The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

LIGHTWEIGHT AMG PRECONDITIONERS FOR CFD SIMULATIONS ON SYMMETRIC DOMAINS

Àdel Alsalti-Baldellou[1,2], Carlo Janna[3], Xavier Álvarez-Farré[4], and F. Xavier Trias[5]

[1] Department of Information Engineering,
University of Padova, Via Giovanni Gradenigo, 6b, 35131 Padova PD, Italy
adel.alsaltibaldellou@unipd.it

[2] Termo Fluids SL
Carrer de Magí Colet, 8, 08204 Sabadell (Barcelona), Spain

[3] Department of Civil, Environmental and Architectural Engineering,
University of Padova, Via Francesco Marzolo, 9, 35131 Padova PD, Italy

[4] High-Performance Computing Team, SURF
Science Park 140, 1098 XG Amsterdam, The Netherlands

[5] Heat and Mass Transfer Technological Center, Polytechnic University of Catalonia
Carrer de Colom, 11, 08222 Terrassa (Barcelona), Spain

Key words: Spatial symmetries, Multigrid reduction, AMG, Poisson's equation, SpMM

Summary. Divergence constraints appear in the governing equations of many physical phenomena, often leading to a Poisson equation whose solution is one of the most challenging parts of incompressible CFD codes. Algebraic Multigrid (AMG) is likely the most effective preconditioner for Poisson's equation. Its effectiveness stems from the combined roles of the smoother, which dampens high-frequency error components, and the coarse-grid correction, which reduces low-frequency modes. This work presents an AMG reduction framework to leverage spatial symmetries, often present in academic and industrial configurations, for making AMG more compute-intensive. In particular, we introduce an aggressive coarsening to the top level of the multigrid hierarchy, reducing the setup, memory footprint and application costs of the top-level smoother. Numerical experiments on a model problem confirm the advantages of the resulting preconditioner.

1 INTRODUCTION

Divergence constraints are commonly found in the governing equations of many physical problems, usually adhering to fundamental conservation principles like mass or electrical charge conservation, and usually result in a Poisson equation for a scalar potential, making its solution play a critical role in numerous scientific and engineering fields, including computational fluid dynamics (CFD), linear elasticity and electrostatics. Indeed, it is typically one of the most challenging and costly parts of scientific simulation codes.

Over the years, the most common approach to solving Poisson's equation has been through iterative methods relying on Krylov subspaces [1]. These methods are easy to implement and parallelise, as they only require basic linear algebra operations such as matrix by vector products, scalar products, and vector updates. However, they are very sensitive to the properties of the problem at hand and generally require potent preconditioners to be effective [2, 3]. Preconditioners based on incomplete factorisations were widely used in the beginning of numerical linear algebra. Nevertheless, the rise of parallel computers caused them to lose popularity due to their sequential nature. Instead, preconditioners based on approximate inverses gained attraction because their application solely requires the easily parallelisable sparse matrix-vector product (`SpMV`). However, they are not optimal because, as the problem grows, so does the number of iterations required to attain the same accuracy.

This problem worsens nowadays, as extreme-scale linear systems must be solved on massively parallel supercomputers, making single-level preconditioners require excessive iterations. Multilevel preconditioners like Geometric Multigrid (GMG) or Algebraic Multigrid (AMG) [4] help overcome the problem of scalability. Indeed, thanks to the effective combination of relaxation and coarse-grid correction, they often solve a given PDE with a number of iterations independent of the mesh size. Many freely available multigrid packages exist, such as Hypre [5], Trilinos [6], or PETSc [7], providing excellent implementations.

It is important to recognise the current hardware limitations in order to devise algorithms that overcome them. For instance, the low arithmetic intensity of most sparse linear algebra kernels and the limited memory available in massively parallel accelerators. This work extends past strategies to leverage spatial symmetries [8, 9, 10], often present in academic and industrial configurations, to make AMG lighter and more compute-intensive. The resulting preconditioner relies on imposing a consistent ordering that leads to a multigrid reduction framework. In particular, we introduce an aggressive coarsening to the top level of the multigrid hierarchy, reducing the setup, memory footprint and application costs of the top-level smoother.

The remaining sections of this work are organised as follows. Section 2 derives the proposed AMG reduction framework. Section 3 discusses its practical implementation. Section 4 analyses the benefits of our proposal by comparing it with a standard AMG in a model problem, and section 5 gives some concluding remarks.

2 AMG REDUCTION FRAMEWORK

AMGR emerges from applying a non-overlapping domain decomposition, which must align with the specific problem to fully leverage its benefits. Figure 1 illustrates a domain partitioned into n_b subdomains, along with the corresponding classification of unknowns. These unknowns are divided into inner ($\Omega_1 \cup \ldots \cup \Omega_{n_b}$) and interface ($\Gamma_1 \cup \ldots \cup \Gamma_{n_b}$) categories.

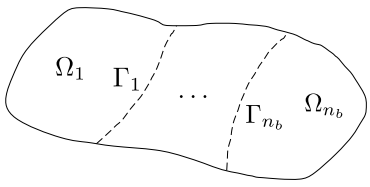

Figure 1: Simplified example of a non-overlapping domain decomposition.

Irrespective of the chosen decomposition, the quantity of inner unknowns should significantly exceed the number of interface unknowns, *i.e.*, $n_{\text{inn}} \gg n_{\text{ifc}}$. Furthermore, by rearranging the coefficient matrix so that inner unknowns are indexed before interface unknowns, it adheres to the following block structure:

$$A = \begin{pmatrix} \bar{K} & \bar{B} \\ \bar{B}^T & \bar{C} \end{pmatrix} \in \mathbb{R}^{n \times n}, \tag{1}$$

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

Although we have not yet discussed how to define appropriate decompositions, the structure of eq. (1) suggests constructing an AMG reduction by designating only the interface unknowns as coarse. However, this approach leads to excessively large fine-coarse distances. To achieve accurate prolongation, some inner nodes must be converted into coarse by ensuring a maximum interpolation distance k.

For simplicity, from this point forward, the terms inner and interface will refer not to the original decomposition (regardless of its specifics) but to their expanded versions. Similarly, n_{ifc} will represent the enlarged interface, while n_{inn} will refer to the remaining inner unknowns. With this, we can introduce the following prolongation:

$$P \coloneqq \begin{pmatrix} \bar{W} \\ \mathbb{I}_{n_{\text{ifc}}} \end{pmatrix} \in \mathbb{R}^{n \times n_{\text{ifc}}}, \tag{2}$$

where $\bar{W} \in \mathbb{R}^{n_{\text{inn}} \times n_{\text{ifc}}}$ and $\mathbb{I}_{n_{\text{ifc}}} \in \mathbb{R}^{n_{\text{ifc}} \times n_{\text{ifc}}}$. The coarsening introduced by the AMG reduction becomes more aggressive as k increases. However, this improvement comes at the cost of lowering the accuracy of P. Numerical experiments have shown that selecting $k = 2$ offers an optimal trade-off.

The other components of the AMG reduction framework follow naturally. Firstly, the top-level smoother is defined as:

$$M^{-1} \coloneqq \begin{pmatrix} \bar{M}_K^{-1} & \\ & \bar{M}_C^{-1} \end{pmatrix} \in \mathbb{R}^{n \times n}, \tag{3}$$

where $\bar{M}_K^{-1} \simeq \bar{K}^{-1}$ and $\bar{M}_C^{-1} \simeq \bar{C}^{-1}$. In eq. (3), we can ignore A's off-diagonal blocks without harming the overall quality of AMGR.

On the other hand, the reduced operator, $A_c \in \mathbb{R}^{n_{\text{ifc}} \times n_{\text{ifc}}}$, reads:

$$A_c \coloneqq P^T \begin{pmatrix} \bar{K} & \bar{B} \\ \bar{B}^T & \bar{C} \end{pmatrix} P = \bar{W}^T \bar{K} \bar{W} + \bar{W}^T \bar{B} + \bar{B}^T \bar{W} + \bar{C}, \tag{4}$$

whose inverse is approximated through a standard AMG, $M_c^{-1} \simeq A_c^{-1}$, therefore defining the following error propagation:

$$E_{\text{AMGR}} = (\mathbb{I}_n - M^{-T})^{\nu_2} (\mathbb{I}_n - P M_c^{-1} P^T)(\mathbb{I}_n - M^{-1})^{\nu_1}, \tag{5}$$

where ν_1 and ν_2 correspond to the number of pre-smoothing and post-smoothing steps, respectively.

It is important to emphasize that, irrespective of the domain decomposition, AMGR introduces an aggressive coarsening strategy that reduces the cost of the two-grid correction.

Furthermore, implementing a suitable inner-interface partitioning provides additional numerical and computational benefits.

In particular, CFD simulations on domains with reflection symmetries or composed of repeated "substructures" (such as the plates of a finned-tube heat exchanger) allow assigning a different subdomain to each substructure, results in a coefficient matrix that satisfies eq. (1). Moreover, applying the same ordering within each subdomain ensures the following property:

$$\bar{K} = \mathbb{I}_{n_b} \otimes K \text{ and } \bar{M}_K^{-1} = \mathbb{I}_{n_b} \otimes M_K^{-1}, \tag{6}$$

where $K \in \mathbb{R}^{n_{\text{inn}}/n_b \times n_{\text{inn}}/n_b}$ stands for the restriction of $\bar{K}$ to the substructure, and $M_K^{-1} \simeq K^{-1}$.

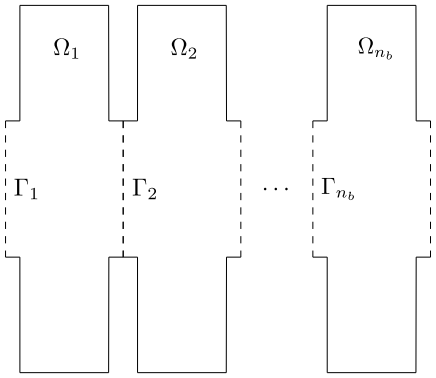

Figure 2: Adequate domain decomposition of periodic structures. Dashed lines identify the interface.

Equation (6) is particularly advantageous because it enables the assembly of K and M_K^{-1} rather than the entire $\bar{K}$ and $\bar{M}_K^{-1}$. As a result, AMGR significantly reduces the setup costs and memory footprint of $\bar{M}_K^{-1}$ by a factor of n_b, which is especially beneficial given the resource demands of the top-level smoother, M^{-1}, in a multigrid hierarchy. Additionally, the decomposition in eq. (6) enables replacing the standard `SpMV` by $\bar{K}$:

$$\texttt{SpMV:} \quad \begin{pmatrix} y_1 \\ \vdots \\ y_{n_b} \end{pmatrix} = \begin{pmatrix} K & & \\ & \ddots & \\ & & K \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{n_b} \end{pmatrix} \in \mathbb{R}^{n_{\text{inn}}} \tag{7}$$

with the more compute-intensive sparse matrix-matrix product (`SpMM`):

$$\texttt{SpMM:} \quad (y_1 \dots y_{n_b}) = K(x_1 \dots x_{n_b}) \in \mathbb{R}^{n_{\text{inn}}/n_b \times n_b}. \tag{8}$$

The fact that `SpMM` reads K n_b fewer times makes it significantly more compute-intensive, and since `SpMV` and `SpMM` are typically memory-bound kernels, such an increase of the arithmetic intensity results in significant speed-ups. Importantly, `SpMM` can also be utilized on $\bar{M}_K^{-1}$ when employing smoothers that operate via `SpMV`, such as Factored Sparse Approximate Inverse (FSAI).

3 PARALLEL IMPLEMENTATION

Discretising complex geometries is simplified by exploiting spatial symmetries, as AMGR only necessitates meshing the *base mesh*, which, with n_b subdomains, corresponds to a $1/n_b$ portion of

the entire domain. The implementation then expands the *base mesh* by applying a symmetry-aware ordering and taking advantage of the resulting operators' structure. Consequently, it is not needed to create perfectly symmetric meshes, and significant savings in memory and computational resources are granted.

Similarly, to replace `SpMV` with `SpMM`, a consistent domain partitioning is required. This involves distributing the *base mesh* across the available computing resources and extending such partitioning to the remaining subdomains applying each of the symmetries. Figure 3 illustrates this process on a 2D problem.

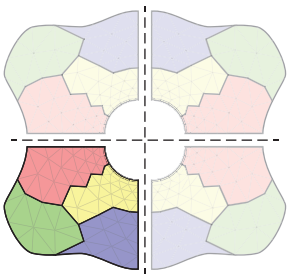

Figure 3: Adequate partitioning of a mesh with 2 reflection symmetries.

The proposed methods have been implemented within the MATLAB interface of Chronos [11], a sparse linear algebra library tailored for parallel computing environments. Chronos offers iterative solvers for linear systems and eigenvalue problems, along with advanced preconditioners based on approximate inverses and AMG. The library is developed in C++ with a strong object-oriented architecture, facilitating both its development and maintenance. It employs a hybrid programming model, utilizing MPI for inter-node communication, and OpenMP and CUDA to harness the capabilities of manycore processors and GPU accelerators, respectively.

4 NUMERICAL EXPERIMENTS

To demonstrate the behaviour of AMGR, let us consider the following Poisson's equation with homogeneous Neumann boundary conditions, representative of the problems encountered in incompressible CFD simulations:

$$\begin{aligned} -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 u}{\partial z^2} &= f \text{ in } \Omega \\ \frac{\partial u}{\partial n} &= 0 \text{ on } \partial\Omega \end{aligned} \tag{9}$$

where f is a random field that satisfies the compatibility condition $\iiint_\Omega f \, dV = 0$. The domain used for the model problem is a unit cube, discretized with a standard 7-point stencil and incorporating the following hyperbolic stretching at the walls:

$$x_i = \frac{1}{2}\left(1 + \frac{\tanh\left(\gamma_x\left(2\frac{(i-1)}{n_x} - 1\right)\right)}{\tanh\left(\gamma_x\right)}\right) \quad \forall i \in \{1, \ldots, n_x + 1\}, \tag{10}$$

applied identically in the y- and z-directions with $\gamma_x = \gamma_y = \gamma_z = 1.5$.

Table 1 presents the results obtained with AMGR on eq. (9). They correspond to a MATLAB implementation using $k = 2$, Dynamic Pattern Least Squares (DPLS) [12] and energy minimisation [13]. Conversely, the AMG applied to the reduced operator used a PMIS coarsening [14], Extended+I interpolation [15] and an adaptive-pattern FSAI smoother [12]. As it can be seen, adopting the aggressive coarsening that spatial symmetries induce does not harm convergence and allows for the computational advantages of replacing `SpMV` with `SpMM`.

Table 1: Results on the model problem with $n = 64^3$ and comparing Conjugate Gradients [1] preconditioned with AMG and AMGR.

	iterations			
preconditioner	$n_b = 1$	$n_b = 2$	$n_b = 4$	$n_b = 8$
AMG	6	na	na	na
AMGR	6	9	9	9

5 CONCLUSIONS

This work introduced an AMG reduction framework tailored to exploit spatial symmetries frequently encountered in academic and industrial CFD simulations. By incorporating an aggressive coarsening strategy at the top level of the multigrid hierarchy, AMGR, the proposed method, reduces the setup costs, memory footprint, and application costs associated with the top-level smoother. The numerical experiments conducted on a model problem demonstrate that this approach does not harm AMG's excellent convergence despite enabling significant computational advantages.

Indeed, by implementing a consistent domain decomposition and a symmetry-aware ordering, the standard `SpMV` can be replaced with the more compute-intensive `SpMM`, enabling significant memory and computational savings. This advantage is crucial given the memory-bound nature of sparse linear algebra kernels, especially on modern parallel computing architectures. Future work focus on tackling its parallel implementation and testing AMGR on industrial applications.

ACKNOWLEDGEMENTS

This project was partially funded by the competitive R+D project RETOtwin (PDC2021-120970-I00), given by MCIN/AEI/10.13039/501100011033 and European Union Next GenerationEU. Numerical experiments were conducted on the JFF cluster at the Heat and Mass Transfer Technological Center. The authors thankfully acknowledge these institutions.

REFERENCES

[1] Y. Saad, *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003.

[2] Y. Saad and H. A. V. d. Vorst, "Iterative solution of linear systems in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, pp. 1 – 33, 10 2000.

[3] M. Benzi, "Preconditioning Techniques for Large Linear Systems: A Survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418 – 477, 2002.

[4] A. S. Ulrich Trottenberg, Cornelius W. Oosterlee, *Multigrid*. Elsevier, 2000.

[5] "*hypre*: High performance preconditioners." `https://llnl.gov/casc/hypre`, `https://github.com/hypre-space/hypre`.

[6] L. Berger-Vergiat, C. A. Glusa, J. J. Hu, M. Mayr, A. Prokopenko, C. M. Siefert, R. S. Tuminaro, and T. A. Wiesner, "MueLu multigrid framework." `http://trilinos.org/packages/muelu`, 2019.

[7] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang, "PETSc Web page." `http://www.mcs.anl.gov/petsc`, 2015.

[8] A. Alsalti-Baldellou, X. Álvarez Farré, F. X. Trias, and A. Oliva, "Exploiting spatial symmetries for solving poisson's equation," *Journal of Computational Physics*, vol. 486, p. 112133, 8 2023.

[9] A. Alsalti-Baldellou, X. Álvarez Farré, G. Colomer, A. Gorobets, C. D. Pérez-Segarra, A. Oliva, and F. X. Trias, "Lighter and faster simulations on domains with symmetries," *Computers & Fluids*, vol. 275, p. 106247, 5 2024.

[10] A. Alsalti-Baldellou, C. Janna, X. Álvarez Farré, and F. X. Trias, "Exploiting symmetries for preconditioning poisson's equation in CFD simulations," pp. 1–9, ACM, 6 2023.

[11] G. Isotton, M. Frigo, N. Spiezia, and C. Janna, "Chronos: A General Purpose Classical AMG Solver for High Performance Computing," *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. C335–C357, 2021.

[12] V. A. Paludetto Magri, A. Franceschini, and C. Janna, "A Novel Algebraic Multigrid Approach Based on Adaptive Smoothing and Prolongation for Ill-Conditioned Systems," *SIAM Journal on Scientific Computing*, vol. 41, pp. A190–A219, Jan. 2019.

[13] C. Janna, A. Franceschini, J. B. Schroder, and L. Olson, "Parallel Energy-Minimization Prolongation for Algebraic Multigrid," *SIAM Journal on Scientific Computing*, vol. 45, no. 5, pp. A2561–A2584, 2023.

[14] H. De Sterck, U. M. Yang, and J. J. Heys, "Reducing complexity in parallel algebraic multigrid preconditioners," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 4, pp. 1019–1039, 2006.

[15] H. De Sterck, R. D. Falgout, J. W. Nolting, and U. M. Yang, "Distance-two interpolation for parallel algebraic multigrid," *Numerical Linear Algebra with Applications*, vol. 15, no. 2-3, pp. 115–139, 2008.