An AMG reduction framework for Poisson's equation in CFD simulations

Àdel Alsalti-Baldellou[1,2] Carlo Janna[3] Xavier Álvarez-Farré[1]
F. Xavier Trias[1]

[1]Heat and Mass Transfer Technological Center,
Technical University of Catalonia

[2]Termo Fluids SL,
http://www.termofluids.com/

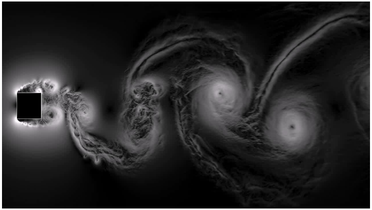
[3]Department of Civil, Environmental and Architectural Engineering,
University of Padova

May 11th 2023

Index

Context of the work

CFD applications – 1



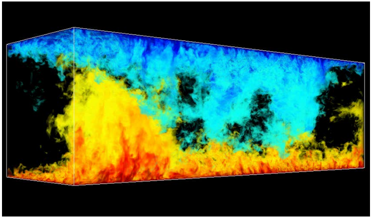


Figure: Simulation of flow around a square cylinder[1] and Rayleigh-Bénard convection[2].

[1] F.X. Trias et al. (2015). "Turbulent flow around a square cylinder at Reynolds number 22000: a DNS study" in *Computers and Fluids*.

[2] F. Dabbagh et al. (2017). "A priori study of subgrid-scale features in turbulent Rayleigh-Bénard convection" in *Physics of Fluids*.

CFD applications – 2

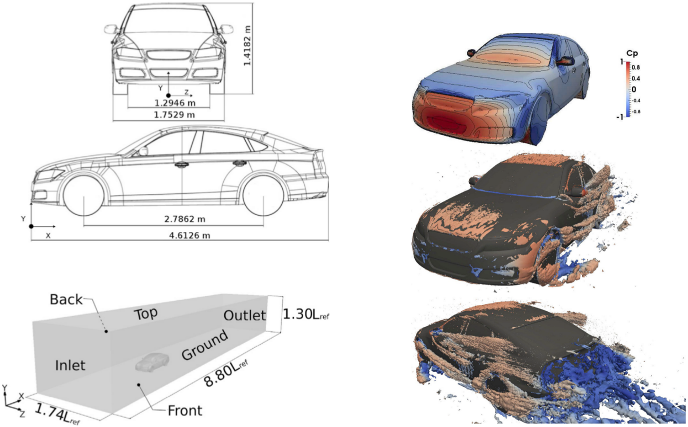


Figure: Simulation of turbulent flow over the DrivAer fastback vehicle model[3].

[3] D. E. Aljure et al. (2018). "Flow over a realistic car model: Wall modeled large eddy simulations assessment and unsteady effects" in *Journal of Wind Engineering and Industrial Aerodynamics*.

CFD applications – 3

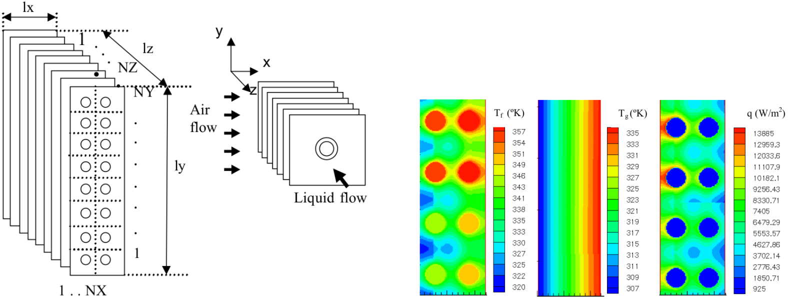
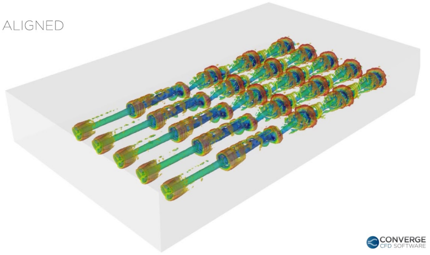


Figure: Simulation of brazed and expanded tube-fin heat exchangers[4].

[4] L. Paniagua et al. (2014). "Large Eddy Simulations (LES) on the Flow and Heat Transfer in a Wall-Bounded Pin Matrix" in *Numerical Heat Transfer, Part B: Fundamentals*.

CFD applications – 4



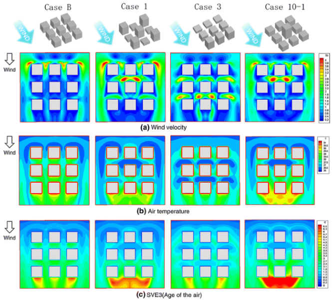


Figure: Simulation of wind plant and array of "buildings" (from the internet).

Poisson's equation in incompressible CFD

Fractional Step Method (FSM)

1. Evaluate the auxiliar vector field $\mathbf{r}(\mathbf{v}^n) := -(\mathbf{v} \cdot \nabla)\mathbf{v} + \nu \Delta \mathbf{v}$
2. Evaluate the predictor velocity $\mathbf{v}^p := \mathbf{v}^n + \Delta t \left(\frac{3}{2}\mathbf{r}(\mathbf{v}^n) - \frac{1}{2}\mathbf{r}(\mathbf{v}^{n-1}) \right)$
3. Obtain the pressure field by solving a **Poisson equation**:

$$\nabla \cdot \left(\frac{1}{\rho} \nabla p^{n+1} \right) = \frac{1}{\Delta t} \nabla \cdot \mathbf{v}^p$$

4. Obtain the new divergence-free velocity $\mathbf{v}^{n+1} = \mathbf{v}^p - \nabla p^{n+1}$

Poisson's equation in incompressible CFD

Fractional Step Method (FSM)

1. Evaluate the auxiliar vector field $\mathbf{r}(\mathbf{v}^n) := -(\mathbf{v} \cdot \nabla)\mathbf{v} + \nu\Delta\mathbf{v}$
2. Evaluate the predictor velocity $\mathbf{v}^p := \mathbf{v}^n + \Delta t \left(\frac{3}{2}\mathbf{r}(\mathbf{v}^n) - \frac{1}{2}\mathbf{r}(\mathbf{v}^{n-1})\right)$
3. Obtain the pressure field by solving a **Poisson equation**:

$$\nabla \cdot \left(\frac{1}{\rho}\nabla p^{n+1}\right) = \frac{1}{\Delta t}\nabla \cdot \mathbf{v}^p$$

4. Obtain the new divergence-free velocity $\mathbf{v}^{n+1} = \mathbf{v}^p - \nabla p^{n+1}$

Poisson's equation for incompressible single-phase flows

- Continuous:

$$\Delta p = \frac{\rho}{\Delta t}\nabla \cdot \mathbf{v}^p$$

Poisson's equation in incompressible CFD

Fractional Step Method (FSM)

1. Evaluate the auxiliar vector field $\mathbf{r}(\mathbf{v}^n) := -(\mathbf{v} \cdot \nabla)\mathbf{v} + \nu \Delta \mathbf{v}$
2. Evaluate the predictor velocity $\mathbf{v}^p := \mathbf{v}^n + \Delta t \left(\frac{3}{2}\mathbf{r}(\mathbf{v}^n) - \frac{1}{2}\mathbf{r}(\mathbf{v}^{n-1}) \right)$
3. Obtain the pressure field by solving a **Poisson equation**:

$$\nabla \cdot \left(\frac{1}{\rho} \nabla p^{n+1} \right) = \frac{1}{\Delta t} \nabla \cdot \mathbf{v}^p$$

4. Obtain the new divergence-free velocity $\mathbf{v}^{n+1} = \mathbf{v}^p - \nabla p^{n+1}$

Poisson's equation for incompressible single-phase flows

- Continuous:

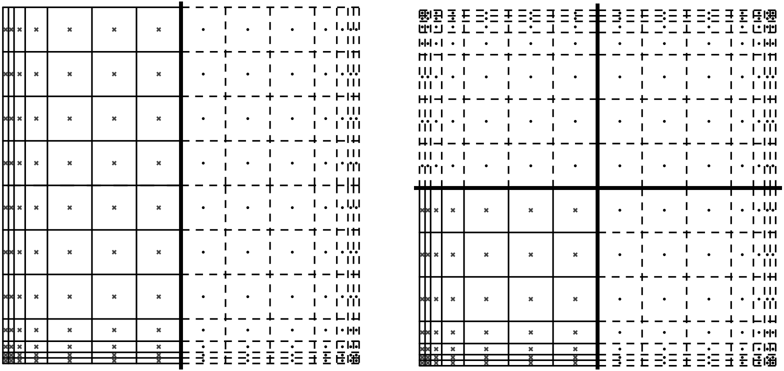
$$\Delta p = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{v}^p$$

- Discrete:

$$\mathsf{L} p_h = \frac{\rho}{\Delta t} \mathsf{M} v_h^p$$

Solving Poisson's equation

Meshes with symmetries

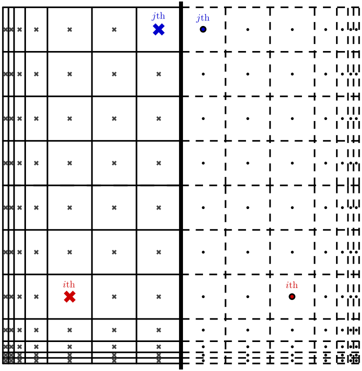


(a) 1 symmetry

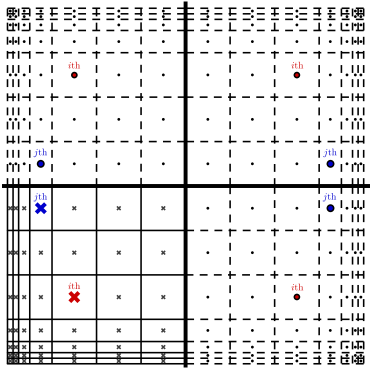
(b) 2 symmetries

Figure: 2D meshes with varying number of symmetries.

"Mirrored" unknowns' ordering



(a) 1 symmetry



(b) 2 symmetries

Figure: "Mirrored" ordering on 2D meshes with a varying no. of symmetries.

Discrete Laplace operator and mesh symmetries

Let L be the discrete Laplace operator arising from a mesh with s symmetries, and let us define the following change of basis:

$$\mathsf{P} = \frac{1}{\sqrt{2^s}} \left[\bigotimes_{i=1}^{p} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right] \otimes \mathbb{I}_{n/2^s} \in \mathbb{R}^{n \times n}$$

Discrete Laplace operator and mesh symmetries

Let L be the discrete Laplace operator arising from a mesh with s symmetries, and let us define the following change of basis:

$$\mathsf{P} = \frac{1}{\sqrt{2^s}} \left[\bigotimes_{i=1}^{p} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right] \otimes \mathbb{I}_{n/2^s} \in \mathbb{R}^{n \times n}$$

Then, thanks to the "mirrored" ordering, P transforms L:

$$\mathsf{L} = \begin{pmatrix} \mathsf{L}_{1-1} & \dots & \mathsf{L}_{1-2^s} \\ \vdots & \ddots & \vdots \\ \mathsf{L}_{2^s-1} & \dots & \mathsf{L}_{2^s-2^s} \end{pmatrix} \in \mathbb{R}^{n \times n}$$

into 2^s decoupled subsystems[5]:

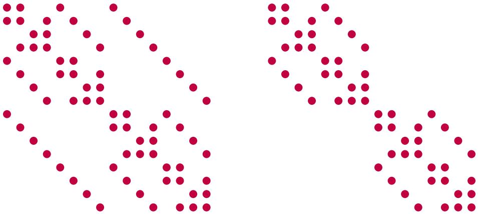
$$\hat{\mathsf{L}} = \begin{pmatrix} \hat{\mathsf{L}}_1 & & \\ & \ddots & \\ & & \hat{\mathsf{L}}_{2^s} \end{pmatrix} \in \mathbb{R}^{n \times n}$$

[e] A. Alsalti-Baldellou et al. (2023). "Exploiting spatial symmetries for solving Poisson's equation", in *Journal of Computational Physics*.

Discrete Laplace operator and mesh symmetries

Let L be the discrete Laplace operator arising from a mesh with s symmetries, and let us define the following change of basis:

$$\mathsf{P} = \frac{1}{\sqrt{2^s}} \left[\bigotimes_{i=1}^{p} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right] \otimes \mathbb{I}_{n/2^s} \in \mathbb{R}^{n \times n}$$



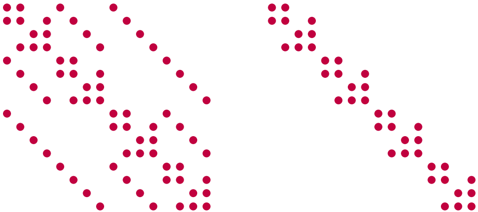
(a) L (b) $\hat{\mathsf{L}} = \mathsf{P}\mathsf{L}\mathsf{P}^{-1}$

Figure: 3D structured mesh exploiting $s = 1$ symmetries.

Discrete Laplace operator and mesh symmetries

Let L be the discrete Laplace operator arising from a mesh with s symmetries, and let us define the following change of basis:

$$\mathsf{P} = \frac{1}{\sqrt{2^s}} \left[\bigotimes_{i=1}^{p} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right] \otimes \mathbb{I}_{n/2^s} \in \mathbb{R}^{n \times n}$$



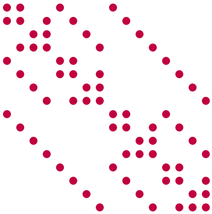
(a) L　　(b) $\hat{\mathsf{L}} = \mathsf{P}\mathsf{L}\mathsf{P}^{-1}$

Figure: 3D structured mesh exploiting $s = 2$ symmetries.

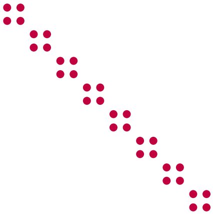
Discrete Laplace operator and mesh symmetries

Let L be the discrete Laplace operator arising from a mesh with s symmetries, and let us define the following change of basis:

$$\mathsf{P} = \frac{1}{\sqrt{2^s}} \left[\bigotimes_{i=1}^{p} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right] \otimes \mathbb{I}_{n/2^s} \in \mathbb{R}^{n \times n}$$



(a) L



(b) $\hat{\mathsf{L}} = \mathsf{P}\mathsf{L}\mathsf{P}^{-1}$

Figure: 3D structured mesh exploiting $s = 3$ symmetries.

Resulting algorithm

Algorithm Poisson solver exploiting s mesh symmetries

1. Transform forward the RHS: $\hat{b} = \mathsf{P}b$
2. Decoupled solution of the 2^s subsystems: $\hat{\mathsf{L}}\hat{x} = \hat{b}$
3. Transform backward the solution: $x = \mathsf{P}^{-1}\hat{x}$

where:

$$\mathsf{P} = \frac{1}{\sqrt{2^s}} \left[\bigotimes_{i=1}^{p} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right] \otimes \mathbb{I}_{n/2^s}, \quad \mathsf{P}^{-1} = \mathsf{P},$$

and Step 2 corresponds to inverting:

$$\begin{pmatrix} \hat{\mathsf{L}}_1 & & \\ & \ddots & \\ & & \hat{\mathsf{L}}_{2^s} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}_1 \\ \vdots \\ \hat{\mathbf{x}}_{2^s} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{b}}_1 \\ \vdots \\ \hat{\mathbf{b}}_{2^s} \end{pmatrix}$$

Iterative solvers and mesh symmetries

The subsystems' smaller size has multiple immediate advantages. Namely:

- A reduction in Poisson solvers' iteration count
- A reduction in Poisson solvers' memory footprint
- An increase in Poisson solvers' arithmetic intensity

Iterative solvers and mesh symmetries

The subsystems' smaller size has multiple immediate advantages. Namely:

- A reduction in Poisson solvers' iteration count
- A reduction in Poisson solvers' memory footprint
- **An increase in Poisson solvers' arithmetic intensity**

In general, $\hat{\mathsf{L}}$ can be split as:

$$\hat{\mathsf{L}} = \cdots = \begin{pmatrix} \mathsf{L}_{\text{inn}} & & \\ & \ddots & \\ & & \mathsf{L}_{\text{inn}} \end{pmatrix} + \begin{pmatrix} \mathsf{L}_{\text{out}}^{(1)} & & \\ & \ddots & \\ & & \mathsf{L}_{\text{out}}^{(2^s)} \end{pmatrix}$$

Iterative solvers and mesh symmetries

The subsystems' smaller size has multiple immediate advantages. Namely:

- A reduction in Poisson solvers' iteration count
- A reduction in Poisson solvers' memory footprint
- **An increase in Poisson solvers' arithmetic intensity**

In general, $\hat{\mathsf{L}}$ can be split as:

$$\hat{\mathsf{L}} = \cdots = \begin{pmatrix} \mathsf{L}_{\text{inn}} & & \\ & \ddots & \\ & & \mathsf{L}_{\text{inn}} \end{pmatrix} + \begin{pmatrix} \mathsf{L}_{\text{out}}^{(1)} & & \\ & \ddots & \\ & & \mathsf{L}_{\text{out}}^{(2^s)} \end{pmatrix}$$

In particular, compact stencils only coupling adjacent nodes result in:

$$\hat{\mathsf{L}}\mathbf{v} = \underbrace{\left(\mathbb{I}_{2^s} \otimes \mathsf{L}_{\text{inn}}\right)\mathbf{v}}_{\substack{\text{Sparse matrix-matrix} \\ \text{product (SpMM)}}} + \underbrace{\operatorname{diag}\left(\mathbf{l}_{\text{out}}\right)\mathbf{v}}_{\substack{\text{Element-wise product} \\ \text{of vectors (axty)}}}$$

Sparse matrix-matrix product

Given $\mathbf{v} \in \mathbb{R}^n$, the products by $\hat{\mathsf{L}}$ can be accelerated by replacing:

$$\texttt{SpMV}: \begin{pmatrix} \mathsf{L}_{\mathsf{inn}} & & \\ & \ddots & \\ & & \mathsf{L}_{\mathsf{inn}} \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{2^s} \end{pmatrix}$$

Sparse matrix-matrix product

Given $\mathbf{v} \in \mathbb{R}^n$, the products by $\hat{\mathsf{L}}$ can be accelerated by replacing:

$$\texttt{SpMV: } \begin{pmatrix} \mathsf{L}_{\mathsf{inn}} & & \\ & \ddots & \\ & & \mathsf{L}_{\mathsf{inn}} \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{2^s} \end{pmatrix} \text{ with } \texttt{SpMM: } \mathsf{L}_{\mathsf{inn}} \left(\mathbf{v}_1 \dots \mathbf{v}_{2^s} \right)$$

Sparse matrix-matrix product

Given $\mathbf{v} \in \mathbb{R}^n$, the products by $\hat{\mathsf{L}}$ can be accelerated by replacing:

$$\texttt{SpMV:} \begin{pmatrix} \mathsf{L}_{\mathsf{inn}} & & \\ & \ddots & \\ & & \mathsf{L}_{\mathsf{inn}} \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{2^s} \end{pmatrix} \text{ with } \texttt{SpMM:}\ \mathsf{L}_{\mathsf{inn}} \left(\mathbf{v}_1 \dots \mathbf{v}_{2^s} \right)$$

Hence:

- $\hat{\mathsf{L}}$'s `SpMV`s can be replaced with a combination of `SpMM` and `axty`
- Since `SpMV` and `SpMM` are memory-bound kernels, `SpMM`'s acceleration equals $\mathsf{I}_{\texttt{SpMM}}/\mathsf{I}_{\texttt{SpMV}}$
- `SpMM` reads $\mathsf{L}_{\mathsf{inn}}$ once, whereas `SpMV` reads $\mathsf{L}_{\mathsf{inn}}$ 2^s times.

SpMM- vs SpMV-based solution of $\hat{L}$'s subsystems

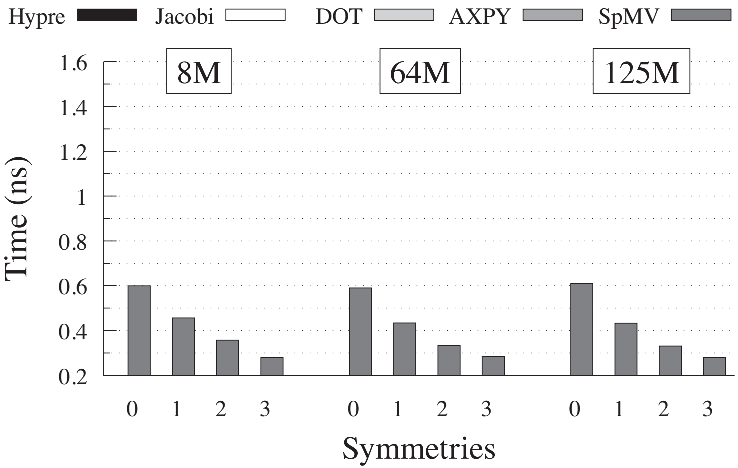


Figure: Normalized time per Jacobi-PCG iteration on 2 Intel Xeon 8160 CPUs.

SpMM- vs SpMV-based solution of $\hat{L}$'s subsystems

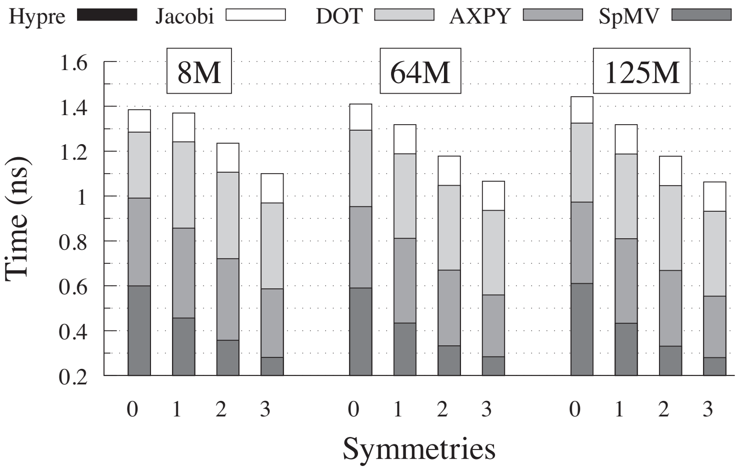


Figure: Normalized time per Jacobi-PCG iteration on 2 Intel Xeon 8160 CPUs.

Summary

Summary:

- The overhead of the two (communication-free) transforms is negligible.

Summary

Summary:

- The overhead of the two (communication-free) transforms is negligible.
- Exploiting symmetries **reduces the setup costs** of the matrices.
- Exploiting symmetries **reduces the memory footprint** of the matrices.
- Exploiting symmetries **reduces the time complexity** of the solvers.

Summary

Summary:

- The overhead of the two (communication-free) transforms is negligible.
- Exploiting symmetries **reduces the setup costs** of the matrices.
- Exploiting symmetries **reduces the memory footprint** of the matrices.
- Exploiting symmetries **reduces the time complexity** of the solvers.
- `SpMM` naturally applies to all operators of the form $\hat{\mathsf{A}} = \mathbb{I}_{2^s} \otimes \mathsf{A}$.
- `SpMM` **increases considerably the I** of *all* the matrix multiplications.

Summary

Summary:

- The overhead of the two (communication-free) transforms is negligible.
- Exploiting symmetries **reduces the setup costs** of the matrices.
- Exploiting symmetries **reduces the memory footprint** of the matrices.
- Exploiting symmetries **reduces the time complexity** of the solvers.
- `SpMM` naturally applies to all operators of the form $\hat{\mathsf{A}} = \mathbb{I}_{2^s} \otimes \mathsf{A}$.
- `SpMM` **increases considerably the I** of *all* the matrix multiplications.

Still missing...

Since all the subsystems are (slightly) different, so are their preconditioners, and `SpMM` cannot be applied with them!

Preconditioning Poisson's equation

Right, left and split preconditioning

Let $A \in \mathbb{R}^n$ and $x, b \in \mathbb{R}^n$. Then, given the linear system $Ax = b$, we can consider the following preconditioning techniques:

Left preconditioning

Given the preconditioner $M^{-1} \simeq A^{-1}$, the left-preconditioned system is:

$$M^{-1}Ax = M^{-1}b$$

Right, left and split preconditioning

Let $A \in \mathbb{R}^n$ and $x, b \in \mathbb{R}^n$. Then, given the linear system $Ax = b$, we can consider the following preconditioning techniques:

Left preconditioning

Given the preconditioner $M^{-1} \simeq A^{-1}$, the left-preconditioned system is:

$$M^{-1}Ax = M^{-1}b$$

Right preconditioning

Given the preconditioner $M^{-1} \simeq A^{-1}$, the right-preconditioned system is:

$$AM^{-1}y = b, \text{ where } Mx = y$$

Right, left and split preconditioning

Let $A \in \mathbb{R}^n$ and $x, b \in \mathbb{R}^n$. Then, given the linear system $Ax = b$, we can consider the following preconditioning techniques:

Left preconditioning

Given the preconditioner $M^{-1} \simeq A^{-1}$, the left-preconditioned system is:

$$M^{-1}Ax = M^{-1}b$$

Right preconditioning

Given the preconditioner $M^{-1} \simeq A^{-1}$, the right-preconditioned system is:

$$AM^{-1}y = b, \text{ where } Mx = y$$

Split preconditioning

Given the preconditioner $M^{-1} = M_1^{-1}M_2^{-1} \simeq A^{-1}$, the split-preconditioned system is:

$$M_1^{-1}AM_2^{-1}y = M_1^{-1}b, \text{ where } M_2x = y$$

Right, left and split preconditioning

Let $A \in \mathbb{R}^n$ and $x, b \in \mathbb{R}^n$. Then, given the linear system $Ax = b$, we can consider the following preconditioning techniques:

Left preconditioning

Given the preconditioner $M^{-1} \simeq A^{-1}$, the left-preconditioned system is:

$$M^{-1}Ax = M^{-1}b$$

Right preconditioning

Given the preconditioner $M^{-1} \simeq A^{-1}$, the right-preconditioned system is:

$$AM^{-1}y = b, \text{ where } Mx = y$$

Split preconditioning

Given the preconditioner $M^{-1} = M_1^{-1}M_2^{-1} \simeq A^{-1}$, the split-preconditioned system is:

$$M_1^{-1}AM_2^{-1}y = M_1^{-1}b, \text{ where } M_2x = y$$

Thus, preconditioning reduces to operations of the type: $y = M^{-1}x$

Low-rank corrections on meshes with symmetries

As we saw, symmetric directions allow decomposing Poisson's equation, $\mathsf{L}x = b$, into 2^s decoupled subsystems with the following structure:

$$\begin{pmatrix} \mathsf{L}_{\mathsf{inn}} + \mathsf{L}_{\mathsf{out}}^{(1)} & & \\ & \ddots & \\ & & \mathsf{L}_{\mathsf{inn}} + \mathsf{L}_{\mathsf{out}}^{(2^s)} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}_1 \\ \vdots \\ \hat{\mathbf{x}}_{2^s} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{b}}_1 \\ \vdots \\ \hat{\mathbf{b}}_{2^s} \end{pmatrix},$$

and such that:

$$\boxed{\operatorname{rank}(\mathsf{L}_{\mathsf{out}}^{(i)}) = n_{\mathsf{ifc}} \ll \operatorname{rank}(\mathsf{L}_{\mathsf{inn}}) = n}$$

Low-rank corrections on meshes with symmetries

As we saw, symmetric directions allow decomposing Poisson's equation, $\mathsf{L}x = b$, into 2^s decoupled subsystems with the following structure:

$$\begin{pmatrix} \mathsf{L}_{\mathsf{inn}} + \mathsf{L}_{\mathsf{out}}^{(1)} & & \\ & \ddots & \\ & & \mathsf{L}_{\mathsf{inn}} + \mathsf{L}_{\mathsf{out}}^{(2^s)} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}_1 \\ \vdots \\ \hat{\mathbf{x}}_{2^s} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{b}}_1 \\ \vdots \\ \hat{\mathbf{b}}_{2^s} \end{pmatrix},$$

and such that:

$$\boxed{\operatorname{rank}(\mathsf{L}_{\mathsf{out}}^{(i)}) = n_{\mathsf{ifc}} \ll \operatorname{rank}(\mathsf{L}_{\mathsf{inn}}) = n}$$

Low-rank corrected preconditioners

Let M_{inn} be a preconditioner for $\mathsf{L}_{\mathsf{inn}}$, *i.e.*, $M_{\mathsf{inn}}^{-1} \simeq \mathsf{L}_{\mathsf{inn}}^{-1}$. Then, we can seek low-rank corrections for M_{inn} such that:

$$\hat{\mathsf{L}}^{-1} \simeq \mathbb{I}_{2^s} \otimes M_{\mathsf{inn}} + \begin{pmatrix} W_k^{(1)} \Theta_k^{(1)} {W_k^{(1)}}^t & & \\ & \ddots & \\ & & W_k^{(2^s)} \Theta_k^{(2^s)} {W_k^{(2^s)}}^t \end{pmatrix},$$

Low-rank corrections on meshes with symmetries

As we saw, symmetric directions allow decomposing Poisson's equation, $\mathsf{L}x = b$, into 2^s decoupled subsystems with the following structure:

$$\begin{pmatrix} \mathsf{L}_{\text{inn}} + \mathsf{L}_{\text{out}}^{(1)} & & \\ & \ddots & \\ & & \mathsf{L}_{\text{inn}} + \mathsf{L}_{\text{out}}^{(2^s)} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}_1 \\ \vdots \\ \hat{\mathbf{x}}_{2^s} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{b}}_1 \\ \vdots \\ \hat{\mathbf{b}}_{2^s} \end{pmatrix},$$

and such that:

$$\boxed{\operatorname{rank}(\mathsf{L}_{\text{out}}^{(i)}) = n_{\text{ifc}} \ll \operatorname{rank}(\mathsf{L}_{\text{inn}}) = n}$$

Low-rank corrected preconditioners

Let M_{inn} be a preconditioner for L_{inn}, *i.e.*, $M_{\text{inn}}^{-1} \simeq \mathsf{L}_{\text{inn}}^{-1}$. Then, we can seek low-rank corrections for M_{inn} such that:

$$\hat{\mathsf{L}}^{-1} \simeq \mathbb{I}_{2^s} \otimes M_{\text{inn}} + \begin{pmatrix} W_k^{(1)} \Theta_k^{(1)} {W_k^{(1)}}^t & & \\ & \ddots & \\ & & W_k^{(2^s)} \Theta_k^{(2^s)} {W_k^{(2^s)}}^t \end{pmatrix},$$

As a result: **lower setup costs**, **decoupled corrections** and `SpMM`!

Low-rank corrections for FSAI – 1

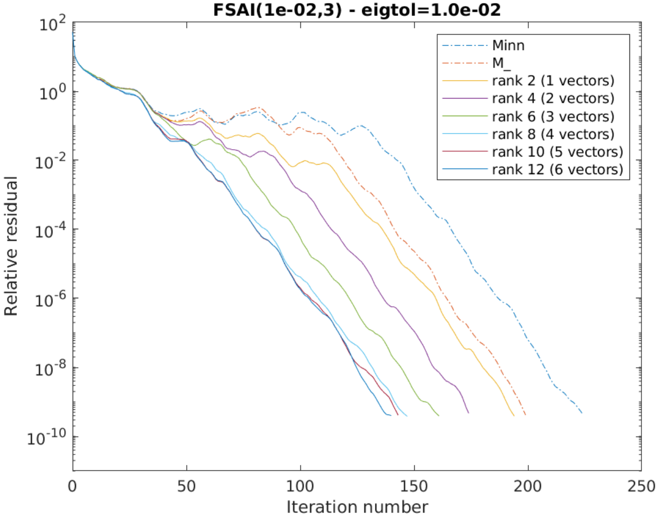


Figure: Low-rank corrected FSAI+PCG on a 100^3 mesh with $s = 1$ symmetries.

Low-rank corrections for FSAI – 2

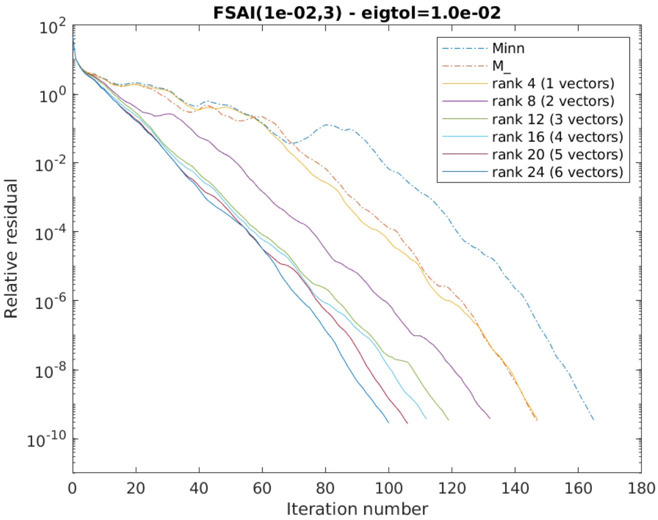


Figure: Low-rank corrected FSAI+PCG on a 100^3 mesh with $s = 2$ symmetries.

Low-rank corrections for FSAI – 3

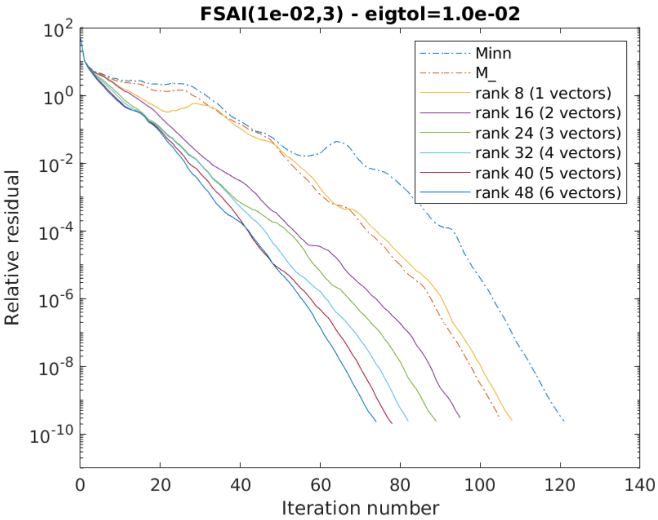


Figure: Low-rank corrected FSAI+PCG on a 100^3 mesh with $s = 3$ symmetries.

Low-rank corrections for FSAI – 4

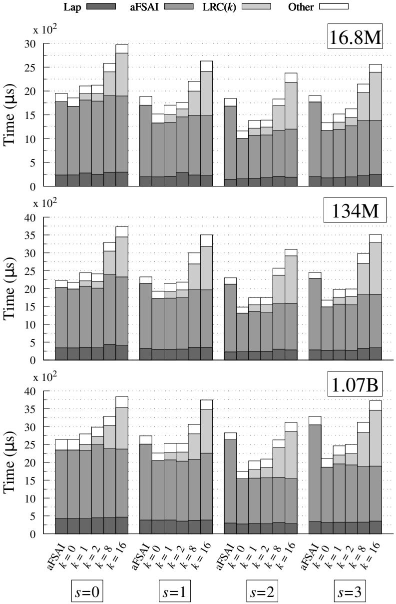


Figure: Normalised time per PCG+LRCFSAI(k) iteration on MARCONI100.

Final goal: `SpMM`-based AMG – 1

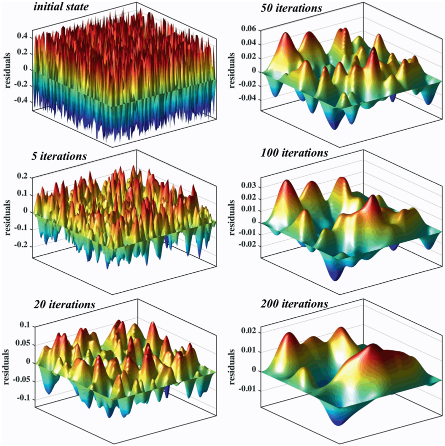


Figure: Single-grid smoothing.

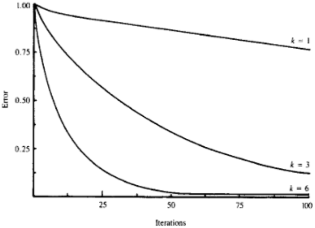


Figure: Eigencomponents' reduction.

Final goal: SpMM-based AMG – 2

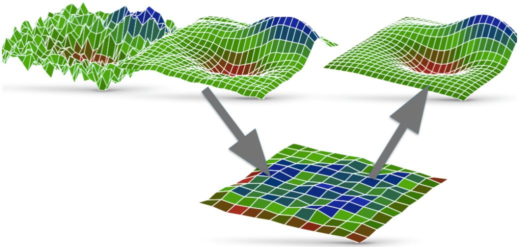


Figure: Two-grid smoothing.

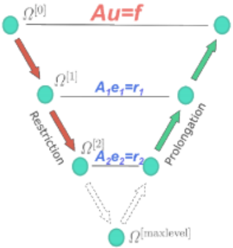


Figure: General V-cycle.

Final goal: SpMM-based AMG – 2

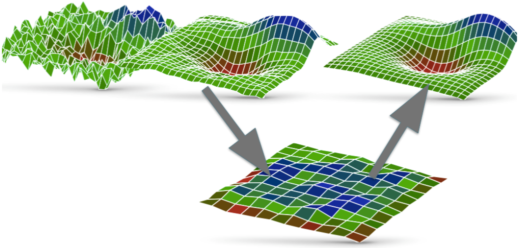


Figure: Two-grid smoothing.

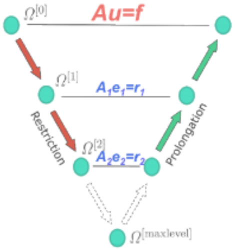


Figure: General V-cycle.

Still missing...

AMG heavily relies on matrix multiplications and, therefore, would particularly benefit from SpMM.

As a result: **lower setup costs** and **significant accelerations**!

Low-rank corrections for AMG – 1

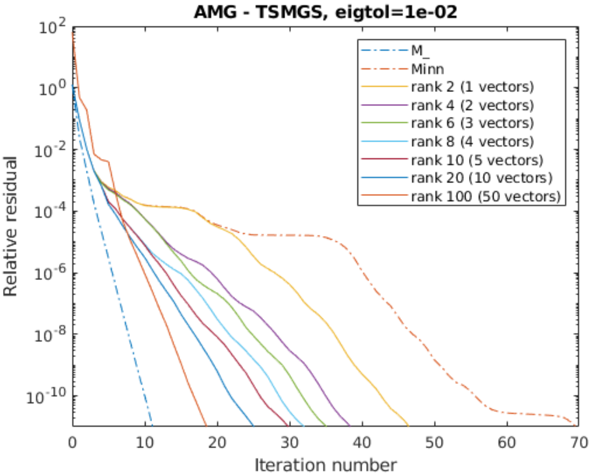


Figure: Low-rank corrected AMG+PCG on a 100^3 mesh with $s = 1$ symmetries.

Low-rank corrections for AMG – 2

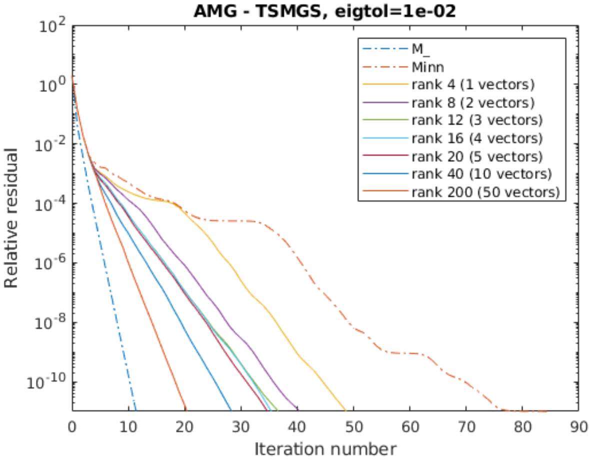


Figure: Low-rank corrected AMG+PCG on a 100^3 mesh with $s = 2$ symmetries.

Low-rank corrections for AMG – 3

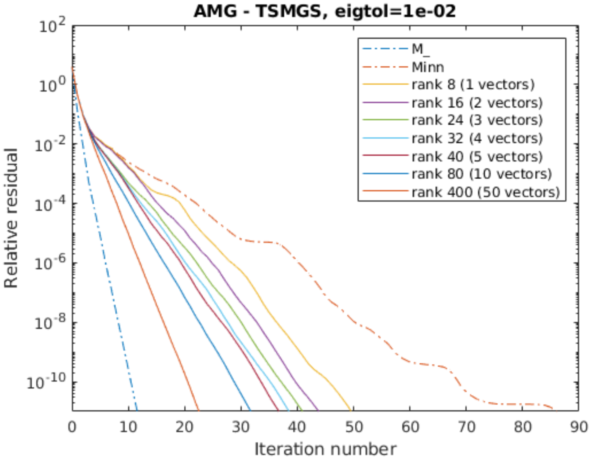
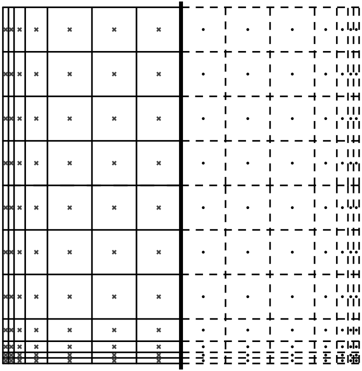
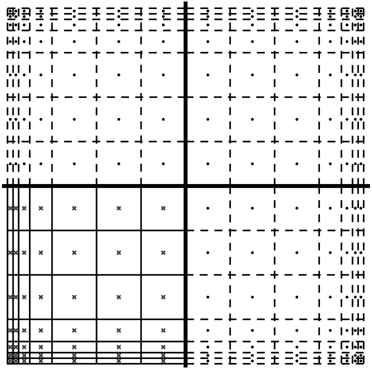


Figure: Low-rank corrected AMG+PCG on a 100^3 mesh with $s = 3$ symmetries.

Meshes with symmetries



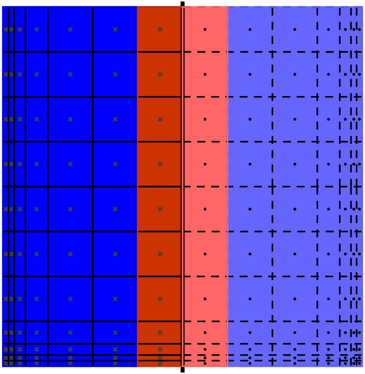
(a) 1 symmetry



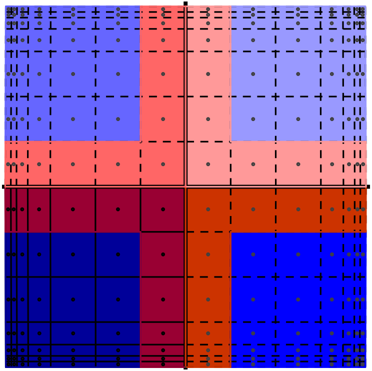
(b) 2 symmetries

Figure: 2D meshes with varying number of symmetries.

"Inner-interface" unknowns' ordering



(a) 1 symmetry



(b) 2 symmetries

Figure: "Inner-interface" ordering on 2D meshes with a varying number of symmetries. Blue: inner nodes, red: interface nodes.

Inn-Ifc discrete Laplace operator – 1

Let L be the discrete Laplace operator arising from a mesh with **1 symmetry** and an "inner-interface" unknowns' ordering. Then:

$$\mathsf{L} = \begin{pmatrix} \bar{A} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix} \in \mathbb{R}^{n \times n},$$

where $\bar{A} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{inn}}}$, $\bar{B} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{ifc}}}$ and $\bar{C} \in \mathbb{R}^{n_{\text{ifc}} \times n_{\text{ifc}}}$ account for the inner-inner, inner-interface and interface-interface couplings, respectively.

Inn-Ifc discrete Laplace operator – 1

Let L be the discrete Laplace operator arising from a mesh with **1 symmetry** and an "inner-interface" unknowns' ordering. Then:

$$\mathsf{L} = \begin{pmatrix} \bar{A} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix} \in \mathbb{R}^{n \times n},$$

where $\bar{A} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{inn}}}$, $\bar{B} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{ifc}}}$ and $\bar{C} \in \mathbb{R}^{n_{\text{ifc}} \times n_{\text{ifc}}}$ account for the inner-inner, inner-interface and interface-interface couplings, respectively. Moreover, they satisfy that:

$$\bar{A} = \begin{pmatrix} A & \\ & A \end{pmatrix}, \quad \bar{B} = \begin{pmatrix} B & \\ & B \end{pmatrix} \text{ and } \bar{C} = \begin{pmatrix} C_{\text{inn}} & C_{\text{out}} \\ C_{\text{out}} & C_{\text{inn}} \end{pmatrix}.$$

Inn-Ifc discrete Laplace operator – 1

Let L be the discrete Laplace operator arising from a mesh with **1 symmetry** and an "inner-interface" unknowns' ordering. Then:

$$\mathsf{L} = \begin{pmatrix} \bar{A} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix} \in \mathbb{R}^{n \times n},$$

where $\bar{A} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{inn}}}$, $\bar{B} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{ifc}}}$ and $\bar{C} \in \mathbb{R}^{n_{\text{ifc}} \times n_{\text{ifc}}}$ account for the inner-inner, inner-interface and interface-interface couplings, respectively. Moreover, they satisfy that:

$$\bar{A} = \begin{pmatrix} A & \\ & A \end{pmatrix}, \quad \bar{B} = \begin{pmatrix} B & \\ & B \end{pmatrix} \text{ and } \bar{C} = \begin{pmatrix} C_{\text{inn}} & C_{\text{out}} \\ C_{\text{out}} & C_{\text{inn}} \end{pmatrix}.$$

Eureka!

Given a **geometry repeated n_b times**, a (non-mirrored) "inner-interface" ordering leads to the same Laplacian but only satisfying $\bar{A} = \mathbb{I}_{n_b} \otimes A$.

Inn-Ifc discrete Laplace operator – 1

Let L be the discrete Laplace operator arising from a mesh with **1 symmetry** and an "inner-interface" unknowns' ordering. Then:

$$\mathsf{L} = \begin{pmatrix} \bar{A} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix} \in \mathbb{R}^{n \times n},$$

where $\bar{A} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{inn}}}$, $\bar{B} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{ifc}}}$ and $\bar{C} \in \mathbb{R}^{n_{\text{ifc}} \times n_{\text{ifc}}}$ account for the inner-inner, inner-interface and interface-interface couplings, respectively. Moreover, they satisfy that:

$$\bar{A} = \begin{pmatrix} A & \\ & A \end{pmatrix}, \quad \bar{B} = \begin{pmatrix} B & \\ & B \end{pmatrix} \text{ and } \bar{C} = \begin{pmatrix} C_{\text{inn}} & C_{\text{out}} \\ C_{\text{out}} & C_{\text{inn}} \end{pmatrix}.$$

Eureka!

Given a **geometry repeated n_b times**, a (non-mirrored) "inner-interface" ordering leads to the same Laplacian but only satisfying $\bar{A} = \mathbb{I}_{n_b} \otimes A$.

Additionally, the "inner-interface" ordering works with symmetric domains with **non-symmetric boundary conditions**!

Overview of the AMGR framework – 1

We had that n_b repeated geometries (with s symmetries, $n_b = 2^s$) lead to:

$$\mathsf{L} = \begin{pmatrix} \bar{A} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix}, \text{ where } \bar{A} = \begin{pmatrix} A & & \\ & \ddots & \\ & & A \end{pmatrix}.$$

Then, we will consider a two-level AMG with the following fine-level smoother:

$$M_{\mathsf{L}} = \begin{pmatrix} \mathbb{I}_{n_b} \otimes M_A & \\ & M_{\bar{C}} \end{pmatrix},$$

Overview of the AMGR framework – 1

We had that n_b repeated geometries (with s symmetries, $n_b = 2^s$) lead to:

$$\mathsf{L} = \begin{pmatrix} \bar{A} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix}, \text{ where } \bar{A} = \begin{pmatrix} A & & \\ & \ddots & \\ & & A \end{pmatrix}.$$

Then, we will consider a two-level AMG with the following fine-level smoother:

$$M_{\mathsf{L}} = \begin{pmatrix} \mathbb{I}_{n_b} \otimes M_A & \\ & M_{\bar{C}} \end{pmatrix},$$

the following prolongation:

$$P = \begin{pmatrix} W \\ \mathbb{I}_{n_{\mathsf{ifc}}} \end{pmatrix} \in \mathbb{R}^{n \times n_{\mathsf{ifc}}},$$

Overview of the AMGR framework – 1

We had that n_b repeated geometries (with s symmetries, $n_b = 2^s$) lead to:

$$\mathsf{L} = \begin{pmatrix} \bar{A} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix}, \text{ where } \bar{A} = \begin{pmatrix} A & & \\ & \ddots & \\ & & A \end{pmatrix}.$$

Then, we will consider a two-level AMG with the following fine-level smoother:

$$M_{\mathsf{L}} = \begin{pmatrix} \mathbb{I}_{n_b} \otimes M_A & \\ & M_{\bar{C}} \end{pmatrix},$$

the following prolongation:

$$P = \begin{pmatrix} W \\ \mathbb{I}_{n_{\mathsf{ifc}}} \end{pmatrix} \in \mathbb{R}^{n \times n_{\mathsf{ifc}}},$$

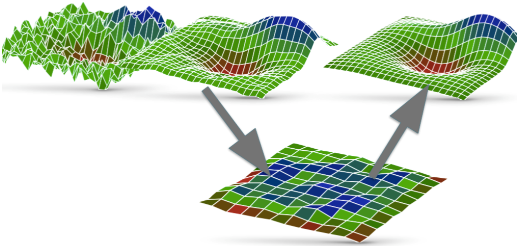
and the following coarse-level operator:

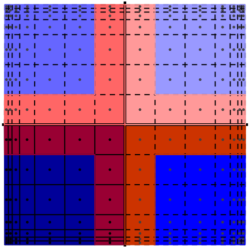
$$\mathsf{L}_c = P^T \mathsf{L} P \in \mathbb{R}^{n_{\mathsf{ifc}} \times n_{\mathsf{ifc}}}.$$

Overview of the AMGR framework – 2

In summary, our two-level AMGR will consist of:

$$M_{\mathsf{L}} = \begin{pmatrix} \mathbb{I}_{n_b} \otimes M_A & \\ & M_{\bar{C}} \end{pmatrix}, \quad P = \begin{pmatrix} W \\ \mathbb{I}_{n_{\text{ifc}}} \end{pmatrix}, \quad \mathsf{L}_c = P^T \mathsf{L} P, \quad \text{and } M_{\mathsf{L}_c} = \mathsf{AMG}_{\mathsf{L}_c}.$$





Concluding remarks

Conclusions

Summary:

- AMGR applies to both **mirrored and repeated geometries** regardless of the boundary conditions.

Conclusions

Summary:

- AMGR applies to both **mirrored and repeated geometries** regardless of the boundary conditions.
- AMGR preconditioner does not require decoupling Poisson's equation.
- Despite its aggressive coarsening, AMGR converges as the standard AMG.

Conclusions

Summary:

- AMGR applies to both **mirrored and repeated geometries** regardless of the boundary conditions.
- AMGR preconditioner does not require decoupling Poisson's equation.
- Despite its aggressive coarsening, AMGR converges as the standard AMG.
- AMGR **reduces the setup costs** of AMG.
- AMGR **reduces the memory footprint** of AMG.
- AMGR **increases the arithmetic intensity** of AMG.

Conclusions

Summary:

- AMGR applies to both **mirrored and repeated geometries** regardless of the boundary conditions.
- AMGR preconditioner does not require decoupling Poisson's equation.
- Despite its aggressive coarsening, AMGR converges as the standard AMG.
- AMGR **reduces the setup costs** of AMG.
- AMGR **reduces the memory footprint** of AMG.
- AMGR **increases the arithmetic intensity** of AMG.

Ongoing work:

- Test AMGR on real CFD and structural problems.
- Test `SpMM` in simulations presenting symmetries or repeated geometries.

Thanks for your attention!