An AMG reduction framework for domains with symmetries

Àdel Alsalti-Baldellou[1,2] Carlo Janna[2,3] Andrea Franceschini[2]
Gianluca Mazzucco[2] Xavier Álvarez-Farré[4] F. Xavier Trias[1]

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

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

[3]M3E s.r.l.,
http://www.m3eweb.it/

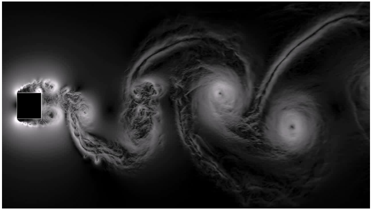
[4]High-Performance Computing and Visualization Team, SURF

SIAM Conference on Parallel Processing for Scientific Computing (PP24)
March 5-8 2024 – Baltimore, USA

Index

Context of the work

CFD applications – 1



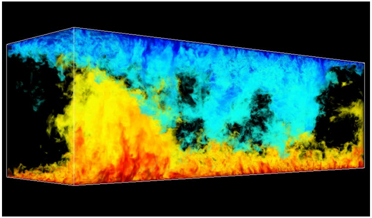


Figure: Simulation of flow around a square cylinder and Rayleigh-Bénard convection.

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

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

CFD applications – 2

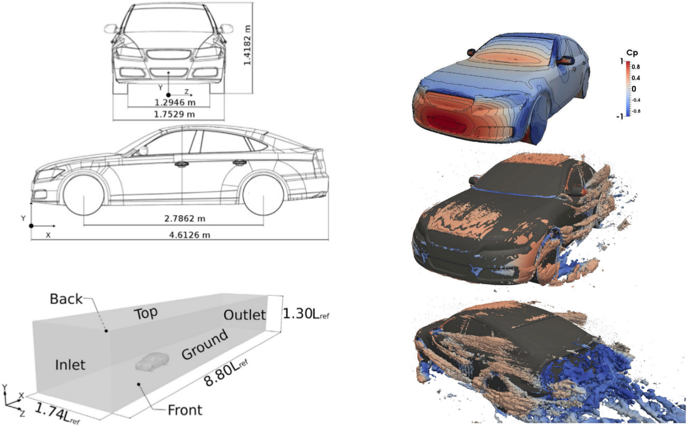


Figure: Simulation of turbulent flow over the DrivAer fastback vehicle model.

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

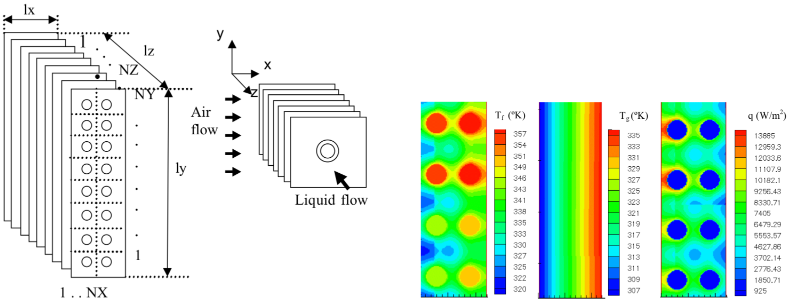


Figure: Simulation of brazed and expanded tube-fin heat exchangers.

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

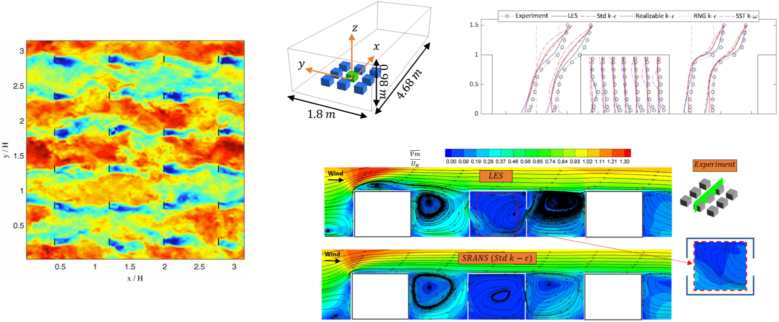


Figure: Simulation of wind plant and array of “buildings”.

M. Calaf et al. (2010). “Large eddy simulation study of fully developed wind-turbine array boundary layers” in *Physics of Fluids*.

P. A. Mirzaei (2021). “CFD modeling of micro and urban climates: Problems to be solved in the new decade” in *Sustainable Cities and Society*.

Poisson's equation in incompressible CFD

Fractional Step Method (FSM)

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

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

4. Obtain the new divergence-free velocity $\boldsymbol{u}^{n+1} = \boldsymbol{u}^p - \Delta t \nabla p^{n+1}$

Poisson's equation in incompressible CFD

Fractional Step Method (FSM)

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

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

4. Obtain the new divergence-free velocity $\boldsymbol{u}^{n+1} = \boldsymbol{u}^p - \Delta t \nabla p^{n+1}$

Poisson's equation for incompressible single-phase flows

- Continuous:

$$\Delta p = \frac{\rho}{\Delta t}\nabla \cdot \boldsymbol{u}^p$$

Poisson's equation in incompressible CFD

Fractional Step Method (FSM)

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

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

4. Obtain the new divergence-free velocity $\boldsymbol{u}^{n+1} = \boldsymbol{u}^p - \Delta t \nabla p^{n+1}$

Poisson's equation for incompressible single-phase flows

- Continuous:

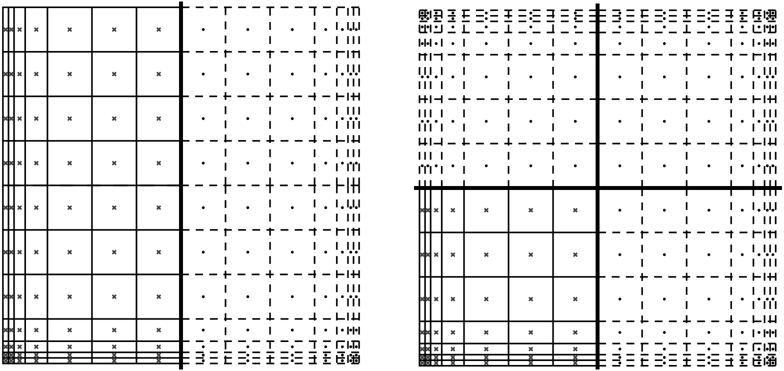
$$\Delta p = \frac{\rho}{\Delta t} \nabla \cdot \boldsymbol{u}^p$$

- Discrete:

$$Ax = b$$

Mesh symmetries and SpMM

Meshes with symmetries – 1

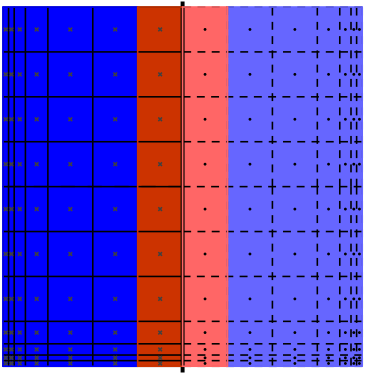


(a) 1 symmetry

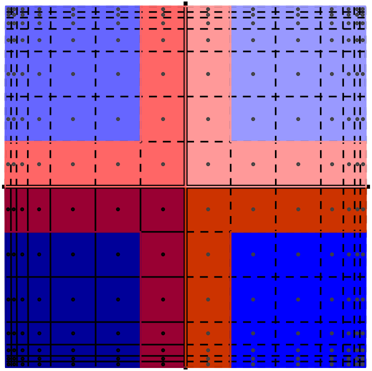
(b) 2 symmetries

Figure: 2D meshes with varying number of symmetries.

Meshes with symmetries – 2



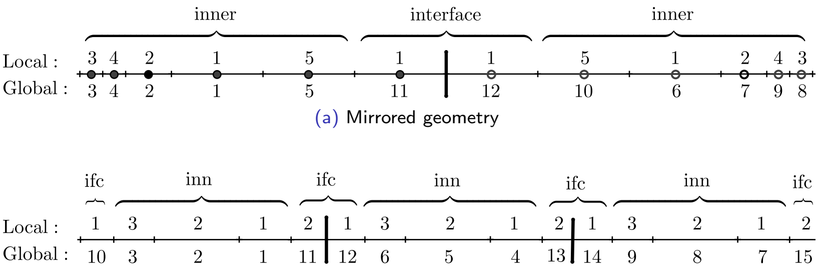
(a) 1 symmetry



(b) 2 symmetries

Figure: 2D meshes with varying number of symmetries. Blue: inner nodes, red: interface nodes.

Meshes with symmetries – 3

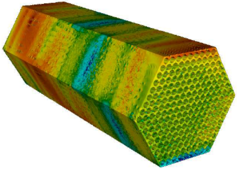


(a) Mirrored geometry

(b) Repeated geometry

Figure: 1D meshes with a random mirrored/repeated ordering.

Using SpMM throughout the simulations – 1



E. Merzari et al. (2020). "Wall resolved large eddy simulation of reactor core flows with the spectral element method", in *Nuclear Engineering and Design*.

Using SpMM throughout the simulations – 1

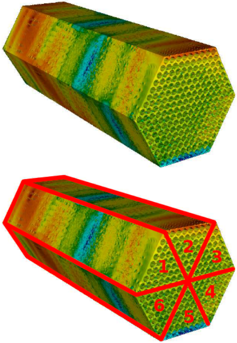


Figure: Pressure field on a 217-pin rod bundle.

E. Merzari et al. (2020). "Wall resolved large eddy simulation of reactor core flows with the spectral element method", in *Nuclear Engineering and Design*.

Using SpMM throughout the simulations – 1

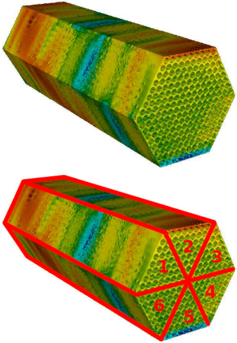


Figure: Pressure field on a 217-pin rod bundle.

Applying an "inner-interface" ordering makes the discrete Laplacian satisfy:

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

where $\bar{K} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{inn}}}$, $\bar{B} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{ifc}}}$, $\bar{C} \in \mathbb{R}^{n_{\text{ifc}} \times n_{\text{ifc}}}$.

E. Merzari et al. (2020). "Wall resolved large eddy simulation of reactor core flows with the spectral element method", in *Nuclear Engineering and Design*.

Using SpMM throughout the simulations – 1

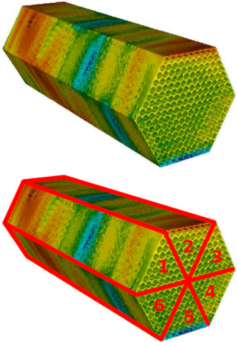


Figure: Pressure field on a 217-pin rod bundle.

Applying an "inner-interface" ordering makes the discrete Laplacian satisfy:

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

where $\bar{K} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{inn}}}$, $\bar{B} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{ifc}}}$, $\bar{C} \in \mathbb{R}^{n_{\text{ifc}} \times n_{\text{ifc}}}$.

Then, thanks to the mirrored/repeated ordering:

$$\bar{K} = \mathbb{I}_6 \otimes K \text{ and } \bar{B} = \mathbb{I}_6 \otimes B.$$

E. Merzari et al. (2020). "Wall resolved large eddy simulation of reactor core flows with the spectral element method", in *Nuclear Engineering and Design*.

Using SpMM throughout the simulations – 2

On a domain with n_b repeated/mirrored subdomains, virtually all operators satisfy structures equivalent to:

$$\bar{H} = \mathbb{I}_{n_b} \otimes H \in \mathbb{R}^{n \times m} \text{ s.t. } H \in \mathbb{R}^{n/n_b \times m/n_b}.$$

Using SpMM throughout the simulations – 2

On a domain with n_b repeated/mirrored subdomains, virtually all operators satisfy structures equivalent to:

$$\bar{H} = \mathbb{I}_{n_b} \otimes H \in \mathbb{R}^{n \times m} \text{ s.t. } H \in \mathbb{R}^{n/n_b \times m/n_b}.$$

Then, given $x \in \mathbb{R}^m$, the products by $\bar{H}$ can be accelerated by replacing:

$$\texttt{SpMV:} \begin{pmatrix} H & & \\ & \ddots & \\ & & H \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{n_b} \end{pmatrix} \text{ with } \texttt{SpMM:}\ H\left(x_1 \dots x_{n_b}\right)$$

Using SpMM throughout the simulations – 2

On a domain with n_b repeated/mirrored subdomains, virtually all operators satisfy structures equivalent to:

$$\bar{H} = \mathbb{I}_{n_b} \otimes H \in \mathbb{R}^{n \times m} \text{ s.t. } H \in \mathbb{R}^{n/n_b \times m/n_b}.$$

Then, given $x \in \mathbb{R}^m$, the products by $\bar{H}$ can be accelerated by replacing:

$$\texttt{SpMV:} \begin{pmatrix} H & & \\ & \ddots & \\ & & H \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{n_b} \end{pmatrix} \text{ with } \texttt{SpMM:} \; H \left(x_1 \dots x_{n_b} \right)$$

SpMM vs SpMV

- SpMV reads H n_b times, whereas SpMM once
- $\bar{H}$ takes n_b times more memory than H

Algebraic Multigrid reduction framework

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: $y = M^{-1}x$

AMGR preconditioner

AMGR relies on the following **prolongation**:

$$P := \begin{pmatrix} \bar{W} \\ \mathbb{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c} \text{ s.t. } \bar{W} \in \mathbb{R}^{n_f \times n_c} \text{ and } \mathbb{I}_{n_c} \in \mathbb{R}^{n_c \times n_c},$$

where n_f and n_c are the number of fine and coarse nodes.

C. Janna and M. Ferronato (2011). "Adaptive pattern research for block FSAI preconditioning" in *SIAM Journal on Scientific Computing*.

G. Isotton et al. (2021). "Chronos: A general purpose AMG solver for high performance computing" in *SIAM Journal on Scientific Computing*.

AMGR preconditioner

AMGR relies on the following **prolongation**:

$$P := \begin{pmatrix} \bar{W} \\ \mathbb{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c} \text{ s.t. } \bar{W} \in \mathbb{R}^{n_f \times n_c} \text{ and } \mathbb{I}_{n_c} \in \mathbb{R}^{n_c \times n_c},$$

where n_f and n_c are the number of fine and coarse nodes.

Then, we apply a standard AMG to the **reduced operator**:

$$A_c := P^T \begin{pmatrix} \bar{K} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix} P$$

C. Janna and M. Ferronato (2011). "Adaptive pattern research for block FSAI preconditioning" in *SIAM Journal on Scientific Computing*.

G. Isotton et al. (2021). "Chronos: A general purpose AMG solver for high performance computing" in *SIAM Journal on Scientific Computing*.

AMGR preconditioner

AMGR relies on the following **prolongation**:

$$P := \begin{pmatrix} \bar{W} \\ \mathbb{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c} \text{ s.t. } \bar{W} \in \mathbb{R}^{n_f \times n_c} \text{ and } \mathbb{I}_{n_c} \in \mathbb{R}^{n_c \times n_c},$$

where n_f and n_c are the number of fine and coarse nodes.

Then, we apply a standard AMG to the **reduced operator**:

$$A_c := P^T \begin{pmatrix} \bar{K} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix} P \overset{\substack{n_f = n_{\text{inn}} \\ n_c = n_{\text{ifc}}}}{=} \bar{W}^T \bar{K} W + \bar{W}^T \bar{B} + \bar{B}^T \bar{W} + \bar{C}.$$

C. Janna and M. Ferronato (2011). "Adaptive pattern research for block FSAI preconditioning" in *SIAM Journal on Scientific Computing*.

G. Isotton et al. (2021). "Chronos: A general purpose AMG solver for high performance computing" in *SIAM Journal on Scientific Computing*.

AMGR preconditioner

AMGR relies on the following **prolongation**:

$$P := \begin{pmatrix} \bar{W} \\ \mathbb{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c} \text{ s.t. } \bar{W} \in \mathbb{R}^{n_f \times n_c} \text{ and } \mathbb{I}_{n_c} \in \mathbb{R}^{n_c \times n_c},$$

where n_f and n_c are the number of fine and coarse nodes.

Then, we apply a standard AMG to the **reduced operator**:

$$A_c := P^T \begin{pmatrix} \bar{K} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix} P \overset{\substack{n_f = n_{\text{inn}} \\ n_c = n_{\text{ifc}}}}{=} \bar{W}^T \bar{K} W + \bar{W}^T \bar{B} + \bar{B}^T \bar{W} + \bar{C}.$$

The fastest **coarsening** is $n_c = n_{\text{ifc}}$, but it results in excessive f-c distances. Hence, to allow for an accurate interpolation, we turn inner nodes into coarse:

- Pick a strength of connection measure
- Filter the resulting adjacency graph, T
- Compute a maximum independent set on T^k.

C. Janna and M. Ferronato (2011). "Adaptive pattern research for block FSAI preconditioning" in *SIAM Journal on Scientific Computing*.

G. Isotton et al. (2021). "Chronos: A general purpose AMG solver for high performance computing" in *SIAM Journal on Scientific Computing*.

AMGR preconditioner

AMGR relies on the following **prolongation**:

$$P := \begin{pmatrix} \bar{W} \\ \mathbb{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c} \text{ s.t. } \bar{W} \in \mathbb{R}^{n_f \times n_c} \text{ and } \mathbb{I}_{n_c} \in \mathbb{R}^{n_c \times n_c},$$

where n_f and n_c are the number of fine and coarse nodes.

Then, we apply a standard AMG to the **reduced operator**:

$$A_c := P^T \begin{pmatrix} \bar{K} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix} P \overset{\substack{n_f = n_{\text{inn}} \\ n_c = n_{\text{ifc}}}}{=} \bar{W}^T \bar{K} W + \bar{W}^T \bar{B} + \bar{B}^T \bar{W} + \bar{C}.$$

The fastest **coarsening** is $n_c = n_{\text{ifc}}$, but it results in excessive f-c distances.

Finally, the **top-level smoother** is:

$$M := \begin{pmatrix} M_{\bar{K}} & \\ & M_{\bar{C}} \end{pmatrix} \in \mathbb{R}^{n \times n}$$

C. Janna and M. Ferronato (2011). "Adaptive pattern research for block FSAI preconditioning" in *SIAM Journal on Scientific Computing*.

G. Isotton et al. (2021). "Chronos: A general purpose AMG solver for high performance computing" in *SIAM Journal on Scientific Computing*.

AMGR preconditioner

AMGR relies on the following **prolongation**:

$$P := \begin{pmatrix} \bar{W} \\ \mathbb{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c} \text{ s.t. } \bar{W} \in \mathbb{R}^{n_f \times n_c} \text{ and } \mathbb{I}_{n_c} \in \mathbb{R}^{n_c \times n_c},$$

where n_f and n_c are the number of fine and coarse nodes.

Then, we apply a standard AMG to the **reduced operator**:

$$A_c := P^T \begin{pmatrix} \bar{K} & \bar{B} \\ \bar{B}^t & \bar{C} \end{pmatrix} P \overset{\substack{n_f = n_{\text{inn}} \\ n_c = n_{\text{ifc}}}}{=} \bar{W}^T \bar{K} W + \bar{W}^T \bar{B} + \bar{B}^T \bar{W} + \bar{C}.$$

The fastest **coarsening** is $n_c = n_{\text{ifc}}$, but it results in excessive f-c distances.

Finally, the **top-level smoother** is:

$$M := \begin{pmatrix} M_{\bar{K}} & \\ & M_{\bar{C}} \end{pmatrix} \in \mathbb{R}^{n \times n} \text{ s.t. } M_{\bar{K}} = \mathbb{I}_{n_b} \otimes M_K.$$

C. Janna and M. Ferronato (2011). "Adaptive pattern research for block FSAI preconditioning" in *SIAM Journal on Scientific Computing*.

G. Isotton et al. (2021). "Chronos: A general purpose AMG solver for high performance computing" in *SIAM Journal on Scientific Computing*.

Numerical experiments: DrivAer fastback

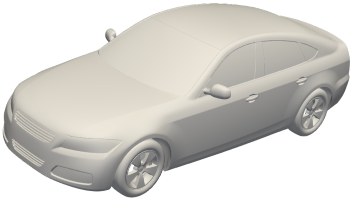


Table: DrivAer car with 106.4M DOFs on five JFF nodes (2x Intel Xeon 6230).

preconditioner	n_b	coarsening ratio	avg nnzr	its	t-sol (s)	speed-up
AMG	1	0.36	14.7			
AMGR	2	0.14	37.4			

A. I. Heft et al. (2012). "Introduction of a new realistic generic car model for aerodynamic investigations" in *SAE International*.

Numerical experiments: DrivAer fastback

Table: DrivAer car with 106.4M DOFs on five JFF nodes (2x Intel Xeon 6230).

preconditioner	n_b	coarsening ratio	avg nnzr	its	t-sol (s)	speed-up
AMG	1	0.36	14.7	26		
AMGR	2	0.14	37.4	26		

A. I. Heft et al. (2012). "Introduction of a new realistic generic car model for aerodynamic investigations" in *SAE International*.

Numerical experiments: DrivAer fastback

Table: DrivAer car with 106.4M DOFs on five JFF nodes (2x Intel Xeon 6230).

preconditioner	n_b	coarsening ratio	avg nnzr	its	t-sol (s)	speed-up
AMG	1	0.36	14.7	26	7.71	1.00
AMGR	2	0.14	37.4	26	5.39	1.43

A. I. Heft et al. (2012). "Introduction of a new realistic generic car model for aerodynamic investigations" in *SAE International*.

Numerical experiments: DrivAer fastback

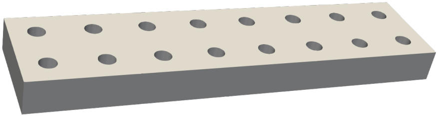


Table: Heat exchanger with 18.4M DOFs on two JFF nodes (2x Intel Xeon 6230).

preconditioner	n_b	coarsening ratio	avg nnzr	its	t-sol (s)	speed-up
AMG	1	0.36	14.5			
AMGR	2	0.14	37.5			
AMGR	4	0.15	37.4			
AMGR	8	0.15	37.6			

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*.

Numerical experiments: DrivAer fastback

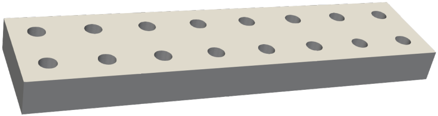


Table: Heat exchanger with 18.4M DOFs on two JFF nodes (2x Intel Xeon 6230).

preconditioner	n_b	coarsening ratio	avg nnzr	its	t-sol (s)	speed-up
AMG	1	0.36	14.5	20		
AMGR	2	0.14	37.5	19		
AMGR	4	0.15	37.4	19		
AMGR	8	0.15	37.6	18		

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*.

Numerical experiments: DrivAer fastback

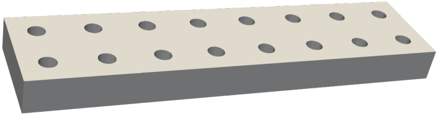


Table: Heat exchanger with 18.4M DOFs on two JFF nodes (2x Intel Xeon 6230).

preconditioner	n_b	coarsening ratio	avg nnzr	its	t-sol (s)	speed-up
AMG	1	0.36	14.5	20	1.54	1.00
AMGR	2	0.14	37.5	19	1.12	1.38
AMGR	4	0.15	37.4	19	1.03	1.50
AMGR	8	0.15	37.6	18	0.91	1.68

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*.

Concluding remarks

Conclusions

Summary:

- Exploiting symmetries reduces the setup costs of the matrices.
- Exploiting symmetries reduces the memory footprint of the matrices.

Conclusions

Summary:

- Exploiting symmetries reduces the setup costs of the matrices.
- Exploiting symmetries reduces the memory footprint of the matrices.
- `SpMM` naturally applies to all operators of the form $\bar{H} = \mathbb{I}_{n_b} \otimes H$.
- `SpMM` makes matrix multiplications considerably more compute-intensive.

Conclusions

Summary:

- Exploiting symmetries reduces the setup costs of the matrices.
- Exploiting symmetries reduces the memory footprint of the matrices.
- `SpMM` naturally applies to all operators of the form $\bar{H} = \mathbb{I}_{n_b} \otimes H$.
- `SpMM` makes matrix multiplications considerably more compute-intensive.
- AMGR reduces the **memory footprint** of the top-level smoother.
- AMGR reduces the **setup costs** of the top-level smoother.

Conclusions

Summary:

- Exploiting symmetries reduces the setup costs of the matrices.
- Exploiting symmetries reduces the memory footprint of the matrices.
- `SpMM` naturally applies to all operators of the form $\bar{H} = \mathbb{I}_{n_b} \otimes H$.
- `SpMM` makes matrix multiplications considerably more compute-intensive.
- AMGR reduces the **memory footprint** of the top-level smoother.
- AMGR reduces the **setup costs** of the top-level smoother.
- AMGR does not harm AMG's convergence.
- AMGR results in up to **1.68x overall speedups**.

Conclusions

Summary:

- Exploiting symmetries reduces the setup costs of the matrices.
- Exploiting symmetries reduces the memory footprint of the matrices.
- `SpMM` naturally applies to all operators of the form $\bar{H} = \mathbb{I}_{n_b} \otimes H$.
- `SpMM` makes matrix multiplications considerably more compute-intensive.
- AMGR reduces the **memory footprint** of the top-level smoother.
- AMGR reduces the **setup costs** of the top-level smoother.
- AMGR does not harm AMG's convergence.
- AMGR results in up to **1.68x overall speedups**.

Ongoing work:

- Test AMGR on denser problems.

Thanks for your attention!