D1.4
Enhancements in Pre-Exascale Modules of 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	Enhancements in Pre-Exascale Modules
Deliverable number	D1.4
Deliverable version	1.0
Date of delivery	August 31, 2025
Actual date of delivery	August 30, 2025
Nature of deliverable	Report
Dissemination level	Public
Work Package	WP1
Partner responsible	UNIPI

Abstract	This report documents the progress achieved during the second semester of the dealii-X project (Months 7-11) in enhancing the deal.II finite element library towards exascale readiness. Building upon the foundations laid in D1.2, this deliverable focuses on the consolidation of the new features, their integration into real-world biomedical applications, and preliminary scaling tests on EuroHPC pre-exascale systems. Major achievements include significant improvements in GPU matrix-free solvers, the stabilization and extension of the generalized interface for coupling operators, the maturation of the polygonal discretization module, and the integration of external solver technologies (PSCToolkit and MUMPS) into the deal.II workflow.
Keywords	Mesh handling; Polygonal discretization; Non-matching methods and preconditioning

Document Control Information

Version	Date	Author	Changes Made
0.1	01/08/2025	Luca Heltai	Initial draft
0.2	27/08/2025	Luca Heltai	Collection of contributions
0.3	28/08/2025	Marco Feder	Add preconditioner section
0.4	29/08/2025	Marco Feder	Improve preconditioner section
0.5	29/08/2025	Martin Kronbichler	Minor cleanups in various places
1.0	30/08/2025	Luca Heltai	Final version

Approval Details

Approved by: Martin Kronbichler
Approval Date: 30 August 2025

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

The goal of Work Package 1 (WP1) is to strengthen the numerical and software backbone of the dealii-X Centre of Excellence by extending the capabilities of the deal.II library, enabling the exascale transition of digital twins for human organs. Deliverable D1.4 provides a consolidated report on the technical progress, following the preliminary results described in D1.2, and emphasizes integration, benchmarking, and readiness for multiphysics biomedical applications.

2 Purpose of the Document

The purpose of this document is to provide a comprehensive overview of the advancements made in the deal.II library as part of the dealii-X project. It aims to inform stakeholders about the current state of the library, the challenges encountered, and the strategies employed to overcome them.

3 Improvements for Exascale Readiness (WP1.1)

The 9.7 release of deal.II has introduced several improvements directly aligned with WP1.1 on GPU-enabled matrix-free solvers and exascale readiness. Key advances concern the matrix-free infrastructure, which was extended with:

- Support for multi-component and vector-valued finite elements in the matrix-free evaluation routines for the Kokkos-based portable branch. This substantially broadens the class of multiphysics applications (e.g. poromechanics in brain and liver) that can be ported to GPUs efficiently.
- Portability layer `Portable::MatrixFree` built on top of `kokkos`, ensuring code can target NVIDIA, AMD, and Intel GPUs with the same interface. The release consolidates temporary storage structures into new `DeviceVector` and `DeviceBlockVector` classes, simplifying GPU memory management.
- Performance optimizations to sum factorization kernels, yielding measurable speedups especially on NVIDIA A100/H100 hardware.

- Extended ghost cell handling and lexicographic patch smoothers for geometric multigrid preconditioners, crucial to scaling implicit solvers to 10^{11}-10^{12} DoFs on EuroHPC machines.

In addition, new solver strategies were enabled by the `TensorProductMatrixCreator` for diagonalization-based inverses, applicable in the context of multigrid smoothers and coarse solvers. These improvements directly support the objective of saturating node-level bandwidth on GPUs and minimizing data transfer overheads, as required for exascale digital twin workloads.

Ongoing developments for the matrix-free kernels, in particular node-level experiments for various GPU vendors, optimization of Kokkos-based and CUDA-based algorithms as well as energy efficiency improvements have also been made in the past few months. These contributions are described in more detail, along with an extensive evaluation of performance, in the report on deliverable D3.2: “Co-Design AMD Energien Efficiency Report I”.

4 Improvement of pre-exascale modules of the deal.II library (WP1.2)

In addition to the enhancements reported in Sections 3 and 7, further work has been carried out on the integration of external modules that are essential for the pre-exascale readiness of deal.II and for supporting the targeted biomedical applications.

4.1 Integration of the Gmsh module

A dedicated module for interfacing with `Gmsh` is currently under review in the official deal.II repository (`https://github.com/dealii/dealii/pull/18759`). The new contribution introduces a function to read *partitioned* mesh files generated by Gmsh directly into a `parallel::fullydistributed::Triangulation`. This development is critical for large-scale applications where meshes must be distributed across thousands of MPI ranks. Report D1.2 provides some details of the preliminary studies we carried over for this integration.

The main features of the new function are:

- Support for 1D, 2D, and 3D meshes partitioned via Gmsh's built-in partitioner (e.g. using the command line flags `-part N -part_ghosts -part_split`).
- Automatic handling of ghost cell identification, vertex coordinates, and cell connectivity across MPI processes.

This enhancement enables scalable parallel mesh handling and substantially simplifies the workflow for distributed simulations using deal.II and Gmsh. By allowing Gmsh to act as a pre-partitioner for distributed meshes, the new functionality removes the need for expensive repartitioning inside deal.II, and integrates seamlessly with adaptive and fully distributed solvers.

The pull request has reached a mature stage and is expected to be merged into deal.II in the coming release cycle, completing the Gmsh-related part of the dealii-X roadmap. This work directly addresses the bottleneck of mesh generation and partitioning for realistic biomedical geometries, such as brain and liver models, in exascale digital twin applications.

4.2 Development of block preconditioners

This section describes the developments of novel block preconditioners for the solution of problems stemming from non-matching discretizations and fictitious domain approaches, which are relevant in the solution of fluid-structure interaction problems. In this context, the use of Lagrange multipliers to enforce coupling conditions leads to the solution of large (and multiple) saddle point linear systems, for which the design of robust and efficient preconditioners is still a challenge.

We started our investigations with Poisson and Stokes interface problems, where the interface conditions are enforced using Lagrange multipliers. In particular, our preconditioning strategy is based on augmented Lagrangian ideas (Benzi and Olshanskii, 2006), where the original saddle point system is recast as an equivalent one, which provides better spectral properties. The resulting linear systems are solved using a Krylov subspace method such as (flexible) GMRES.

A thorough theoretical analysis of the preconditioner has been carried out, providing lower and upper bounds for the spectrum of the preconditioned system and

demonstrating independence with respect to the mesh size. Several numerical experiments have been conducted, in order to validate the theoretical findings and to assess the performance of the preconditioner. The application of the preconditioner on a vector does not require exact inversion of diagonal blocks, but cheaper inexact solves with loose internal tolerances are enough to guarantee low and constant iteration counts. To give a concrete example, we report in Table 1 the number of outer FGMRES iterations required to solve a three dimensional Stokes problem with a toroidal immersed surface over which we impose a constraint on the velocity field $\mathbf{u}$. We denote by V_h and Q_h a classical stable finite element spaces for the velocity and the pressure, respectively, while Λ_h is the Lagrange multiplier space, defined on the immersed surface.

The associated work has been submitted and is currently under revision, while our preprint by Benzi et al. (2025) is available on arXiv. The memory-distributed implementation of this methodology is based on `deal.II` and can be found at the following maintained GitHub repository `https://github.com/fdrmrc/fictitious_domain_AL_preconditioners`.

Iteration counts	
DoF $(\lvert V_h\rvert + \lvert Q_h\rvert + \lvert \Lambda_h\rvert)$	FGMRES iterations
8,367+421+48	13
32,829+1,561+192	14
149,319+6,767+768	14
984,387+42,961+3,072	14
7,132,467+304,897+12,288	13
53,641,515+2,265,401+49,152	13

Table 1: FGMRES iteration counts for the 3D Stokes problem with a toroidal immersed surface.

Current efforts are directed towards the extension of the proposed preconditioning technique to handle the following more complex scenarios:

- Elliptic interface problems with large jumps in the coefficients,
- Fluid-structure interaction problems.

Preliminary results indicate that the preconditioner maintains its robustness and efficiency in these more challenging settings.

References

M. Benzi, and M. Olshanskii "An Augmented Lagrangian-Based Approach to the Oseen Problem", SIAM Journal on Scientific Computing, vol. 28, no. 6, pp. 2095-2113, 2006.

M. Benzi, M. Feder, L. Heltai, and F. Mugnaioni "Scalable augmented Lagrangian preconditioners for fictitious domain problems," https://arxiv.org/abs/2504.11339, 2025.

4.3 VTK interoperability: mesh & field I/O utilities (prototype)

Preliminary VTK support has been prototyped in the `reduced_lagrange_multipliers` project (https://github.com/luca-heltai/reduced_lagrange_multipliers/pull/40), providing a lightweight header (`vtk_utils.h`) with utilities that bridge VTK files and deal.II data structures (guarded by `DEAL_II_WITH_VTK`).

Field readers (serial input).

- `read_cell_data(vtk_filename, cell_data_name, Vector<double>&)`: reads a named *cell* data array (scalar or vector) from a VTK `UnstructuredGrid` into a flat vector, in row-major order ($\text{cell}_0\_\text{comp}_0, \ldots, \text{cell}_1\_\text{comp}_0, \ldots$).
- `read_vertex_data(vtk_filename, vertex_data_name, Vector<double>&)`: same for *point/vertex* data arrays.
- `read_data(vtk_filename, Vector<double>&)`: concatenates all point data first (all fields, all components), then all cell data, in discovery order; useful as a one-shot bulk import.

Mesh readers and FE mapping.

- `read_vtk(vtk_filename, Triangulation&)`: imports a VTK mesh into a (serial) `Triangulation`; optional cleaning can merge overlapping points.
- `vtk_to_finite_element(vtk_filename)`: constructs a suitable `FiniteElement` (returned as a `FESystem`) by inspecting VTK fields: point data $\rightarrow$ `FE_Q` (or `FESystem(FE_Q, n_comps)`), cell data $\rightarrow$ `FE_DGQ` (or `FESystem(FE_DGQ, n_comps)`). Also returns the vector of field names.
- `get_block_indices(fe)`: returns a `BlockIndices` mapping for the FE (workaround for a 9.6 issue).

Data plumbing into DoFs (serial and distributed).

- `read_vtk(vtk_filename, DoFHandler&, Vector<double>&, std::vector<std::string>&)`: end-to-end routine that reads the mesh, builds the FE (as above), distributes DoFs block-wise, and fills a single data vector with all fields; also returns field names.
- `data_to_dealii_vector(serial_tria, data, dh, output_vector)`: maps bulk VTK data (`read_data`) onto a (serial or parallel) `DoFHandler` vector of an FE built via `vtk_to_finite_element`. It handles vertex- vs. cell-based fields with separate block offsets, checks ownership via `locally_owned_dofs`, and assumes serial and parallel meshes preserve cell ordering for cell fields.
- `serial_vector_to_distributed_vector(serial_dh, parallel_dh, serial_vec, distributed_vec)`: transfers per-vertex (and per-cell) data from a serial vector to a distributed one.
- `distributed_to_serial_vertex_indices(serial_tria, parallel_tria)`: builds a map from parallel vertex indices to serial indices for locally-owned vertices; used internally for consistent vertex data scatter.

Impact. These utilities establish a reproducible VTK $\leftrightarrow$ deal.II path for both mesh and multi-field data, enabling: (i) robust import of external unstructured grids, (ii) automatic FE layout matching field arity (point vs. cell, scalar vs. vector), (iii) consistent

serial-to-distributed data transfer. This reduces bespoke conversion scripts and supports large-scale workflows with ParaView/VTK pipelines—crucial for pre-exascale digital twin scenarios.

5 Polygonal Discretization Methods (WP1.3)

The Polygonal discretization module has been further developed, enriched with a suite of test cases and benchmark applications, as reported in D1.5. Work is ongoing to merge the module into the official deal.II repository, including comprehensive documentation and tutorials.

6 Integration of PSCToolkit (WP1.4)

Release 9.7 includes preliminary hooks for the `PSBLAS`/PSCToolkit linear algebra stack. This allows deal.II applications to leverage algebraic multigrid preconditioners and GPU-enabled sparse BLAS routines. The main advances are:

- Configurable support for PSCToolkit backends through CMake, alongside existing PETSc and Trilinos.
- A common interface design consistent with deal.II's existing solver wrappers, easing migration of application codes.
- Planned co-design with AMG4PSBLAS smoothers to support nonlinear and time-dependent biomedical solvers where frequent coarse-grid rebuilds are required.

This step is fundamental to enable hybrid matrix-free/matrix-based solvers where PSCToolkit provides scalable algebraic coarse solvers complementing deal.II's matrix-free fine-scale kernels.

7 Integration of MUMPS (WP1.5)

The 9.7 release has introduced a direct wrapper for MUMPS, no longer requiring integration solely through PETSc or Trilinos. The enhancements include:

- Transparent support for parallel multifrontal factorization and out-of-core capabilities.
- Interfaces for block low-rank compression and mixed-precision algorithms, in line with recent MUMPS developments for exascale readiness.
- GPU acceleration for parts of the factorization, providing robustness in scenarios with ill-conditioned coarse problems such as multiphysics brain–liver coupling.

The integration of MUMPS as a first-class backend within deal.II ensures that robust direct solvers are available when iterative approaches are insufficient. This is particularly relevant for coarse-grid solvers in multigrid hierarchies, and for nonlinear inversion tasks common in digital twin calibration.

8 Challenges and Future Plans

Despite these advances, several challenges remain:

- Ensuring uniform GPU performance across NVIDIA, AMD, and Intel hardware remains a priority; AMD/Intel kernels are not yet tuned to the same level as CUDA-based ones, in particular for the matrix-free kernels of deal.II (see also the report to deliverable D3.2 for the current state on Intel Ponte Vecchio GPUs) and the MUMPS package.
- Minimization of data movement between CPU and GPU during assembly phases is ongoing, as deal.II assembly is still CPU-only in most workflows.
- Interoperability between polygonal discretizations, PSCToolkit AMG solvers, and MUMPS direct solvers must be systematically validated in multiphysics organ applications.

Future work in WP1 will finalize the integration of polygonal discretization into mainline deal.II, extend GPU assembly kernels, and deliver comprehensive benchmarks on EuroHPC systems (LEONARDO, LUMI, JUPITER). Training materials and new tutorials will further disseminate exascale-ready modules to the broader user community.