D1.7
Improvements to deal.II

Project Title	dealii-X: an Exascale Framework for Digital Twins of the Human Body
Project Number	101172493
Funding Program	European High-Performance Computing Joint Undertaking
Project start date	1 October 2024
Duration	27 months

dealii-X has received funding from the European High-Performance Computing Joint Undertaking Programme under grant agreement Nº 101172493

Deliverable title	Improvements to deal.II
Deliverable number	D1.7
Deliverable version	[Version number]
Date of delivery	[Planned date]
Actual date of delivery	[Actual date]
Nature of deliverable	Report
Dissemination level	Public
Work Package	WP1
Partner responsible	UNITOV

Abstract	
Keywords	

Document Control Information

Version	Date	Author	Changes Made
0.1	21/02/2026	Marco Feder	Initial draft
0.2	22/02/2026	Marco Feder	Polygonal discretization module section
0.3	23/02/2026	Marco Feder	Integration and preliminary results for PSCToolkit
0.4	23/02/2026	Chiara Puglisi	Preliminary results with advanced features in MUMPS
0.5	25/02/2026	Luca Heltai	Integration of VTK utilities and new tutorial

Approval Details

Approved by: [Name]
Approval Date: [Date]

Distribution List

- Project Coordinators (PCs)
- Work Package Leaders (WPLs)
- Steering Committee (SC)
- European Commission (EC)

Disclaimer: This project has received funding from the European Union. The views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European High-Performance Computing Joint Undertaking (the “granting authority”). Neither the European Union nor the granting authority can be held responsible for them.

Contents

1 Introduction

1.1 Purpose of the Document

1.2 Structure of the Document

- Section 2: Pre-exascale capabilities of deal.II
- Section **??**: Pre-exascale modules of deal.II
- Section 3: Polygonal discretization module
- Section 4: Integration of PSCToolkit
- Section 5: Integration of MUMPS

2 Pre-exascale capabilities of deal.II

This section documents upstream (deal.II) development items tracked with the `dealii-x` label and summarises recent auxiliary developments relevant to reduced (and mixed-dimensional) modelling workflows. This report discusses (i) Polygonal discretisation workstream in Section 3, (ii) specific PSCToolkit/AMG4PSBLAS integration in Section 4, and (iii) advanced MUMPS factorisation features in Section 5. To avoid repetition, we focus here on *complementary* contributions: a coherent series of VTK interoperability utilities in `deal.II` (from VTK data ingestion to direct conversion between `vtkUnstructuredGrid` and `Triangulation`), and supporting improvements in related infrastructure that enable robust data exchange for dealii-X applications.

A key outcome is an expanded, reusable path for importing and reusing meshes and solution fields from external toolchains (VTK-based) inside deal.II-based dealii-X applications, reducing friction for multi-physics coupling, parameter studies, and reduced modelling workflows where repeated transfer between meshing/visualisation pipelines and solver back-ends is common. The work is coordinated via a dedicated meta-issue (#19056) and realised through a sequence of PRs (beginning with PR:#19023 and PR:#19024). In addition, we briefly note the status of two research-code repositories that support reduced/mixed-dimensional modelling pipelines for dealii-X: the “Reduced Lagrange Multipliers” repository, currently in use by the FAU group for brain brain simulations related to MRE studies and by the WIAS group for Liver simulations, and the “Reduced dimensional blood flow simulator”, currently under active development, and to be coupled with Liver and Brain simulations.

2.1 VTK interoperability and data-ingestion utilities

2.1.1 Motivation and relation to dealii-X

VTK is widely used as an interchange format in scientific computing pipelines (mesh generation, post-processing, experimental datasets, and in-house tools). For dealii-X, the ability to *import* VTK meshes and associated point/cell fields using native VTK APIs, and to *convert* VTK unstructured grids into deal.II triangulations (and back) enables: (i) streamlined integration with external pre-processing/segmentation tools, (ii) reproducible reuse of meshes/fields for reduced modelling surrogates, and (iii) less bespoke glue code in each lighthouse application, including robustness in

future development of dealii-X applications that is inherited from the use of direct VTK apis. The VTK utilities workstream is coordinated by the meta-issue #19056 (opened as of Newsletter #339).

2.2 New tutorials under the `dealii-x` label

PR:#18888: "Step-80 dealii-x" The tutorial implements a finite element immersed boundary method with distributed Lagrange multipliers for fully coupled fluid-structure interaction: an incompressible Navier–Stokes fluid is coupled to a compressible (possibly viscous) hyper-elastic solid on independent, non-matching volumetric meshes, with velocity continuity enforced through an additional multiplier field λ rather than penalization. It combines particle-based transfer, `MappingFEField`-driven solid motion, and a monolithic saddle-point solve for (u, p, w, λ), that will be further developed to include relevant validation scenarios. Within dealii-X, this is directly relevant to coupled multiphysics workloads in cardio-vascular flows, brain mechanics, and lungs/liver mechanics, and it provides a realistic test bench for linear solver and factorization capabilities developed in the project, in particular PSCToolkit-based preconditioning and advanced MUMPS configurations.

The PR implements and showcases the application of the recently developed augmented Lagrangian preconditioner in the UNIPI group, which is designed to handle the saddle-point structure of the coupled system Benzi et al. (2026).

3 Polygonal discretization module

The polygonal discretization module described in Work Package 1.5 has undergone exhaustive testing and validation. The new library associated with this module, `Polydeal`, is available at `https://github.com/fdrmrc/Polydeal`. Some features available in this module are designed to be integrated into the deal.II library. In order to guarantee maximum compatibility, it is updated to the latest version of deal.II, and it is developed following the same coding style and practices. A comprehensive test suite is deployed at each new commit on the continuous integration system to guarantee the integrity of the codebase.

In view of the integration of the core functionalities of `Polydeal` into `deal.II`, a preliminary pull request (`https://github.com/dealii/code-gallery/pull/233`) has been opened to submit an example demonstrating an agglomeration-based solver for the Poisson equation to the code gallery. Building from this, a refactoring process is ongoing to make the codebase of `Polydeal` compatible with `deal.II` coding conventions and ready for merging, expected by the end of the project (deliverable D1.8).

3.1 Geometrically informed multilevel preconditioning

Large-scale simulations require efficient preconditioners for the iterative solution of the linear systems arising from finite element discretizations, for instance with discontinuous Galerkin (DG) methods. For elliptic problems, geometric multigrid methods are among the most effective preconditioners. However, their application requires the construction of a mesh hierarchy, which can be challenging for fine, unstructured geometries. In such cases, AMG methods are often employed as an alternative. One of the key features of polytopal methods is precisely their very good interplay with DG methodologies (see e.g., Cangiani et al. (2014); Antonietti et al. (2013)). In particular, the flexibility of DG methods allows the usage of very general agglomerated grids, i.e. grids obtained by merging together several elements of a finer grid.

By exploiting the efficient agglomeration routine developed in Feder et al. (2025), already available in `Polydeal`, it is possible to construct a hierarchy of *nested* agglomerated grids, for which intergrid transfer operators among consecutive levels are

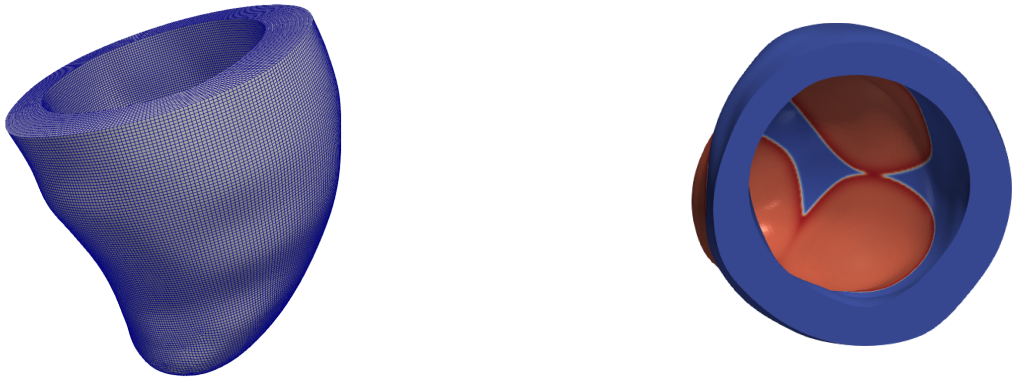
cheap. This hierarchy naturally enables the construction of multilevel preconditioners for DG discretizations of elliptic problems, leveraging polytopal grids on coarser levels, while keeping the *original* grid unchanged. Coarser operators can be obtained by rediscretization of the partial differential equation on the agglomerated grids, or by triple Galerkin projection[1]. The latter approach is particularly appealing, since it allows obtaining coarser operators without the need of rediscretization on agglomerated meshes.

This technique has been successfully applied in Feder and Africa (2026) to the DG discretization of the monodomain model arising in cardiac electrophysiology. More precisely, we have developed a novel multigrid solver for its DG discretization, exploiting agglomerated grids on coarser levels. The resulting preconditioner builds coarser operators in an algebraic multigrid fashion, injecting geometric information through the agglomeration routine. In this sense, we have devised a *geometrically informed* multigrid preconditioner, with the geometric information being injected through the agglomeration routine.

Finally, the linear system of equations associated with the model is solved, at each time-step, with a conjugate-gradient method preconditioned by one V-cycle of our multigrid scheme. The preconditioner has been successfully applied to several test cases, including high-order polynomial degrees, and a realistic 3D ventricular geometry, shown in Figure 1. Since the finest level of the multigrid hierarchy consists of an hexahedral grid, we leverage the tensor-product structure of the basis functions and quadrature points in order to exploit state-of-the-art matrix-free operator evaluation techniques developed in Kronbichler and Kormann (2019), which form a key part of the computational backbone of the deal.II library.

In Figure 2, we report the number of iterations throughout all the simulation, comparing our agglomerated multigrid preconditioner with the AMG implementation available in the Trilinos ML package. The results are reported for polynomial degrees $p = 1, 2$ for the three-dimensional ventricle mesh. It is evident how the iteration counts of the AMG preconditioner are always higher than the agglomeration-based multigrid approach. The actual wall-clock times (in seconds) are reported in Figure 3, showing a huge reduction of the time per iteration when using the agglomeration-based multigrid strategy. A detailed discussion on the costs can be found in Feder and Africa (2026).

[1] This means that coarser operators are recursively obtained by restriction of the finer operators, in an algebraic multigrid sense.



(a) (b)

Figure 1: (a) Hexahedral mesh representing a realistic left ventricle. (b) Propagation of the transmembrane potential.

The obtained results indicate the high effectiveness of the preconditioner in terms of iteration counts and robustness with respect to model parameters. Despite these encouraging results, several research directions remain open to further enhance the efficiency and scalability of the preconditioner. In this sense, integration within existing AMG frameworks is a promising direction, along with the development of matrix-free implementations of the coarser operators, which would further reduce the application cost of the preconditioner.

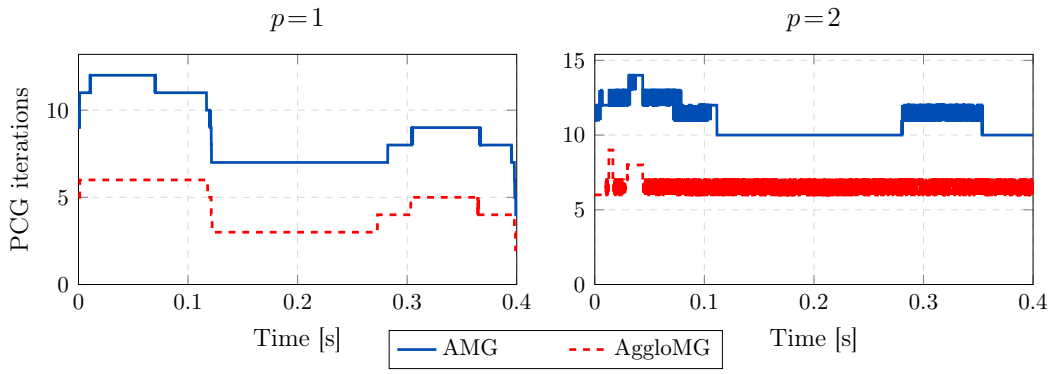


Figure 2: Number of preconditioned conjugate-gradient (PCG) iterations per time step for the monodomain problem, comparing AMG and agglomerated multigrid (AggloMG) for polynomial degrees $p=1,2$ for the three-dimensional ventricle test.

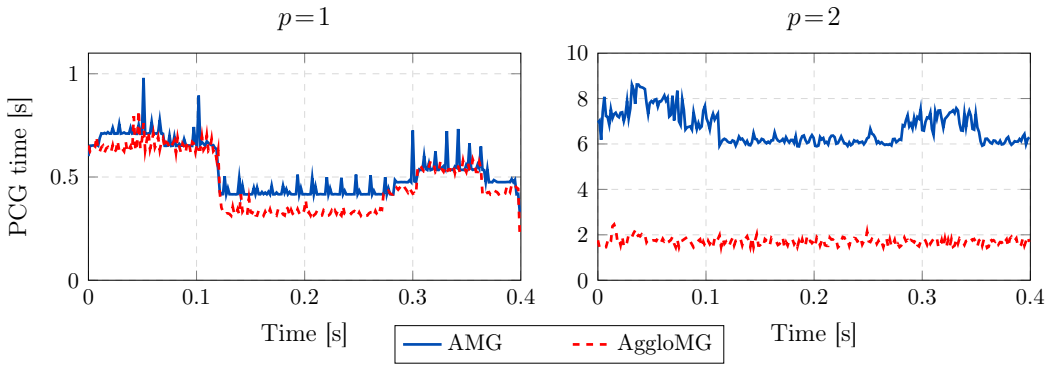


Figure 3: Wall-clock time for PCG iteration time per time step for the monodomain problem, comparing AMG and agglomerated multigrid (AggloMG) for polynomial degrees $p = 1, 2$ for the three-dimensional ventricle test. Results have been obtained on 128 MPI processes on the Toeplitz cluster at University of Pisa.

4 Integration of PSCToolkit

The integration of PSCToolkit (Parallel Sparse Computing Toolkit) D'Ambra et al. (2025) aims to provide scalable algebraic multigrid (AMG) preconditioners through its AMG4PSBLAS package. By merging the preconditioners developed within AMG4PSBLAS into deal.II, it will be possible to exploit them as drop-in replacements in application codes. The integration targets the following new release candidates: PSBLAS v3.9 and AMG4PSBLAS v1.2.

4.1 Integration status

The integration of PSCToolkit into the deal.II master branch is currently in progress, and can be tracked on the official deal.II GitHub repository [2]. Concurrently, a fully working interface (whose code will be progressively upstreamed into the main deal.II repository) has been developed on a separate branch[3], providing complete access to the preconditioners developed within AMG4PSBLAS.

In general, the design of the user interface closely follows the existing structure of the main deal.II linear algebra frameworks, such as the interfaces to Trilinos and PETSc. In the same fashion, the interface to AMG preconditioners by AMG4PSBLAS has been designed following the same design principles of the ex-

[2] See PR:#19050 and PR:#18938

[3] Available at `https://github.com/psctoolkit/dealii/commits/configure_amg4psblas/`.

isting interfaces to Trilinos and PETSc wrappers. Indeed, within the namespace `PSCToolkitWrappers`, a new class `PreconditionAMG` has been introduced, which implements a minimal interface which allows to:

- construct the preconditioner with a given set of user-specified parameters,
- build the AMG hierarchy and related smoothers, out of a given distributed PSBLAS system matrix,
- apply the resulting preconditioner to a given vector through the standardized `vmult(dst,src)` method.

Parameters are exposed by passing an `AdditionalData` object to the constructor of the preconditioner, which is a standardized data structure to pipe additional flags to the preconditioner. In this way, the integration is completely transparent from the user's perspective: switching from, e.g., a Trilinos ML or Hypre AMG preconditioner to an AMG4PSBLAS one requires no structural changes to the application code.

4.2 Preliminary results for AMG4PSBLAS preconditioners

To validate the correctness and assess the performance of the AMG4PSBLAS interface, we have developed a tutorial application implementing a finite element solver for the Poisson problem, employing nodal Lagrangian $\mathcal{Q}^1$ elements. The implementation and results are documented in deliverable D1.6.

To evaluate performance at significantly larger problem sizes, we have been granted access to the MareNostrum 5 supercomputer at the Barcelona Supercomputing Center[4]. These resources will enable us to conduct experiments and benchmarking at scale, providing insight into the behavior of our interface under sizes typical of pre-exascale applications.

4.3 Ongoing and future work

Ongoing work includes several tasks aimed at finalizing the integration and assessing the performance of the AMG4PSBLAS preconditioners in large-scale simula-

[4]MareNostrum 5 information: `https://www.bsc.es/marenostrum/marenostrum-5`

tions:

- finalizing the interface, introducing wrappers for sparse matrix types,
- large-scale performance evaluations on HPC systems to identify potential performance bottlenecks in the interface and in the preconditioner itself,
- GPU support: PSBLAS provides GPU acceleration through CUDA backends. The exposition of this functionality through the deal.II interface will enable GPU-accelerated AMG preconditioning within standard deal.II solver workflows.

5 Integration of MUMPS

The direct solution of a sparse linear system $Ax = b$ relies on the factorization of the matrix A. The robustness of direct methods comes at the cost of a high complexity both in terms of number of floating point operations and of memory footprint.

The complexity of the direct methods approaches can be reduced thanks to Block Low-Rank (BLR) approximation (see Amestoy et al. (2019, 2015)) with adaptive precision Amestoy (Boiteau et al.) to exploit data sparsity. This is illustrated in Table 1 on the Beltrami Flow problem from lifex library. Comparing the first two rows "Full-Rank" factorization and "BLR factorization with adaptive precision" in Table 1 we see that the number of operation is reduce by two orders of magnitude.

	Memory used (Gbytes)	Number of op. to factor matrix	Total time (sec)	Backward error after IR
Full-Rank	620	1.6E15	**364**	2E-15
BLR(1E-5)+adaptive	224	2.3E13	38	1E-10
Single precision factorization in double prec instance with mixed precision IR				
Full-Rank	315	1.6E15	229	2E-12
BLR(1E-5)+adaptive	125	2.0E13	**24**	8E-11

Table 1: Beltrami matrix, performance analysis, AMD Genoa (4MPI x 48th). IR=Iterative Refinement

The performance can be further improved using mixed precision algorithms Amestoy (Buttari et al.) as illustrated with the last two lines of Table 1. In row "Full-Rank", *single precision factorization with mixed precision iterative refinement (IR)* and an efficient approach to access data in low precision Amestoy et al. (2025) enable to divide time and memory by a factor of two. Combining this approach to BLR with adaptive precision leads to an overall significant reduction in time (from 364 to 24 seconds) and memory (from 620 to 125 Gbytes) with a satisfactory backward error. It should be mentioned, as described in Amestoy (Buttari et al.), that on ill-conditioned matrices GMRES based iterative refinements should be used.

Recent experiments with accelerated nodes (Nvidia-GraceHopper and AMD-Mi250 and AMD-Mi300) have shown that we can reduce the time for full-rank factorization of the Beltrami matrix by a factor between 3 and 6 depending on the compute node. Next challenge will be to combine the benefits from adaptive BLR and mixed precision algorithms on accelerated nodes.

6 Conclusion

References

Salvatore Filippone and Michele Colajanni, ”PSBLAS: a library for parallel linear algebra computation on sparse matrices”, ACM Transactions on Mathematical Software, vol. 26, no. 4, pp. 527–550, 2000. https://doi.org/10.1145/365723.365732

Pasqua D’Ambra, Fabio Durastante and Salvatore Filippone, ”PSCToolkit: solving sparse linear systems with a large number of GPUs”, arXiv preprint arXiv:2406.19754, 2025. https://arxiv.org/abs/2406.19754

Andrea Cangiani, Emmanuil Georgoulis and Paul Houston, ”hp-Version discontinuous Galerkin methods on polygonal and polyhedral meshes,” Mathematical Models and Methods in Applied Sciences, vol. 24, no. 4, pp. 2009-2041, 2014.

Paola Antonietti, Stefano Giani, and Paul Houston, ”*hp*-Version Composite Discontinuous Galerkin Methods for Elliptic Problems on Complicated Domains”, SIAM Journal on Scientific Computing, vol. 35, A1417-A1439, 2013.

Marco Feder, Andrea Cangiani and Luca Heltai, ”R3MG: R-tree based agglomeration of polytopal grids with applications to multilevel methods”, Journal of Computational Physics, vol. 526, 113773, 2025.

Marco Feder and Pasquale Claudio Africa, ”An agglomeration-based multigrid solver for the discontinuous Galerkin discretization of cardiac electrophysiology”, arXiv preprint arXiv:2602.16312, 2026.

Michele Benzi, Marco Feder, Luca Heltai and Federica Mugnaioni, ”Scalable Augmented Lagrangian preconditioners for Fictitious Domain problems”, Computer methods in applied mechanics and engineering, vol. 450, p. 118522, 2026.

Martin Kronbichler and Katharina Kormann, ”Fast Matrix-Free Evaluation of Discontinuous Galerkin Finite Element Operators”, ACM Transactions on Mathematical Software, vol. 45, no. 3, Article 29, 2019.

P. Amestoy, O. Boiteau, A. Buttari, M. Gerest, F. Jezequel, J.-Y. L’Excellent, and T. Mary. Mixed precision low-rank approximations and their application to block

low-rank LU factorization. *IMA Journal of Numerical Analysis*, 43:2198–2227, July 2023.

P. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. Mary, and B. Vieublé. Combining sparse approximate factorizations with mixed precision iterative refinement. *ACM Transactions on Mathematical Software*, 49(1):1–29, 2023.

P. Amestoy, A. Jego, J.-Y. L'Excellent, T. Mary, and G. Pichon. BLAS-based Block Memory Accessor with Applications to Mixed Precision Sparse Direct Solvers. preprint submitted to publication, Apr. 2025.

P. R. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker. Improving multifrontal methods by means of block low-rank representations. *SIAM Journal on Scientific Computing*, 37(3):A1451–A1474, 2015.

P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures. *ACM Transactions on Mathematical Software*, 45:2:1–2:26, 2019.