Multigrid Preconditioners for the Cardiac Bidomain Model: a Performance Analysis on HPC Architectures

Edoardo Centofanti
Dept. of Mathematics, Università di Pavia, Italy

Joint work with
Simone Scacchi
Dept. of Mathematics, Università degli studi di Milano, Italy

May 16, 2023

IMPDE2023, Jacques-Louis Lions Laboratory, Sorbonne University, Paris

Introduction

- The bidomain cardiac model[1] is widely used in computational cardiology in order to describe the propagation of the electric potential on the cardiac tissue
- Due to its computational expensiveness, it is important having robust preconditioners in order to solve it numerically
- In this talk we will present the results of tests regarding the application of different Algebraic Multigrid implementations to precondition this problem on multiple CPUs and GPUs

[1] P. Colli Franzone, L.F. Pavarino, and S. Scacchi. *Mathematical Cardiac Electrophysiology*. Reading, Mass.: Springer Cham, 2014.

The Bidomain Model

- We will consider the Parabolic-Elliptic formulation

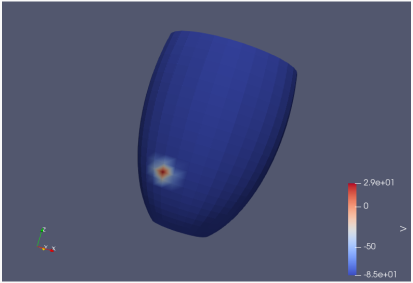
On a space domain Ω (intra and extracellular domains overlap) and a time interval (0,T)

$$
\begin{cases}
\chi C_m \frac{\partial v}{\partial t} - \nabla \cdot (D_i \nabla v) + \nabla \cdot (D_i \nabla u_e) + \chi I_{\text{ion}}(v, \mathbf{w}, \mathbf{c}) = I^i_{\text{app}} & \text{in } \Omega \times (0,T), \\
-\nabla \cdot (D_i \nabla v) - \nabla \cdot ((D_i + D_e) \nabla u_e) = I^i_{\text{app}} + I^e_{\text{app}} & \text{in } \Omega \times (0,T), \\
\frac{\partial \mathbf{w}}{\partial t} - \mathbf{R}(v, \mathbf{w}) = 0 & \text{in } \Omega \times (0,T), \\
\frac{\partial \mathbf{c}}{\partial t} - \mathbf{C}(v, \mathbf{w}, \mathbf{c}) = 0 & \text{in } \Omega \times (0,T), \\
\mathbf{n}^\top D_i \nabla (v + u_e) = 0 & \text{in } \Omega \times (0,T), \\
\mathbf{n}^\top (D_i + D_e) \nabla u_e + \mathbf{n}^\top D_i \nabla v = 0 & \text{in } \Omega \times (0,T).
\end{cases}
$$

Where v is the transmembrane potential, u_e extracellular potential, $\mathbf{w}, \mathbf{c}$ gating and concentration variables related to the ionic model

The Bidomain Model

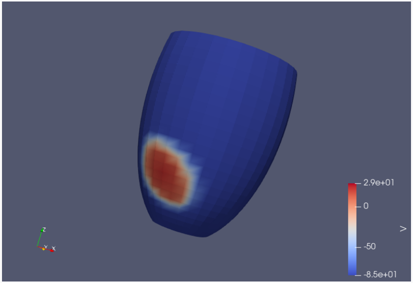
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}^h_{\text{ion}}(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}^{i,h}_{\text{app}} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}^{i,h}_{\text{app}} + \mathbf{I}^{e,h}_{\text{app}}) \end{cases}$$

The Bidomain Model

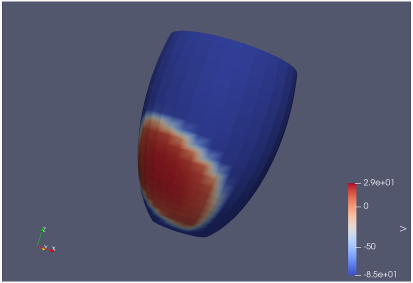
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}_{\text{ion}}^{h}(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}_{\text{app}}^{i,h} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}_{\text{app}}^{i,h} + \mathbf{I}_{\text{app}}^{e,h}) \end{cases}$$

The Bidomain Model

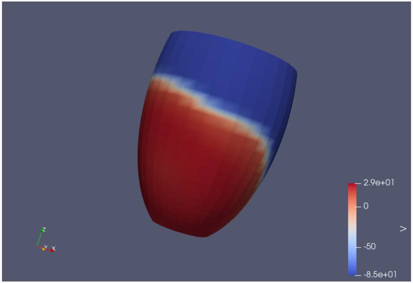
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}_{\text{ion}}^h(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}_{\text{app}}^{i,h} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}_{\text{app}}^{i,h} + \mathbf{I}_{\text{app}}^{e,h}) \end{cases}$$

The Bidomain Model

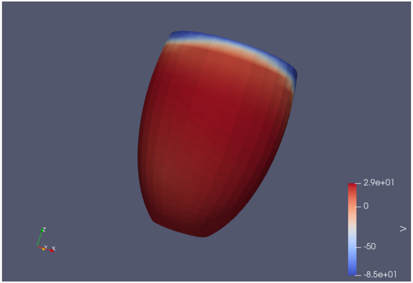
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}_{\text{ion}}^h(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}_{\text{app}}^{i,h} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}_{\text{app}}^{i,h} + \mathbf{I}_{\text{app}}^{e,h}) \end{cases}$$

The Bidomain Model

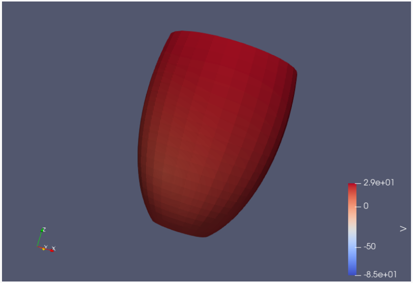
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}_{\text{ion}}^h(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}_{\text{app}}^{i,h} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}_{\text{app}}^{i,h} + \mathbf{I}_{\text{app}}^{e,h}) \end{cases}$$

The Bidomain Model

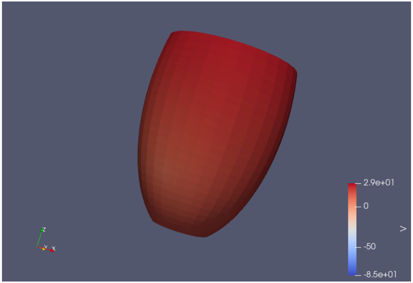
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$
\begin{cases}
\chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}^h_{\mathrm{ion}}(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}^{i,h}_{\mathrm{app}} \\
A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}^{i,h}_{\mathrm{app}} + \mathbf{I}^{e,h}_{\mathrm{app}})
\end{cases}
$$

The Bidomain Model

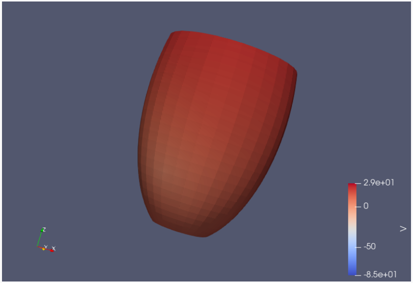
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}_{\text{ion}}^h(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}_{\text{app}}^{i,h} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}_{\text{app}}^{i,h} + \mathbf{I}_{\text{app}}^{e,h}) \end{cases}$$

The Bidomain Model

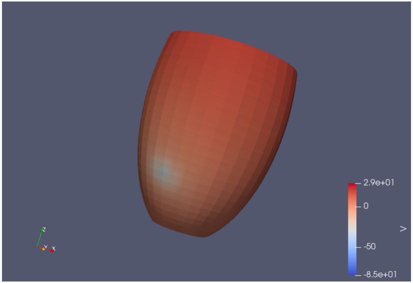
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}^h_{\text{ion}}(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}^{i,h}_{\text{app}} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}^{i,h}_{\text{app}} + \mathbf{I}^{e,h}_{\text{app}}) \end{cases}$$

The Bidomain Model

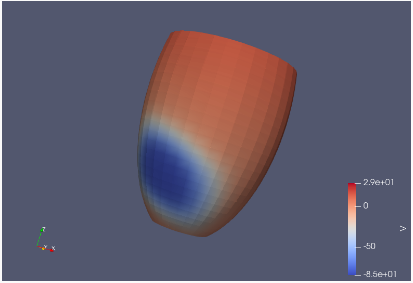
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}^h_{\text{ion}}(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}^{i,h}_{\text{app}} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}^{i,h}_{\text{app}} + \mathbf{I}^{e,h}_{\text{app}}) \end{cases}$$

The Bidomain Model

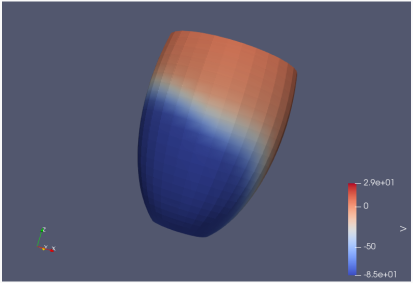
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}_{\text{ion}}^h(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}_{\text{app}}^{i,h} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}_{\text{app}}^{i,h} + \mathbf{I}_{\text{app}}^{e,h}) \end{cases}$$

The Bidomain Model

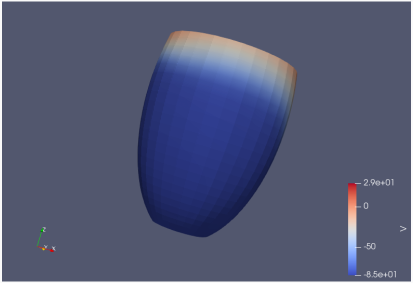
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}_{\text{ion}}^h(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}_{\text{app}}^{i,h} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}_{\text{app}}^{i,h} + \mathbf{I}_{\text{app}}^{e,h}) \end{cases}$$

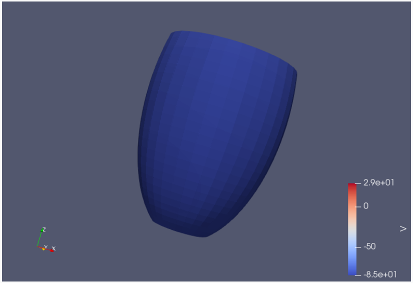
The Bidomain Model

In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}^h_{\text{ion}}(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}^{i,h}_{\text{app}} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}^{i,h}_{\text{app}} + \mathbf{I}^{e,h}_{\text{app}}) \end{cases}$$

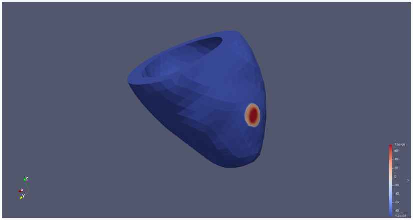
In our setting the problem is spatially discretized through FEM using Q1 elements and ten Tusscher (TT) model is used as ionic model



$$\begin{cases} \chi C_m M \frac{\partial \mathbf{v}_h}{\partial t} + A_i \mathbf{v}_h + A_i \mathbf{u}_{e,h} + \chi M \mathbf{I}_{\text{ion}}^h(\mathbf{v}_h, \mathbf{w}_h, \mathbf{c}_h) = M \mathbf{I}_{\text{app}}^{i,h} \\ A_i \mathbf{v}_h + (A_e + A_i)\mathbf{u}_{e,h} = M(\mathbf{I}_{\text{app}}^{i,h} + \mathbf{I}_{\text{app}}^{e,h}) \end{cases}$$

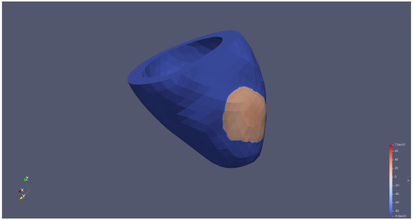
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



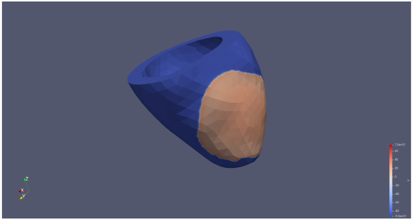
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



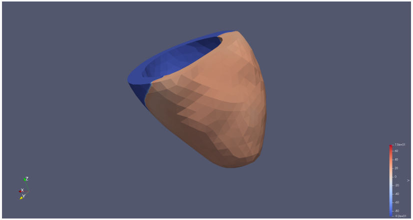
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



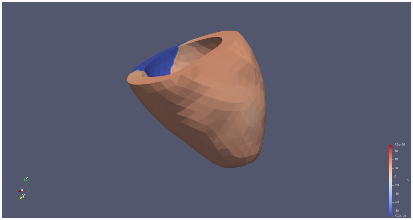
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



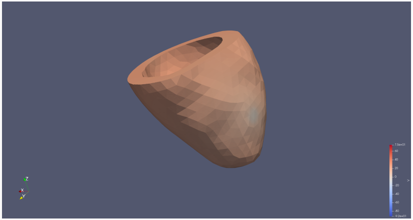
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



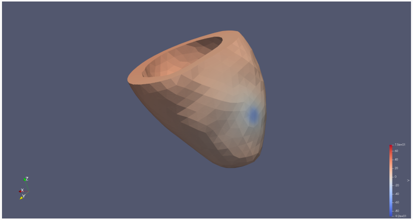
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



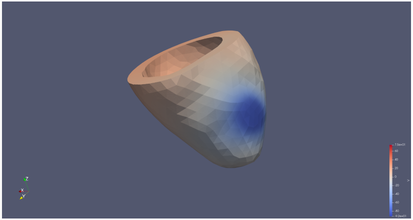
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



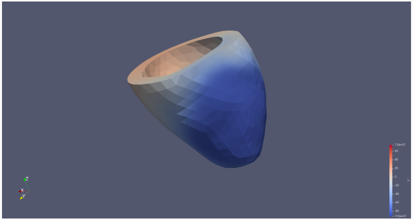
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



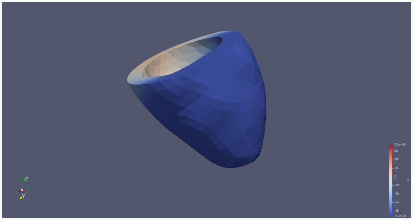
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



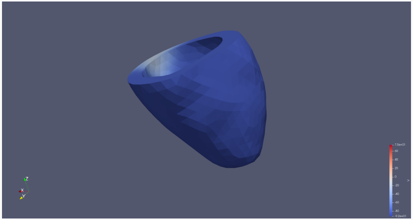
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



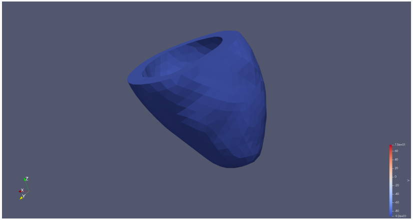
Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



Unstructured mesh

We have solved the same problem presented before on an unstructured mesh representing a ventricle, with 3 different resolutions



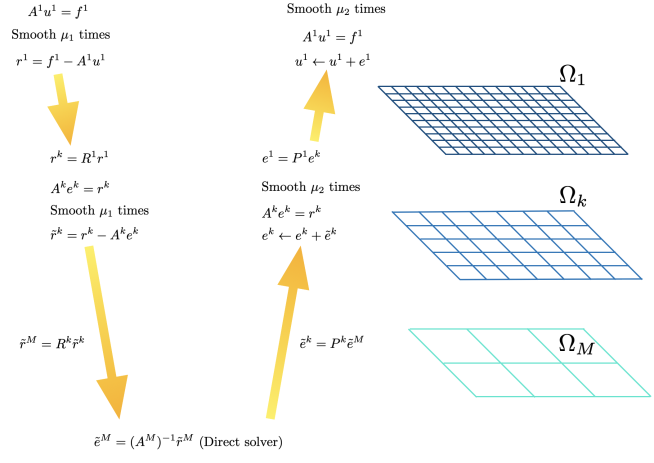
Preconditioned Conjugate Gradient (PCG)

Algorithm 1 PCG

$r_0 = b - A\,x_0$
$z_0 = M^{-1}\,r_0$ ▷ Preconditioning step
$p_0 = z_0$
$k = 0$
for $k = 0;\;\; k <$ maxiter; k++ **do**
  $\alpha_k = \frac{r_k'\,r_k}{p_k'\,A\,p_k}$
  $x_k = x_k + \alpha_k\,p_k$
  $r_k = r_k - \alpha_k\,A\,p_k$
  if r_k is sufficiently small **then**
    exit loop
  end if
  $z_k = M^{-1}\,r_k$ ▷ Preconditioning step
  $\beta_k = \frac{r_k'\,z_k}{r_k'\,z_k}$
  $p_k = z_k + \beta_k\,p_k$
end for
return x_k

Algebraic Multigrid (AMG)

In this work used for preconditioning CG for the elliptic equation



High frequency components of the error are removed by relaxation, smooth components by grid correction

AMG Setup phase

Algorithm 2 AMG setup phase

for $k = 1;\ k < M;\ k++$ **do**

  Partition Ω^k into disjoint sets C^k and F^k.

  Set $\Omega^{k+1} = C^k$.

  Define interpolation P^k.

  Define restriction R^k (often $R^k = (P^k)^\top$).

  $A^{k+1} \leftarrow R^k A^k P^k$

  Set up smoother S^k.

end for

AMG V-cycle algorithm

Algorithm 3 MGV($A^k, R^k, P^k, S^k, u^k, f^k$)

if $k == M$ **then**

  solve $A^M u^M = f^M$ with a direct solver.

else

  apply the smoother S^k μ_1 times to $A^k u^k = f^k$.

  $r^k \leftarrow f^k - A^k u^k$ ▷ Coarse grid correction step

  $r^{k+1} \leftarrow R^k r^k$

  apply MGV($A^{k+1}, R^{k+1}, P^{k+1}, S^{k+1}, e^{k+1}, r^{k+1}$) ▷ Recursion

  $e^k \leftarrow P^k e^{k+1}$ ▷ Interpolation step

  $u^k \leftarrow u^k + e^k$ ▷ Correction

  apply the smoother S^k μ_2 times to $A^k u^k = f^k$.

end if

Algebraic Multigrid (AMG)

Assumption: Smooth error $\longrightarrow$ small residual

$$
\begin{aligned}
&Ae \approx 0 \\
&\sum_{j=1}^{n} a_{ij} e_j \approx 0 \implies a_{ii} e_i \approx -\sum_{j \neq i} a_{ij} e_j
\end{aligned}
$$

Assumption: for any a_{ij} sufficiently small, we can replace e_j with e_i
$\implies$ This motivates the definition of thresholds for coarsening

PETSc (version 3.17) with

- `gamg`, PETSc native implementation
- `BoomerAMG`, high performance parallel implementation provided by the Hypre library[2] (partially wrapped in PETSc)

[2] Robert D. Falgout and Ulrike Meier Yang. “hypre: A Library of High Performance Preconditioners”. In: *Computational Science — ICCS 2002*. Ed. by Peter M. A. Sloot et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 632–641. ISBN: 978-3-540-47789-1.

Threshold (gamg)

Modified Maximal Independent Set (MIS) algorithm[3]

A graph is built from the nodes i, j of the elements $a_{i,j}$ of the matrix A and a weigth $w_{ij} = \frac{|a_{ij}|}{\sqrt{|a_{ii} a_{jj}|}}$ is attributed to each edge (i, j). A threshold on the edge weigth is thus set such that at each coarsening step all the edges with weigth less that the threshold are cut.

[3] Mark F Adams. "Algebraic multigrid methods for constrained linear systems with applications to contact problems in solid mechanics". In: *Numerical linear algebra with applications* 11.2-3 (2004), pp. 141–153.

Strong Threshold (Hypre)

Hybrid-MIS (HMIS) algorithm[4]

A is explored and for each row the coarsening nodes are chosen between the ones satisfying the condition

$$|a_{i,j}| \geq \alpha \max_{k \neq i} |a_{i,k}| \tag{1}$$

with α called *strong threshold* parameter

[4] Hans De Sterck, Ulrike Meier Yang, and Jeffrey J Heys. "Reducing complexity in parallel algebraic multigrid preconditioners". In: *SIAM Journal on Matrix Analysis and Applications* 27.4 (2006), pp. 1019–1039.

PETSc Setup

`gamg`:

- solver: `CG`
- V cycle
- smoother: Chebyshev
- same smoother post and pre coarse grid correction
- Coarsening: Maximal Independent Set (MIS)
  - Threshold: from 0.0 (default) up to 0.07
- Number of levels: 2

`Hypre`:

- solver: `CG`
- V cycle
- smoother: Hybrid Gauss-Seidel
- same smoother post and pre coarse grid correction
- Coarsening: Hybrid Modified Independent set (HMIS)
  - Strong threshold: from 0.25 (default) up to 0.8
- Number of levels: 2

MARCONI100 (CINECA)

- 980 compute nodes with:
  - 2x16 cores IBM POWER9 AC922 at 3.1 GHz
  - 4 x NVIDIA Volta V100 GPUs, Nvlink 2.0, 16GB
  - 256 GB RAM
- Disk Space: 8PB GPFS storage

Tuning Threshold for Structured Mesh

32768 dofs

Results for different threshold parameters for Hypre GPU

Threshold	$\mathrm{It}_{\mathrm{mean}}$ parab	$\mathrm{It}_{\mathrm{mean}}$ ellip	$T_{\text{memb, mean}}$ (s)	$T_{\text{parab, mean}}$ (s)	$T_{\text{ellip, mean}}$ (s)
0.25	25.22	23.30	8.9E-03	1.4E-02	9.6E-02
0.3	25.22	19.29	8.9E-03	1.4E-02	7.4E-02
0.4	25.22	13.92	8.9E-03	1.4E-02	5.6E-02
0.5	25.22	22.81	8.9E-03	1.4E-02	0.10
0.6	25.22	24.57	8.9E-03	1.5E-02	0.11
0.7	25.22	32.11	8.9E-03	1.4E-02	0.15

Tuning Threshold for Structured Mesh

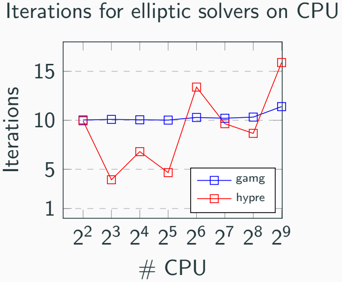
32768 dofs

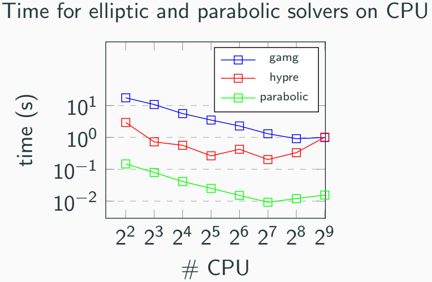
Best results for different implementations

	Threshold	It_{mean} parab	It_{mean} ellip	$T_{\text{memb, mean}}$ (s)	$T_{\text{parab, mean}}$ (s)	$T_{\text{ellip, mean}}$ (s)
Hypre (GPU)	0.4	25.22	13.92	8.9E-03	1.4E-02	5.6E-02
Hypre (CPU)	0.5	3.00	6.27	9.4E-03	2.0E-03	2.4E-02
gamg (CPU)	0.07	3.00	9.64	8.8E-03	2.1E-03	0.07

Results for Structured Mesh

2163330 dofs fixed size problem

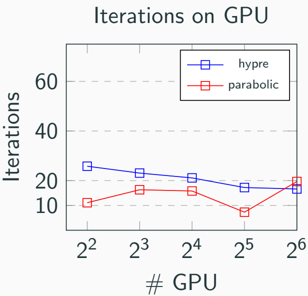


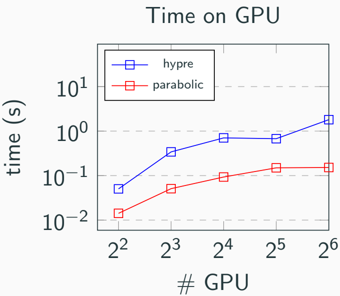


Strong scaling test on CPU. Time and iterations vs number of CPUs.

Results for Structured Mesh

9826 dofs for each GPU

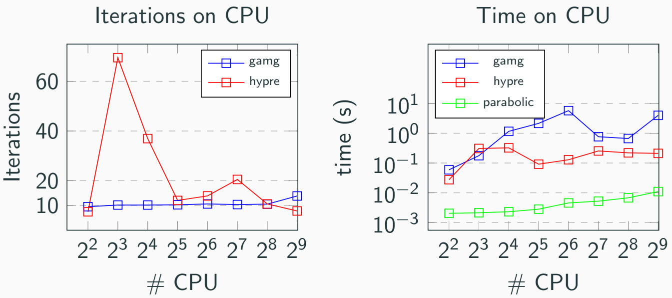




Weak scaling test on GPU (16 nodes) for the structured mesh.

Results for Structured Mesh

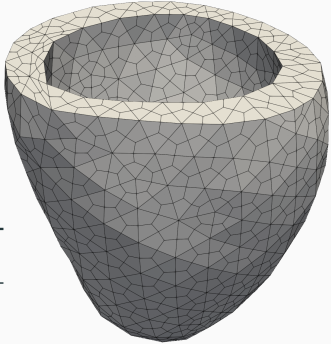
9826 dofs for each CPU



Weak scaling test on CPU (16 nodes) for the structured mesh.

Unstructured mesh

Name	Physical DOFs	Elements
U-mesh 1	35725	30108
U-mesh 2	258415	240864
U-mesh 3	1987285	1926912



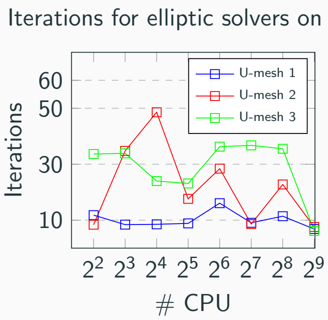
Tuning Threshold for Unstructured Mesh

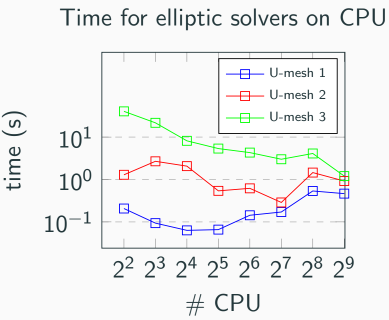
35725 dofs - U-Mesh 1

Best results for different implementations

	Threshold	$\mathrm{It}_{\mathrm{mean}}$ parab	$\mathrm{It}_{\mathrm{mean}}$ ellip	$T_{\text{memb, mean}}$ (s)	$T_{\text{parab, mean}}$ (s)	$T_{\text{ellip, mean}}$ (s)
Hypre (GPU)	0.6	40.28	5.41	1.65e-04	2.51e-02	4.46E-02
Hypre (CPU)	0.5	5.12	3.06	1.09E-02	1.3E-03	3.0E-02
gamg (CPU)	0.07	5.12	11.77	2.8E-03	1.3E-03	4.3E-02

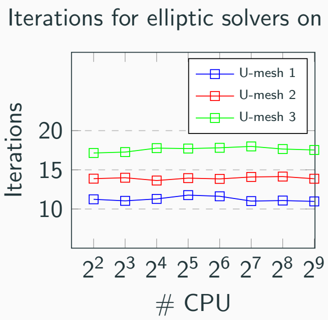
Results for Unstructured Mesh

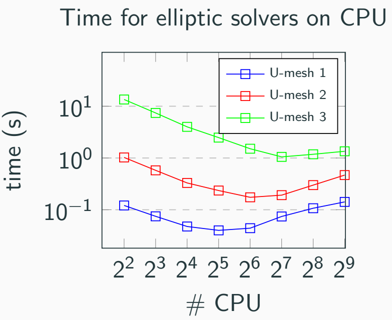




Comparison between times and iterations for solving the elliptic system on different unstructured meshes with Hypre on CPU vs number of GPUs

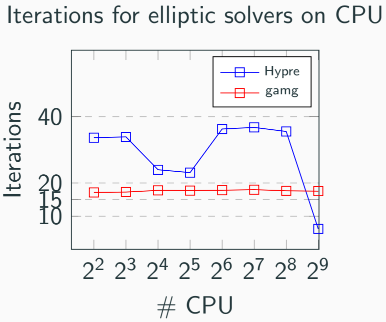
Results for Unstructured Mesh

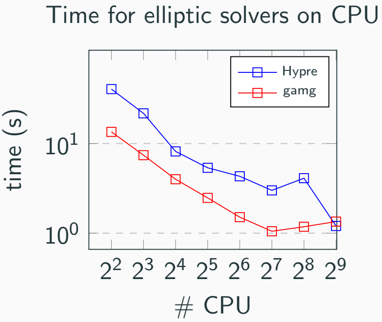




Comparison between times and iterations for solving the elliptic system on different unstructured meshes with gamg on CPU vs number of CPUs

Results for Unstructured Mesh





Comparison between times and iterations for solving the elliptic system on U-mesh 3 with gamg and Hypre on CPU

Results

Best results on CPU and GPU for the structured mesh (2163330 dofs)

Num GPU	It_{mean} parab	It_{mean} ellip	$T_{\text{memb, mean}}$ (s)	$T_{\text{parab, mean}}$ (s)	$T_{\text{ellip, mean}}$ (s)
4 GPU (h)	13.9	5.7	8.4E-04	1.6E-02	5.5E-02
128 CPU (h)	5.0	9.7	1.8E-02	9.2E-03	2.0E-01
256 CPU (g)	5.0	10.3	9.4E-03	5.1E-03	9.1E-01

h stays for Hypre, g for gamg

Results

Best results on CPU and GPU for the unstructured mesh (U-Mesh 3, 1987285 dofs)

Num GPU	$\mathrm{It}_{\text{mean}}$ parab	$\mathrm{It}_{\text{mean}}$ ellip	$T_{\text{memb, mean}}$ (s)	$T_{\text{parab, mean}}$ (s)	$T_{\text{ellip, mean}}$ (s)
8 GPU (h)	57.7	8.4	1.2E-02	1.8E-01	3.3E-01
512 CPU (h)	27.7	6.1	1.2E-02	7.5E-02	1.2
128 CPU (g)	27.7	1.8	9.8E-04	4.9E-02	1.1

h stays for Hypre, g for gamg

Conclusion

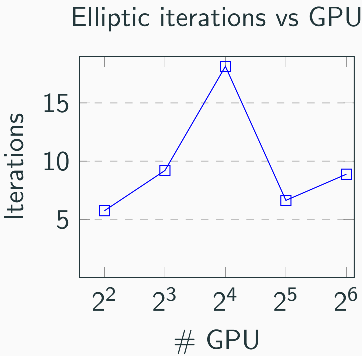
- On CPU we have confirmed the scalability properties of AMG[5], both on structured and unstructured mesh
- On GPU we obtained better performance with respect to CPU
- Benefits and drawbacks of using PETSc with GPUs are well known[6]

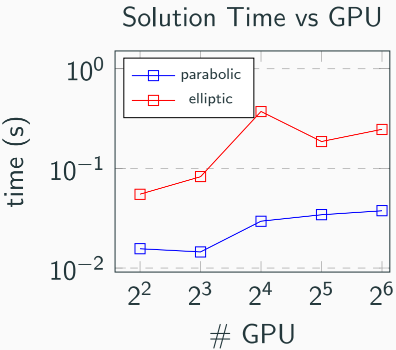
[5] Andrew J Cleary et al. "Robustness and scalability of algebraic multigrid". In: *SIAM Journal on Scientific Computing* 21.5 (2000), pp. 1886–1908.
[6] Richard Tran Mills et al. "Toward performance-portable PETSc for GPU-based exascale systems". In: *Parallel Computing* 108 (2021), p. 102831.

Thank you for your attention

Results for Structured Mesh

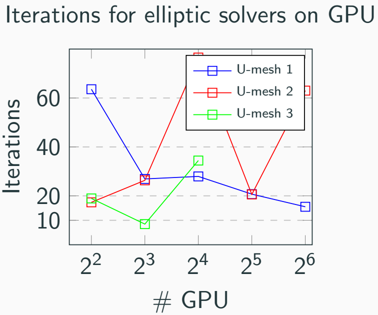
2163330 dofs fixed size problem

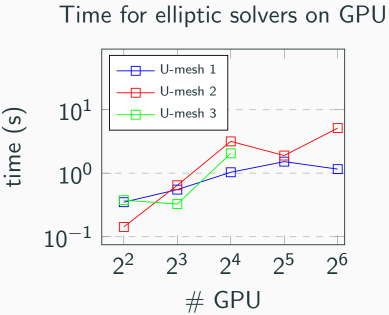




Strong scaling test on GPU. Time and iteration vs number of GPUs.
Notice the effect of synchronization overheads

Results for Unstructured Mesh





Comparison between times and iterations for solving the elliptic system on different unstructured meshes on GPU vs number of GPUs. U-mesh 3 goes out of memory with 32 and 64 GPUs.