

$$K_{qj} = \phi_j(x_q) \qquad D_\mu, D_\beta, D_\gamma \in \mathbb{R}^{N_q \times N_q}$$

Mathematical Background

Non-Homogeneous lifting

Deal.ii Matrix-Free classes **do not** natively support **non-homogeneous problems**...

Our solution:

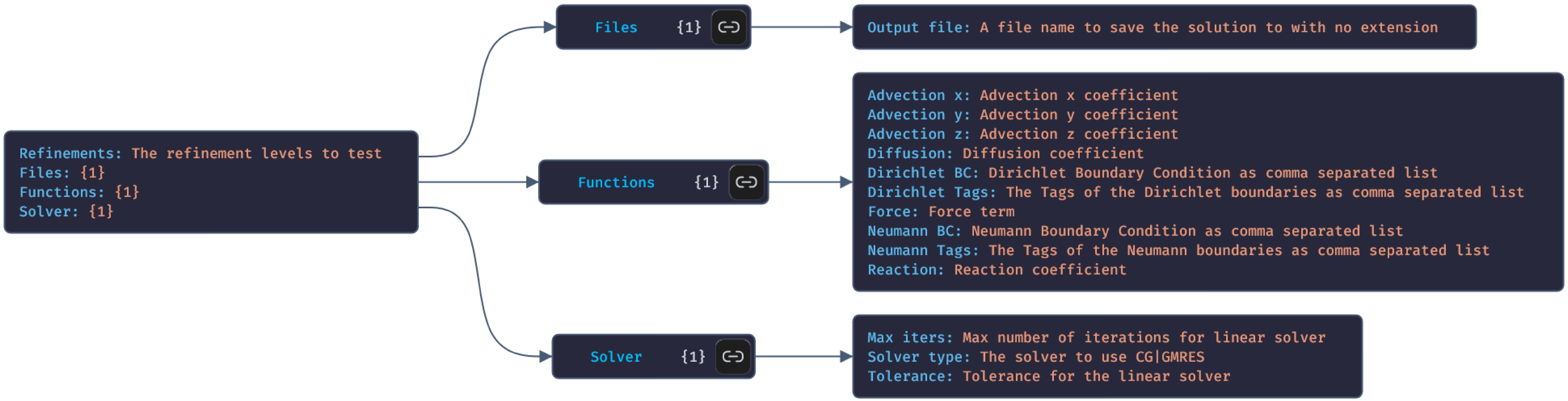
Solve for $Au_0 = \hat{f}$

Where $\hat{f} = f - Au_g$ and $u_g = \begin{cases} g & \text{on the boundary nodes} \\ 0 & \text{elsewhere} \end{cases}$

Project Architecture

Problem definition system

Parameters File (.prm)



Project Architecture

Solver classes

Parameters File (.prm)

Matrix-Based Solver

ADRMBProblem<dim>

→ Multigrid
→ deal.II Jacobi Smoother
→ CG / GMRES

Matrix-Free Solver

ADRProblem<dim>

→ Multigrid
→ Custom Jacobi Smoother
→ CG / GMRES
\+ SIMD Instructions

Usage example

```
Utilities::MPI::MPI_InitFinalize mpi_init(argc, argv,1);
MatrixBasedADR::ADRMBProblem<3> adr_problem;
adr_problem.run("../input/params/pb_3d_mb.prm");
```

Project Architecture

Documentation

Matrix-Free ADR Solver

Implementation of a Matrix-Free solver library for ADR problems

Main Page | Namespaces | Classes | Files

Search

- Matrix-Free ADR Solver
  - Matrix-Free FEM Solver for the Advectio
  - Namespaces
  - Classes
  - Files

Matrix-Free FEM Solver for the Advection-Diffusion-Reaction Equation

A finite element solver for the advection-diffusion-reaction (**ADR**) equation in 2D/3D using deal.II. Compares a **matrix-free** approach (sum factorization + SIMD vectorization + geometric multigrid) against a traditional **matrix-based** approach (sparse matrix assembly) in terms of performance and memory usage.

Table of Contents

Mathematical Formulation

Strong form

$$\begin{cases} -\nabla \cdot (\mu \nabla u) + \beta \cdot \nabla u + \gamma u = f & \text{in } \Omega \subset \mathbb{R}^d, \quad d \in \{1,2,3\} \\ u = g & \text{on } \Gamma_D \subset \partial\Omega \\ \nabla u \cdot \vec{n} = h & \text{on } \Gamma_N = \partial\Omega \setminus \Gamma_D \end{cases}$$

where μ is the diffusion coefficient, β is the advection coefficient, γ is the reaction coefficient, and f is the forcing term.

Weak form

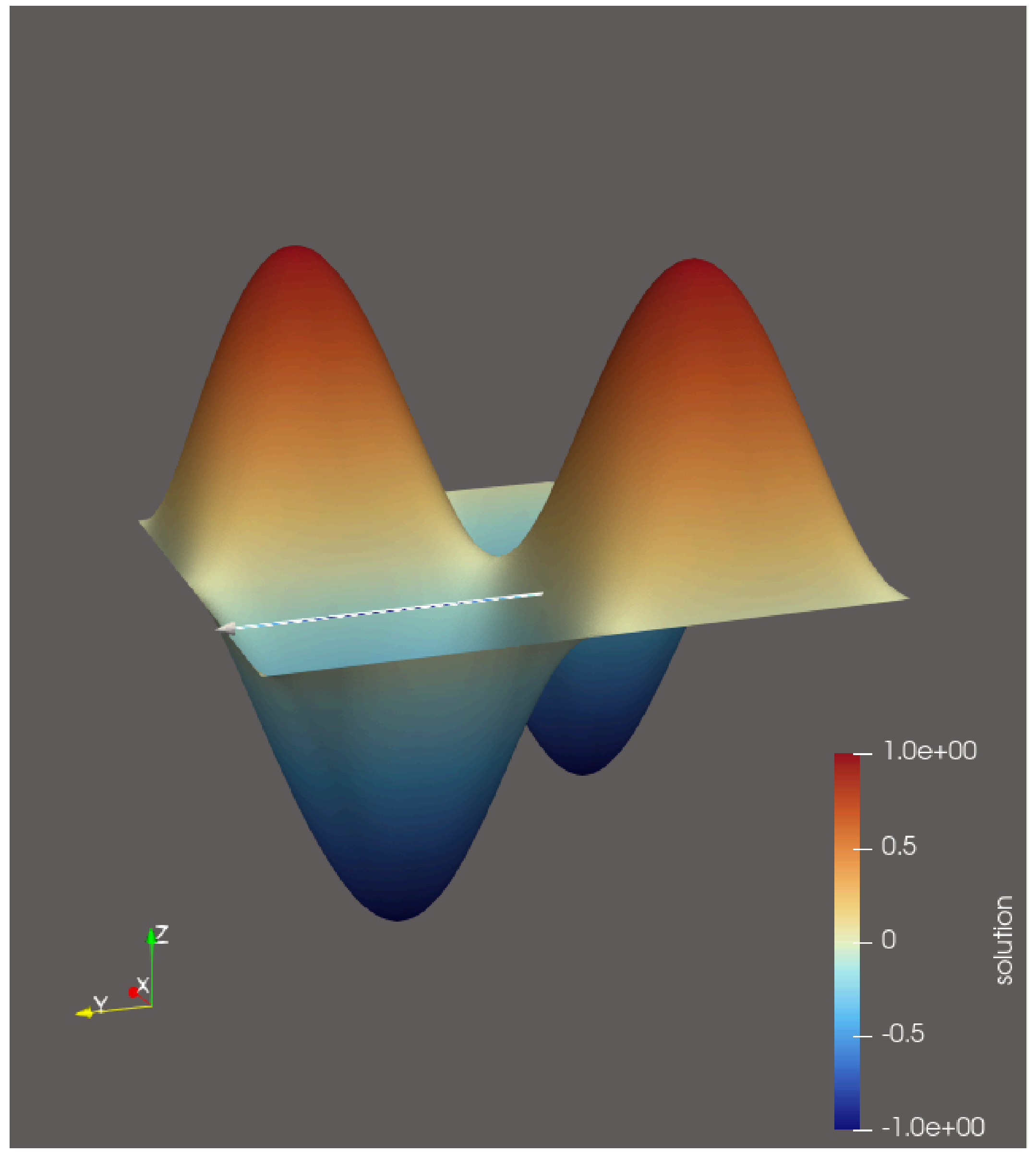
Benchmarks

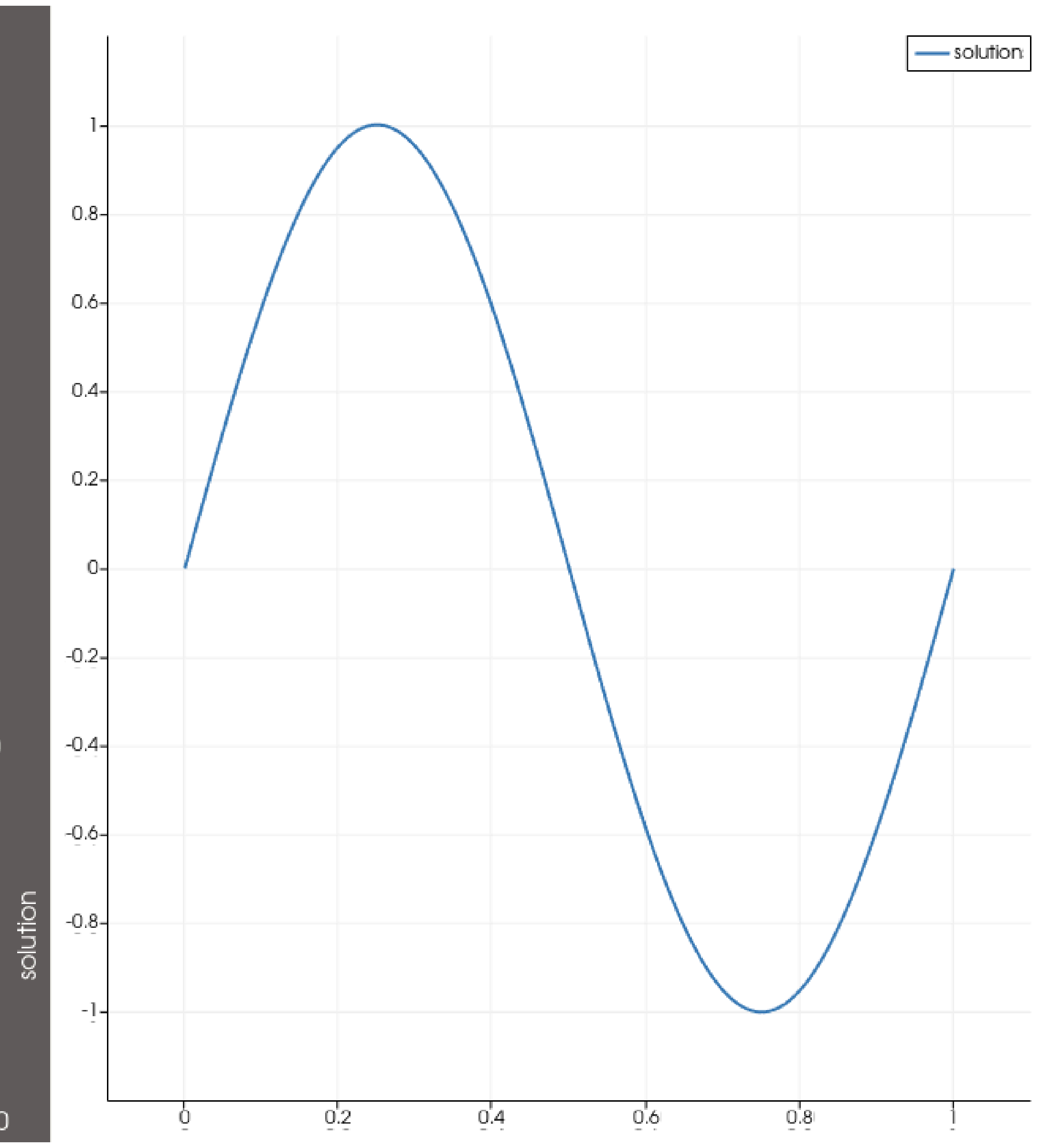
Homogeneous Dirichlet Boundary Conditions

Running on: Intel i7-14700HX @ 5.3GHz
20 Cores | **28 Threads**

2D $u_{ex} = sin(2\pi x) sin(2\pi y) \quad \Omega = [0,1]^2$

$$\beta_{\text{small}} = [0.1 \quad 0.3]^T$$
$$\beta_{\text{medium}} = [10 \quad 30]^T$$
$$\beta_{\text{big}} = [20 \quad 60]^T$$





Matrix free solver, **Single** rank

Ref.	**DoF**	Small **Time (s)**	Medium **Time (s)**	Big **Time (s)**
5	4225	0.54	0.62	0.94
6	16641	2.04	2.43	3.46
7	66049	8.41	9.64	14.40
8	263169	35.74	44.12	57.82

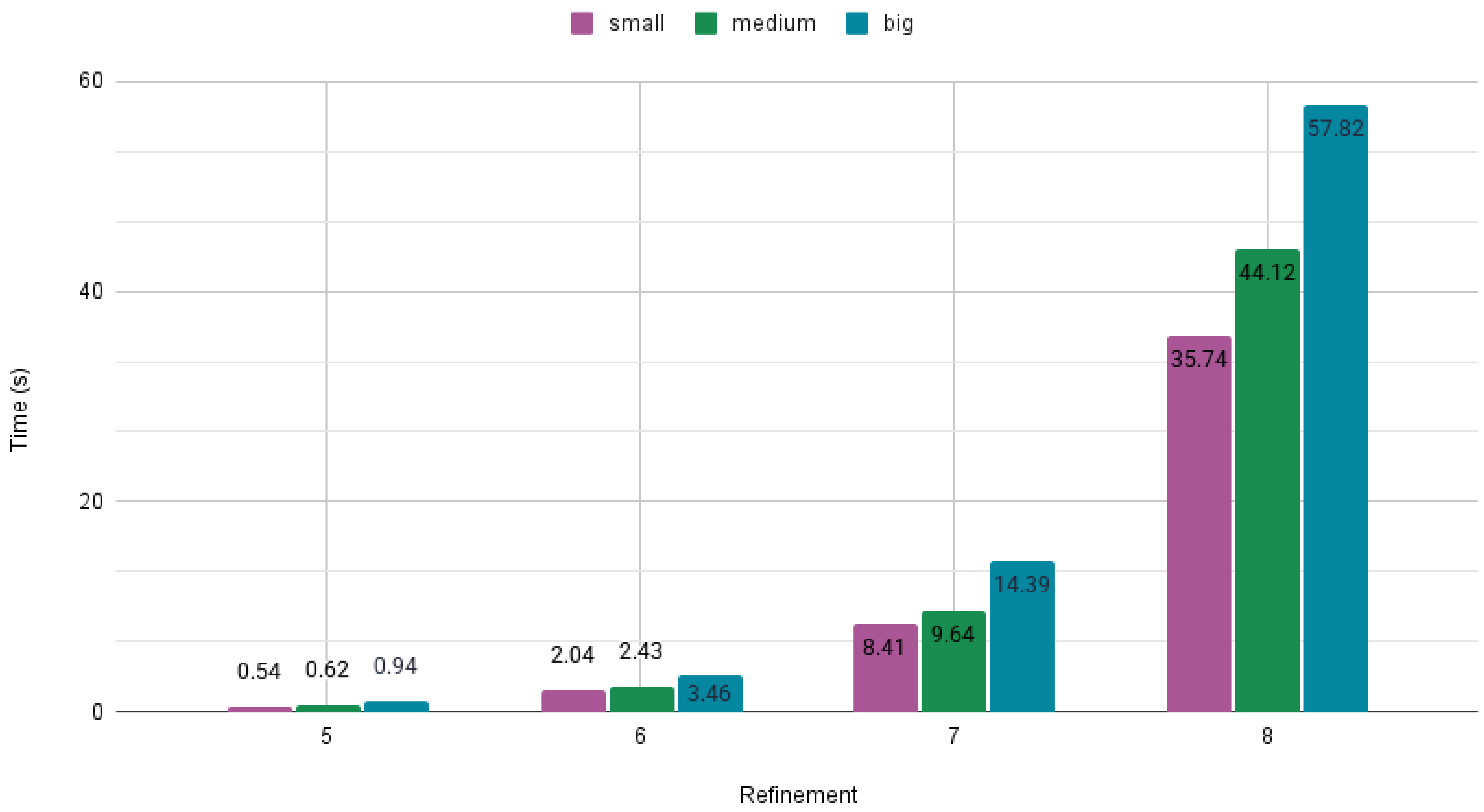
Benchmarks

Homogeneous Dirichlet Boundary Conditions

Running on: Intel i7-14700HX @ 5.3GHz
20 Cores | 28 Threads

2D $u_{ex} = sin(2\pi x) sin(2\pi y) \quad \Omega = [0, 1]^2$

$$\beta_{\text{small}} = [0.1 \quad 0.3]^T$$
$$\beta_{\text{medium}} = [10 \quad 30]^T$$
$$\beta_{\text{big}} = [20 \quad 60]^T$$



$\|\beta\| \ll \mu \implies$ A symmetric Jacobi is a good Smoother

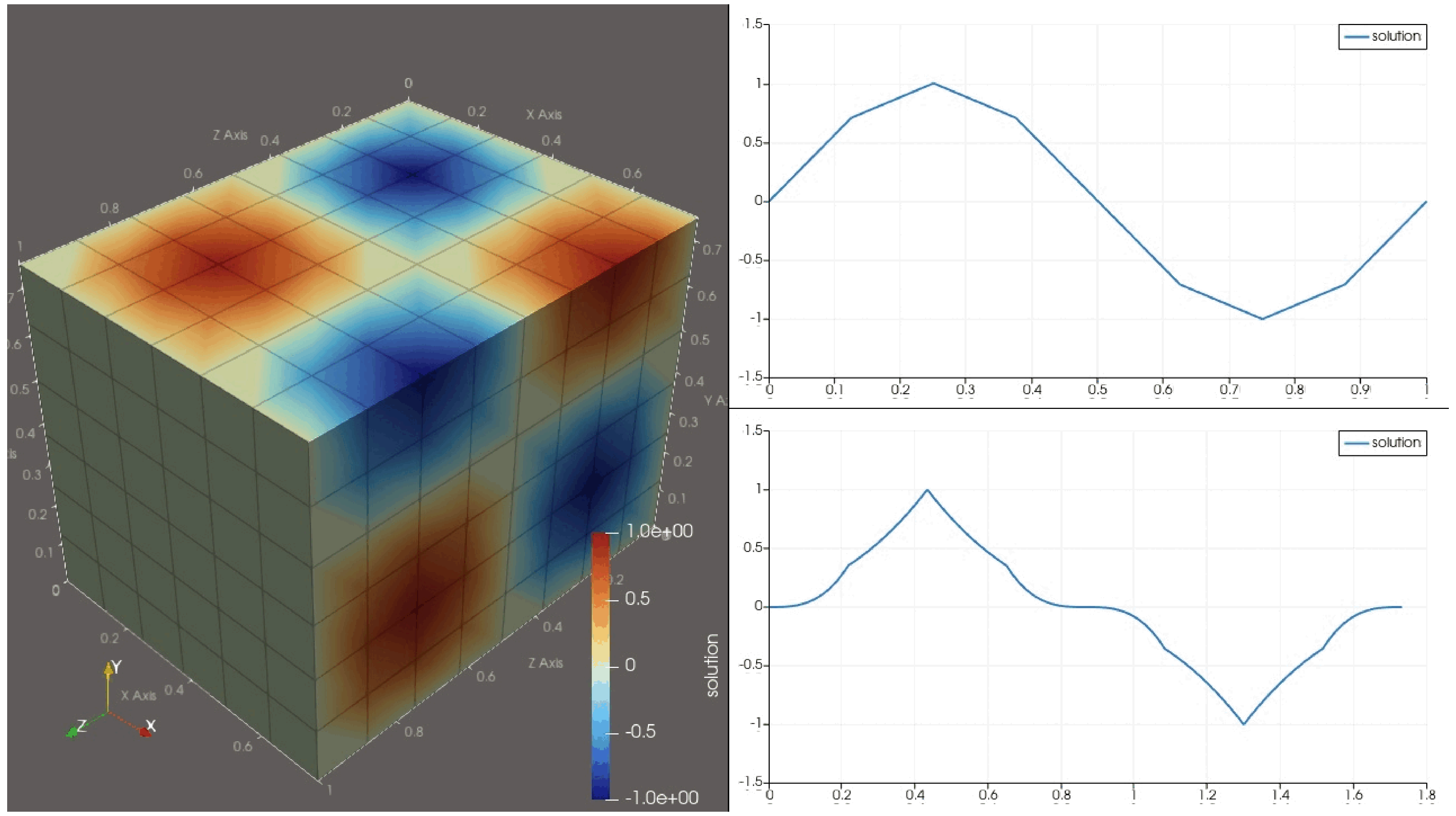
$\|\beta\| \gg \mu \implies$ A not symmetric Jacobi is not a good Smoother

Benchmarks

Homogeneous Dirichlet Boundary Conditions

Running on: Intel i7-14700HX @ 5.3GHz
20 Cores | **28 Threads**

3D $u_{ex} = sin(2\pi x) sin(2\pi y) sin(2\pi z)$ $\Omega = [0,1]^3$ $\beta = [0.1 \quad 0.2 \quad 0.1]^T$



Matrix Free vs **Matrix Based**, **Single** rank

DoF	Matrix Based **Time (s)**	Matrix Free **Time (s)**	Matrix Based **Memory (MB)**	Matrix free **Memory (MB)**
4913	17.14	0.83	4.94	1.28
35937	133.98	6.76	38.29	9.92
274625	1057.77	58.54	301.82	78.26
2146689	8448.95	553.39	2396.80	622.17

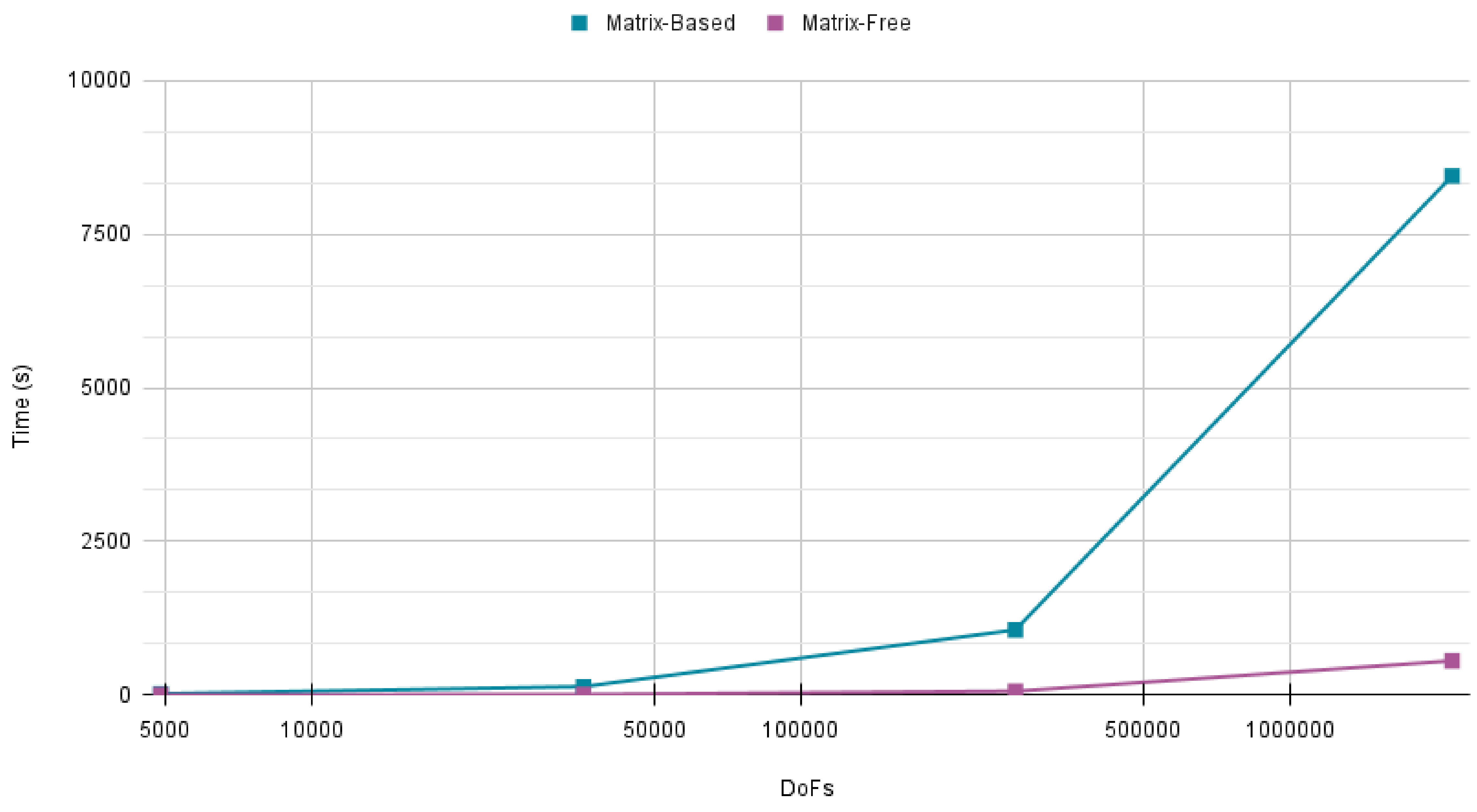
Benchmarks

Homogeneous Dirichlet Boundary Conditions

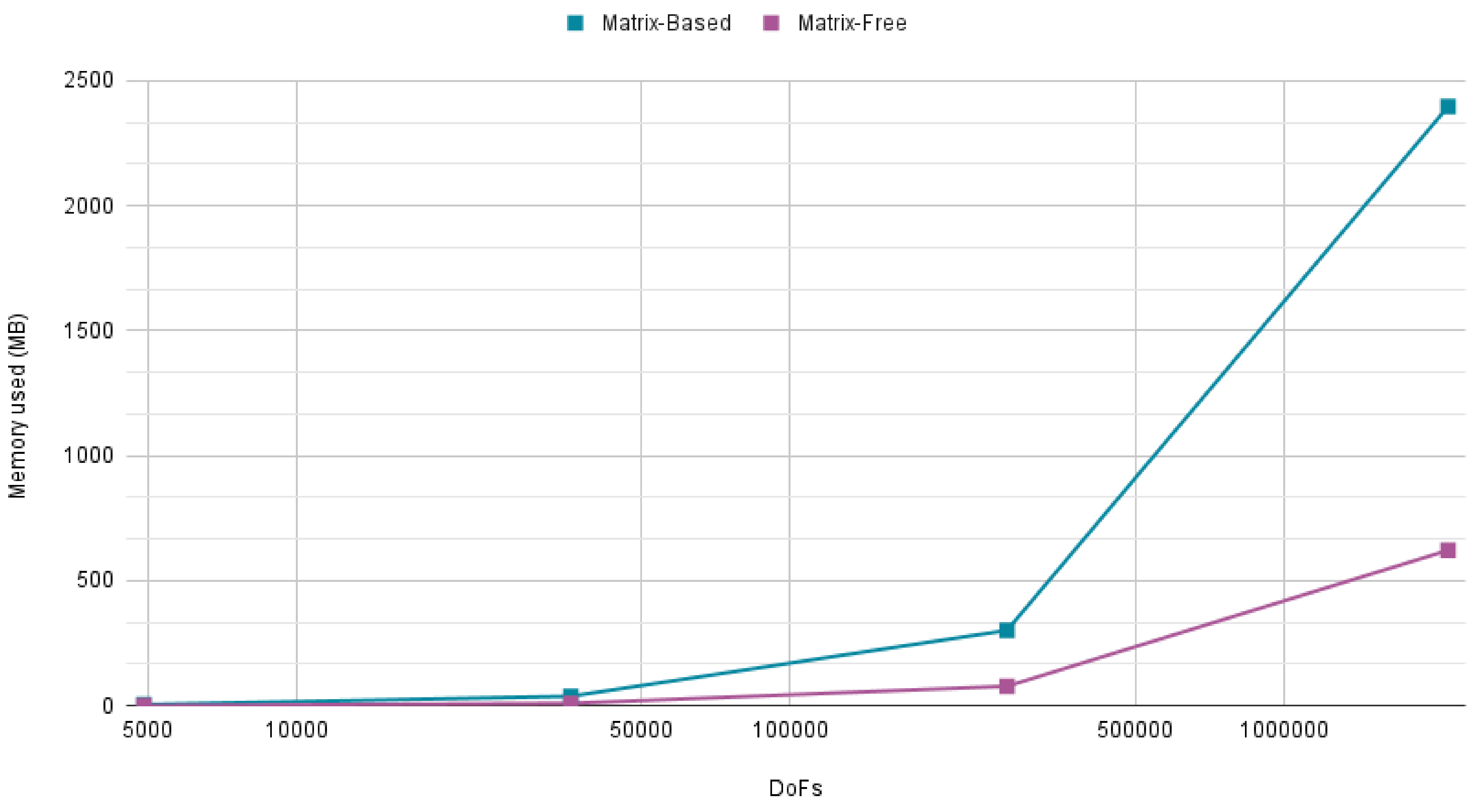
Running on: Intel i7-14700HX @ 5.3GHz
20 Cores | 28 Threads

3D $$u_{ex} = sin(2\pi x)sin(2\pi y)sin(2\pi z) \quad \Omega = [0,1]^3 \quad \beta = \begin{bmatrix}0.1 & 0.2 & 0.1\end{bmatrix}^T$$

Computation time for homogeneous Dirichlet problem



Memory usage for homogeneous Dirichlet problem



Benchmarks

Homogeneous Dirichlet Boundary Conditions

Running on: Intel i7-14700HX @ 5.3GHz
20 Cores | 28 Threads

3D $u_{ex} = sin(2\pi x)sin(2\pi y)sin(2\pi z)$ $\Omega = [0, 1]^3$ $\beta = [0.1 \quad 0.2 \quad 0.1]^T$

Matrix Free solver **Parallel Speedup**

Type	MPI Ranks	Time (s)	Memory (MB)
Matrix based	1	1075.50	8448.95
Matrix free	1	553.39	622.17
Matrix free	4	156.02	161.92
Matrix free	8	109.64	82.80
Matrix free	16	106.66	42.42
Matrix free	20	98.97	34.59

Benchmarks

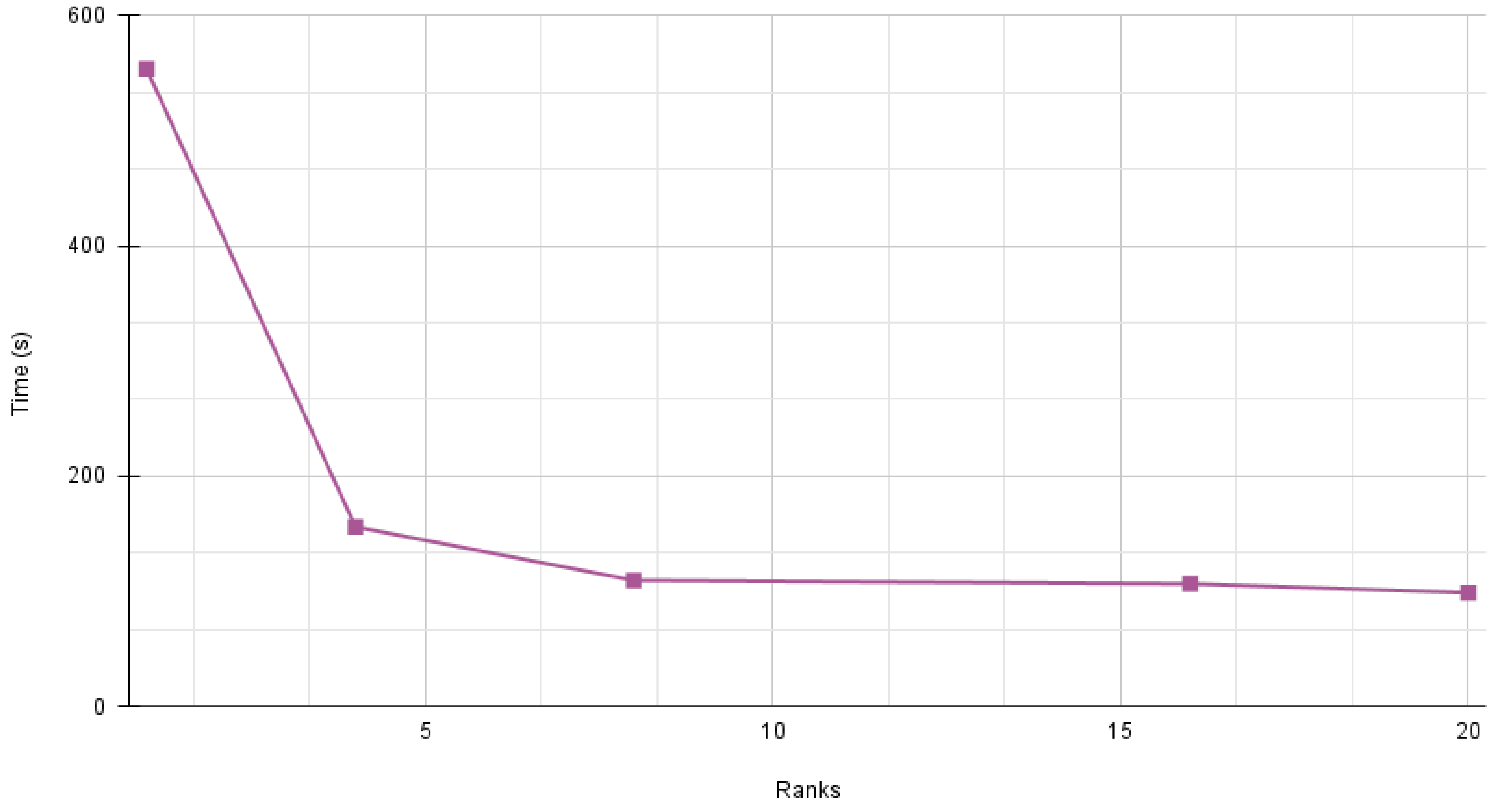
Homogeneous Dirichlet Boundary Conditions

Running on: Intel i7-14700HX @ 5.3GHz
20 Cores | 28 Threads

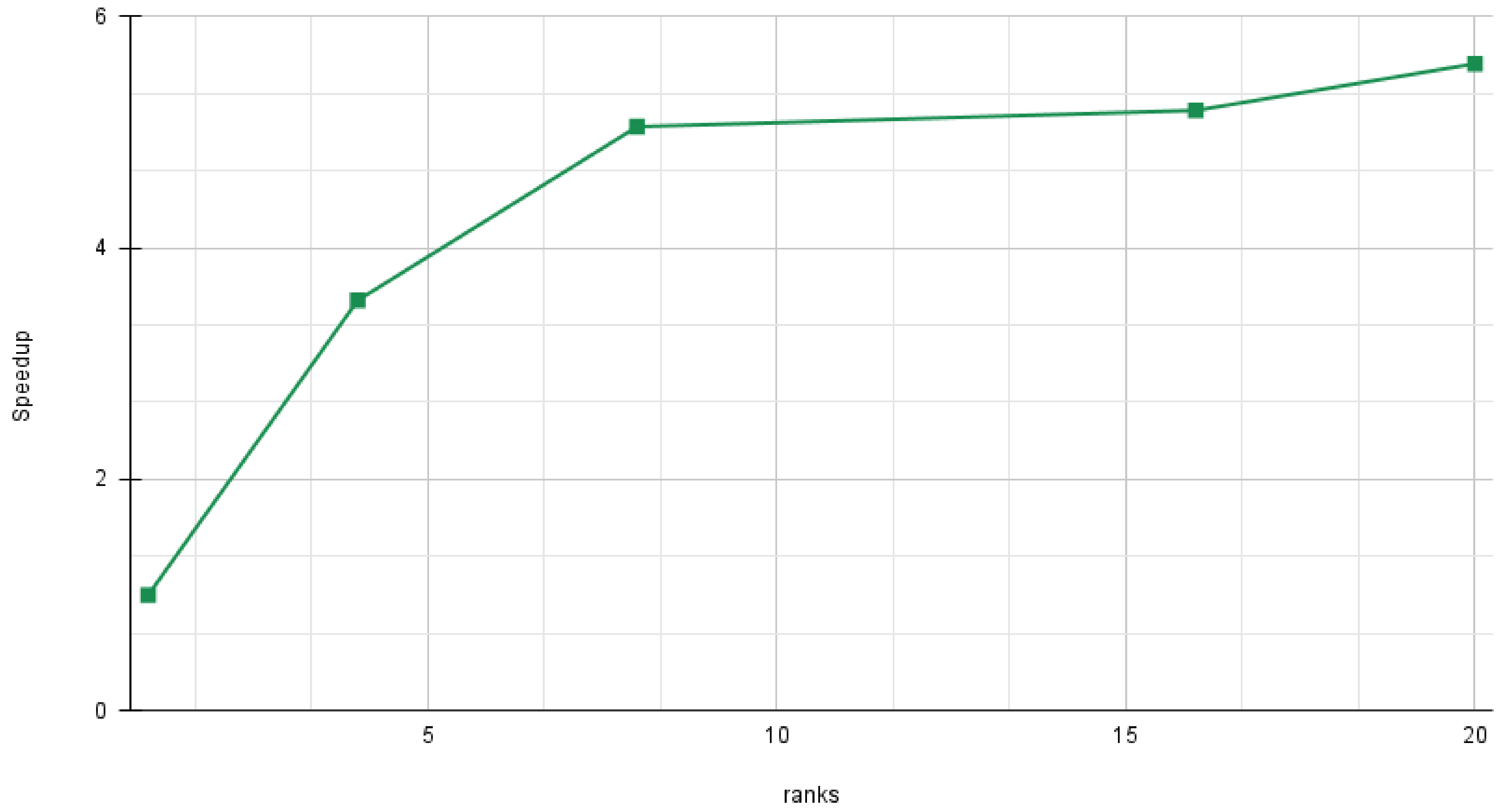
3D $u_{ex} = sin(2\pi x) sin(2\pi y) sin(2\pi z)$ $\quad \Omega = [0,1]^3$ $\quad \beta = [0.1 \quad 0.2 \quad 0.1]^T$

Matrix Free solver **Parallel Speedup**

Computation time vs Ranks



Computation time Speed up

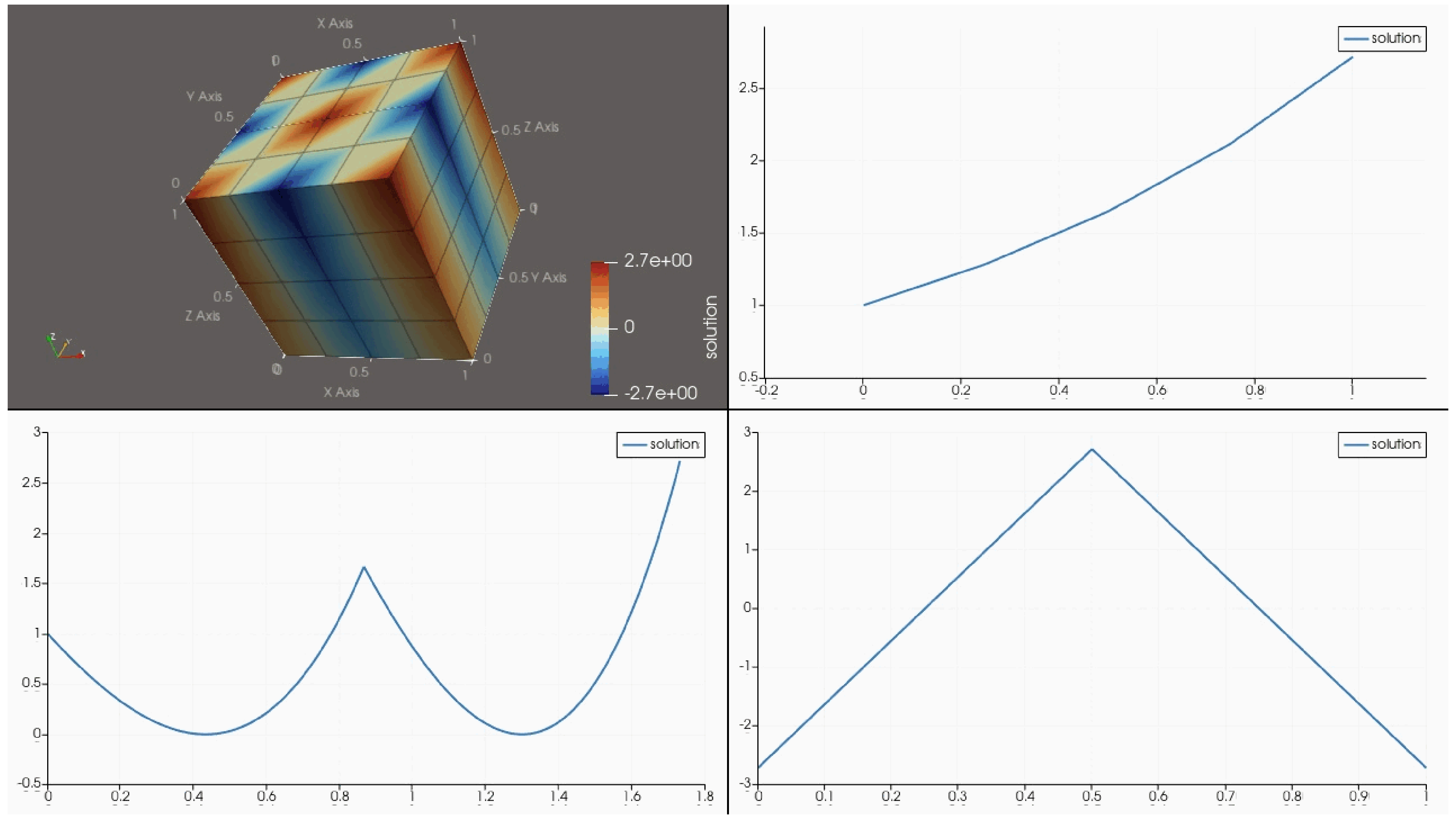


Benchmarks

Non-Homogeneous Dirichlet Boundary Conditions

Running on: Intel i7-14700HX @ 5.3GHz
20 Cores | **28 Threads**

3D $u_{ex} = e^z \cos(2\pi x)\cos(2\pi y) \quad \Omega = [0,1]^3$



Matrix Free vs **Matrix Based**, **Single** rank

DoF	Matrix Based **Time (s)**	Matrix Free **Time (s)**	Matrix Based **Memory (MB)**	Matrix free **Memory (MB)**
729	2.59	0.15	0.66	0.18
4913	20.06	0.97	4.93	1.28
35937	163.84	7.19	38.29	9.91
274625	1075.50	64.74	301.81	78.26

POLITECNICO
MILANO 1863

23.02.2026

Deal.II Matrix Free Solver for ADR Problem

Samuele Allegranza	*samuele.allegranza@polimi.it*
Vale Turco	*vale.turco@polimi.it*
Bianca Michielan	*bianca.michielan@polimi.it*
Valeriia Potrebina	*valeriia.potrebina@polimi.it*