The deal.II Library, Version 9.6

Pasquale Claudio Africa[1], Daniel Arndt[2*], Wolfgang Bangerth[3,4], Bruno Blais[5], Marc Fehling[6], Rene Gassmöller[7], Timo Heister[8], Luca Heltai[9], Sebastian Kinnewig[10], Martin Kronbichler[11,12], Matthias Maier[13], Peter Munch[12,14], Magdalena Schreter-Fleischhacker[15], Jan Philipp Thiele[16], Bruno Turcksin[2*], David Wells[17], and Vladimir Yushutin[8, 18]

[1]SISSA International School for Advanced Studies, mathLab, Via Bonomea, 265, 34136, Trieste, Italy. pafrica@sissa.it

[2]Computational Coupled Physics Group, Computational Sciences and Engineering Division, Oak Ridge National Laboratory, 1 Bethel Valley Rd., TN 37831, USA. arndtd/turcksinbr@ornl.gov

[3]Department of Mathematics, Colorado State University, Fort Collins, CO 80523-1874, USA. bangerth@colostate.edu

[4]Department of Geosciences, Colorado State University, Fort Collins, CO 80523, USA.

[5]Chemical Engineering High-performance Analysis, Optimization and Simulation (CHAOS) laboratory, Department of Chemical Engineering, Polytechnique Montréal, PO Box 6079, Stn Centre-Ville, Montréal, Québec, Canada, H3C 3A7. bruno.blais@polymtl.ca

[6]Department of Mathematical Analysis, Faculty of Mathematics and Physics, Charles University, Sokolovská 49/83, 186 75 Prague 8, Czech Republic. marc.fehling@matfyz.cuni.cz

[7]GEOMAR Helmholtz Centre for Ocean Research Kiel, 24148 Kiel, Germany

[8]School of Mathematical and Statistical Sciences, Clemson University, Clemson, SC, 29634, USA. heister@clemson.edu

[9]University of Pisa, Italy.

[10]Institute for Applied Mathematics, Scientific Computing, Leibniz University Hannover, Welfengarten 1, 30167 Hannover, Germany kinnewig@ifam.uni-hannover.de

[11]Faculty of Mathematics, Ruhr University Bochum, Universitätsstr. 150, 44780 Bochum, Germany. martin.kronbichler@rub.de

[12]Institute of Mathematics, University of Augsburg, Universitätsstr. 12a, 86159 Augsburg, Germany.

[13]Department of Mathematics, Texas A&M University, 3368 TAMU, College Station, TX 77845, USA. maier@math.tamu.edu

[14]Uppsala University, Sweden. peter.munch@it.uu.se

[15]Institute for Computational Mechanics, Technical University of Munich, Boltzmannstraße 15, 85748 Garching, Germany. magdalena.schreter@tum.de

[16]Weierstrass Institute for Applied Analysis and Stochastics, Leibniz Institute in Forschungsverbund Berlin e.V. thiele@wias-berlin.de

[17]Department of Mathematics, University of North Carolina, Chapel Hill, NC 27516, USA. drwells@email.unc.edu

[18]Department of Mathematics, University of Tennessee at Knoxville, Knoxville TN 37996-1320, USA. vyushuti@utk.edu

Abstract: This paper provides an overview of the new features of the finite element library deal.II, version 9.6.

1 Overview

deal.II version 9.6.0 was released August 11, 2024. This paper provides an overview of the new features of this release and serves as a citable reference for the deal.II software library version 9.6. deal.II is an object-oriented finite element library used around the world in the development of finite element solvers. It is available for free under the terms of the *GNU Lesser General Public License* (LGPL). The deal.II project is in the process of relicensing the library under the terms of the *Apache License 2.0 with LLVM Exception*. Downloads are available at https://www.dealii.org/ and https://github.com/dealii/dealii.

The major changes of this release are:

- Substantial performance improvements to the matrix-free and multigrid infrastructure (Section 2.1).
- Additions to the non-matching infrastructure (see Section 2.2). In particular, the new class FERemoteEvaluation provides support for evaluating finite element shape functions and solutions on parts of the domain stored by other MPI processes.
- Much work has gone into writing wrappers for Trilinos' Tpetra stack of linear algebra classes. Tpetra is Trilinos' Kokkos-based replacement for the now-deprecated Epetra stack. See Section 2.3 for more on this.
- Tool classes TaskResult and Lazy that provide ways to compute values on a separate thread or only when first accessed (Section 2.4).
- There are five new tutorial programs, on checkpointing simulations (step-83), integrating time-dependent solvers with external time stepping libraries (step-86, using PETSc's TS library), advanced point evaluation techniques (step-87), non-matching grids (step-89), and trace-based methods for PDEs on embedded surfaces (step-90). See Section 2.5 for more details.
- In Section 2.6 we summarize the motivation and approach behind our relicensing effort to make deal.II available under the terms of the the *Apache License 2.0 with LLVM Exception*.

While all of these major changes are discussed in detail in Section 2, there are a number of other noteworthy changes in the current deal.II release, which we briefly outline in the remainder of this section:

- deal.II now requires and makes use of the C++17 language standard.
- We have continued to make progress in supporting simplex and mixed meshes – mesh types that deal.II has traditionally not supported at all. Specifically, the current release uses better strategies for refinement of tetrahedra that result in better-shaped child cells. It also contains support for cubic finite elements on simplices.
- The FE_NedelecSZ class that contains our implementation of the Nédélec element using the orientation scheme of [68] now supports the computation of hanging node constraints for locally refined, hexahedral meshes. That is, the implementation of hanging node constraints now correctly addresses the sign-conflict that arises for *hp* quadrilateral and hexahedral Nédélec elements; see [45] for details. The other implementation of Nédélec elements, in

the `FE_Nedelec` class, already implements hanging node constraints; therefore, there is no longer a difference for the user between the two classes as far as constraints are concerned. However, the special case where, in 3D, more than four cells with different refinement levels share a common edge is not covered yet.

– The `AffineConstraints` class stores and processes constraints on degrees of freedom in `deal.II`. Such constraints can be of the (homogeneous) form $x_3 = \frac{1}{2}x_{14} + \frac{1}{2}x_{15}$ as is common when using hanging node constraints (here, x_3 could represent the value of the solution at a hanging node, and x_{14} and x_{15} are the values of the two adjacent degrees of freedom on the parent edge that contains the hanging node); or they can be of the (inhomogeneous) form $x_{12} = 42$ as is common when using Dirichlet boundary conditions.

  Since very early in the history of the library, the `AffineConstraints` class interface required building such constraints in multiple steps: First, one declared a degree of freedom as constrained; then one added the dependencies one after the other (e.g., by adding pairs $(\frac{1}{2}, 14)$ and $(\frac{1}{2}, 15)$ in the hanging node example above); then one added inhomogeneities (by setting it to 42 in the Dirichlet example above). This piecemeal approach is cumbersome and prevents the library from performing certain error checking steps because a constraint is not known to be completely built at any given point. The new `add_constraint()` function now allows defining a constraint in one step.

– We refactored the systems for managing relative line and face orientations to significantly improve consistency across several library modules, including finite element classes, periodicity, and the `p4est` interface. `deal.II` has supported using unstructured three-dimensional meshes for a long time. Features like discontinuous Galerkin methods and higher-order elements require that lines and faces of adjacent elements are consistently oriented. For example, consider a line containing two degrees of freedom, which, in reference coordinates, are located at $x_1 = \frac{1}{3}$ and $x_2 = \frac{2}{3}$. If the line is read from left-to-right then we get points (x_1, x_2) whereas if it is read from right-to-left we get (x_2, x_1). `deal.II` guarantees that this order will be consistent across all cells sharing that line by storing a flag indicating whether (relative to the vertex enumeration on the present cell) that particular line is in the standard or reversed orientation. Faces in 3d are treated in a similar way. The only significant difference between lines and 3d faces is that, instead of two possible relative orientations, triangular faces have six and quadrilateral faces have eight possible orientations.

  Historically, different places within the library implemented their own ad-hoc orientation logic, data encodings, and default values. We cleaned up this subsystem and instead of using three different booleans (*orientation*, *rotation*, and *flip*), bitsets, or several other encodings, we now use a single `unsigned char` which we named the *combined orientation*. Since the representation in some library interfaces has changed, most of the old interfaces have been deprecated and a few (such as the low-level interface to periodic boundary conditions) had to be incompatibly altered or removed.

– The `SolverGMRES` class now offers a third orthogonalization method, the classical Gram–Schmidt method with delayed orthogonalization [16]. Furthermore, the solver now specifies the maximal basis size of the Arnoldi basis, rather than the number of auxiliary vectors. Some changes have also been made to the GMRES and F-GMRES implementations, making them use the same underlying kernels as much as possible.

– The new class `FE_Hermite` implements a Hermite interpolation basis of maximum regularity. These bases are always of odd polynomial degree p and have regularity $r = (p-1)/2$.

The changelog – listing more than 180 features and bugfixes – contains a complete record of all changes; see [51].

2 Major changes to the library

This release of deal.II contains a number of large and significant changes, which we will discuss in this section.

2.1 Updates to the multigrid and matrix-free algorithms

We updated a number of significant parts of the multigrid and matrix-free infrastructure in deal.II. These changes include:

- Our own implementation of the std::experimental::simd class proposed for inclusion into the C++ standard, called VectorizedArray, now also supports Arm Neon. Arm Neon is an architecture extension of the ARMv7, ARMv8 and ARMv9 architecture families, used, e.g., for the Arm Cortex-A and Arm Cortex-R series of processors or in the Apple 64-bit silicon series (A7–A18, M1–M4 as of 2024). With these instructions, 2 doubles or 4 floats can be processed in one go. Since the matrix-free infrastructure works directly with the VectorizedArray abstraction as data structure, it automatically benefits from this new implementation.

- The application of the Piola transformation for values and gradients of $H(\mathrm{div})$-conforming Raviart–Thomas elements on non-Cartesian cells has been rewritten for better performance. Together with additional changes in the sum-factorization algorithms, which became more similar to the kernels described in [47], the matrix-free operator evaluation is now 3–5 times faster in these cases.

- Furthermore, we improved the internal data structures of the tensor-product evaluators as well as the evaluators for simplex elements. This speeds up the operator evaluation in several scenarios, especially for simplices (around two times higher throughput for operator evaluation) and multi-component systems. The restructuring that led to these optimizations also reduced the compile times and the size of the generated code slightly.

- In addition, we performed substantial improvements to the global-coarsening multigrid infrastructure: MGTransferMF (previously: MGTransferGlobalCoarsening) and MGTwoLevelTransfer. They now allow performing local smoothing, which is a key step towards unifying all transfer operators in deal.II. Finally, MGTwoLevelTransfer can now be set up with an existing MatrixFree object in the case of p-multigrid, reducing the setup costs and memory consumption significantly.

2.2 Advances in non-matching support

In the non-matching infrastructure of deal.II, we made the following improvements:

- The performance of the non-nested multigrid infrastructure (MGTwoLevelTransferNonNested) has been improved significantly by avoiding redundant copy operations. Furthermore, support for simplex-shaped cells and multiple-component elements has been added.

- Several minor performance improvements in deal.II's evaluator class on sets of unstructured points, FEPointEvaluation, have been made. The changes are particularly useful for evaluating several quantities on the same set of points, which is enabled by NonMatching::MappingInfo.

- We added the new class `FERemoteEvaluation`. This is a class to access data in a distributed matrix-free loop for non-matching discretizations. Interfaces are named in analogy to `FEEvaluation`, in order to seamlessly switch the local evaluator functionality at quadrature points of a computation. The key component is the underlying MPI communication infrastructure, which is performed via `RemotePointEvaluation`. Tutorial `step-89` has been added to present its usage in the context of the application to acoustic conservation equations [35].
- The `FECouplingValues` class provides a powerful tool for computing integrals of functions that contain finite element fields defined on different objects, for example if these fields live on separate grids or have different topological dimensions (e.g., cells, faces, edges). This is particularly useful in the following scenarios:
  - Non-local differential operators: Evaluating fractional Laplacian or boundary element methods.
  - Non-matching discretizations: Integrating data between two independent discretization schemes that overlap in some areas.
  - Bulk-surface coupling: Combining data from a bulk discretization and a surface discretization.

  The new class enables the combination of degrees of freedom indices, shape functions, and quadrature points from two existing `FEValuesBase` objects. The way this combination is performed is controlled by user-provided `DoFCouplingType` and `QuadratureCouplingType` objects, which define how the degrees of freedom and quadrature points are combined.

  The flexibility offered by these coupling types allows users to:
  - Rearrange quadrature points in a tensor product structure for double integration.
  - Reorder quadrature points to easily access shape function values and gradients from two different finite element spaces located on the same point.
  - Identify and integrate over a reordered subset of the quadrature points.

2.3 Interface to the `Trilinos Tpetra` stack

`deal.II`'s parallel linear algebra facilities (apart from matrix-free capabilities) are largely built on wrappers around functionality provided by the `PETSc` and `Trilinos` libraries.

Historically, `Trilinos` has implemented distributed linear algebra classes for vectors and (sparse) matrices in its `Epetra` package that uses `MPI` as its only source of parallelism. Building on that, there are multiple interconnected `Trilinos` packages, e.g. for (non-)linear solvers and preconditioners, commonly referred to as the "`Epetra` stack". For sake of brevity we will also refer to the whole stack as `Epetra` in the following. `deal.II`'s interfaces to `Trilinos` have traditionally been implemented to use it.

However, several years ago, `Trilinos` also introduced the newer `Tpetra` ("templated" Petra) package that provides additional shared memory parallelism and GPU capabilities, referred to as `MPI+X`, by building on `Kokkos`. In recent years, new features have only been implemented in `Tpetra` as it is slated to replace `Epetra`, and the latter is indeed now deprecated with removal slated for 2025. As a consequence, we will eventually have to switch all of our `Trilinos` interfaces to `Tpetra`; the same is true for the need to switch to `Tpetra`-based sub-packages (the "Tpetra stack") that are intended to replace existing `Trilinos` sub-packages (for example, `Ifpack2` instead of `Ifpack`).

In the current release, we have put substantial work into this switch, as outlined below. As this is not yet finished, it is important to note that all these features are optional and their individual availability depends on the installed `Trilinos` packages. As an example, a `deal.II` build will pass with a `Trilinos` installation without `Ifpack2`, but its preconditioners will not be included.

In the following list, we omit the common namespace prefix `LinearAlgebra::` on all mentioned `TpetraWrappers` symbols for readability:

- The `TpetraWrappers::Vector` class has been overhauled to also allow specifying a memory space, and the `TpetraWrappers::BlockVector` class has been added.
- `TpetraWrappers::SparseMatrix` and `TpetraWrappers::SparsityPattern` have been implemented mirroring the functionality of the `TrilinosWrappers::SparseMatrix` and `TrilinosWrappers::SparsityPattern` classes.
- Various incomplete factorization and relaxation preconditioners (from `Ifpack2`) have been wrapped, mirroring the existing classes in `LinearAlgbera::TrilinosWrappers` as closely as possible.
- Two new preconditioner variants `TpetraWrappers::PreconditionL1Jacobi` and `TpetraWrappers::PreconditionL1GaussSeidel`, based on [10], have been added.
- `TpetraWrappers::SolverDirectKLU2` is now available as a direct solver.

Some functionality in these wrapper classes is still missing, most noticeably wrappers for the algebraic-multigrid preconditioner `MueLu` and the iterative solvers from `Belos`. However, the wrapped `Ifpack2` preconditioners can already be used with the iterative solvers of `deal.II`. To do this you will need to explicitly specify the vector type parameter, e.g.
`SolverCG<LinearAlgebra::TpetraWrappers::Vector<double, MemorySpace::Default> >`

The solver and preconditioner classes mentioned above provide reasonable parameter subsets through `AdditionalData` objects, just as the existing `Epetra` wrappers. Additionally, there are new generic classes `TpetraWrappers::SolverDirect` and `TpetraWrappers::PreconditionIfpack` which expose the internal interface through a `Teuchos::ParameterList` and thereby offering the full set of parameters for more experienced `Trilinos` users. This also allows the use of preconditioners or solvers not (yet) wrapped, e.g. `SuperLU_dist` or `MUMPS`.

The design goal was to introduce as few changes as possible for the user to allow for an easy transition to the `TpetraWrappers` classes. However, based on the difference between `Epetra` and `Tpetra`, the following changes were necessary:

- The constructor for the class `TrilinosWrappers::MPI::Vector` only required the `IndexSet locally_owned`, which describes the set of indices locally owned by the current rank. However, if one wants to create a vector that has read or write access to non-locally owned indices, the constructor of the `TpetraWrappers::Vector` requires the `IndexSet locally_owned`, the `IndexSet locally_relevant`, and a boolean flag, whether to initialize the vector in the read-only or the write-only state. This interface is in fact close to the one of `deal.II`'s own parallel vector class.
- A vector created without providing the `IndexSet locally_relevant` is purely local and cannot access non-local indices. Such a vector cannot be copied to a vector that can access non-local indices, as the `IndexSet locally_relevant` must be provided at creating the vector object.
- Some parameters of existing solver and preconditioner are not available in the `Tpetra` wrappers, such that the corresponding `AdditionalData` objects are not identical. Since there are many preconditioners we will not list each individual change but instead refer to our Doxygen documentation.
- The incomplete Cholesky (IC) factorization preconditioner is not available in `Tpetra`.

Another goal was increasing the interoperability of deal.II and Trilinos, allowing users familiar with both to write 'pure' Trilinos code within their applications, e.g. to test or develop a new feature. Therefore, the internal data is stored as it would be in a Trilinos application code (as Teuchos::RCP) and can be accessed through member functions.

Finally, all TpetraWrappers classes take Number and MemorySpace template arguments, in order to control the underlying scalar type and memory space. By default, memory is allocated on the CPU, mirroring the behavior of the TrilinosWrappers, but choosing a different memory space also allows Tpetra to utilize GPUs if Kokkos is configured with a GPU backend. Furthermore, to fully leverage the GPU backend and unlock its full potential for users will require further work. The support for different number types in vector and matrix classes also enables the usage of automatic differentiation-based tools in Trilinos, such as NOX. To make the automatic differentiation tools of Trilinos accessible through the Tpetra interface without requiring users to have an in-depth understanding of Trilinos, additional wrappers for the corresponding Trilinos packages will be added in future releases.

2.4 More support for advanced programming idioms

Over the years, deal.II has accumulated many classes and functions that support modern programming idioms and make it easier to write code at higher levels of abstraction.

In the current release, we have added two classes to the list of tools of this kind:

- Lazy<T> is a class that supports the lazy computation and initialization of variables. Its intended use is for member variables of classes that are *sometimes* needed, but perhaps not for all uses of an object. For example, all finite element classes provide interpolation and restriction matrices to support multigrid and other algorithms. One could (i) always compute and store these matrices in the constructor of the class; or one could (ii) re-compute these matrices every time they are requested. The first of these approaches costs memory and compute time even though most places where one creates a finite element object will not actually query these matrices; the second of these approaches is costly in places that *do* query these matrices repeatedly because they are re-computed every time. Lazy<T> provides a middle ground: It provides an abstraction for an object that is initialized upon first use (that is, the first time the value is requested), and then stores the computed value for cheap use later on.

  (C++ provides functionality via std::async with launch policy std::launch::deferred that can achieve similar outcomes. But this functionality is more difficult to use than Lazy<T> because, among other reasons, the code generating the object has to be specified at the place of construction of the object holding the result, rather than at the place of use; and because the holder object – std::future – can only be asked *once* for its computed value.)

- TaskResult<T> is a class that represents the outcome of a task possibly evaluated on a separate thread. It can be thought of as a "deferred" result of a computation in that one wants to state "This job needs to be done, do it when convenient, and then put the result of the operation into this variable". Accessing the variable then waits for the operation to complete, if it has not already. TaskResult<T> allows classes to efficiently compute member variables in the background, assuming that they may not be needed right away but only later on.

  (Similar to above, the same effect as TaskResult<T> can be achieved using std::async, this time using the launch policy std::launch::async. This approach suffers from the same issue that one can only query the resulting object once. Moreover, std::async does not integrate with the thread pool that underlies deal.II's approach to parallel processing on modern multi-core machines, whereas TaskResult<T> does.)

2.5 New and improved tutorials and code gallery programs

Many of the deal.II tutorial programs were revised in a variety of ways as part of this release: Around 190 of the more than 1900 (non-merge) commits that went into this release touched the tutorial. In addition, there are a number of new tutorial programs:

- step-83 demonstrates how one can implement checkpoint/restart functionality in deal.II-based programs, using the Boost serialization functionality as a foundation. step-83 was written by Pasquale Africa, Wolfgang Bangerth, and Bruno Blais using step-19 as its basis.
- step-86 is a program that solves the heat equation using PETSc's TS (time stepping) framework for the solution of ordinary differential equations. Written by Wolfgang Bangerth, Luca Heltai, and Stefano Zampini (King Abdullah University of Science and Technology), it illustrates how PDE solvers for time-dependent problems can be integrated with existing ODE solver packages to use advanced ODE solver concepts (such as higher-order time integration methods and adaptive time step control), all without sacrificing the things that have traditionally led code authors toward writing their own time stepping routines (like wanting to change the mesh every once in a while, or having to deal with boundary conditions).
- step-87 [61] was contributed by Magdalena Schreter-Fleischhacker and Peter Munch. It presents the advanced point-evaluation functionalities of deal.II, which are useful for evaluating finite element solutions at arbitrary points on meshes that can be distributed among processes. The presented mini-examples are motivated by the application to two-phase flow simulations and demonstrate, for example, the evaluation of solution quantities at a surface mesh embedded in a background mesh, as needed in the case in front-tracking.
- step-89 was contributed by Johannes Heinz (TU Wien), Maximilian Bergbauer (Technical University of Munich), Marco Feder (SISSA), and Peter Munch. It shows how to apply non-matching and/or Chimera methods within matrix-free loops in deal.II.
- step-90 was contributed by Vladimir Yushutin and Timo Heister. It implements the trace finite element method (TraceFEM). TraceFEM solves PDEs posed on a, possibly evolving, $(dim - 1)$-dimensional surface Γ employing a fixed uniform background mesh of a dim-dimensional domain in which the surface is embedded. Such surface PDEs arise in problems involving material films with complex properties and in other situations in which a non-trivial condition is imposed on either a stationary or a moving interface. The program considers a steady, complex, non-trivial surface and the prototypical Laplace-Beltrami equation which is a counterpart of the Poisson problem on flat domains.

In addition, there are three new programs in the code gallery (a collection of user-contributed programs that often solve more complicated problems than tutorial programs, and that are intended as starting points for further research rather than as teaching tools):

- *"Crystal growth phase field model"*, contributed by Umair Hussain;
- *"Nonlinear heat transfer problem"*, contributed by Narasimhan Swaminathan;
- *"Traveling-wave solutions of a qualitative model for combustion waves"*, contributed by Shamil Magomedov.

Furthermore, we added an example to the libCEED library [20]: https://github.com/CEED/libCEED/tree/main/examples/deal.II. libCEED is a library that provides matrix-free evaluation routines for different hardware. The example shows how to interface the deal.II data structures with the libCEED ones and solves the BP1-BP6 benchmarks (scalar/vector Laplace/mass matrix with regular integration and over-integration).

2.6 Relicensing to Apache License 2.0 with LLVM exception

The deal.II project is in the process of relicensing the library under the terms of the *Apache License 2.0 with LLVM Exception* [6, 52]. To this end we require all new code contributions to be dual licensed under the current license (GNU Lesser General Public License v2.1 [50]) and the new license (Apache-2.0 with LLVM-exception).

We have chosen to pursue a relicensing effort for deal.II because of some long standing problems with the LGPL v2.1 license. Most notably, despite our clear intent that the strong copyleft principle should only apply to deal.II source code and not user projects (which should be free to choose their own license freely) this is not necessarily the case for LGPL v2.1 due to the language used in the library. This makes potential industry partners that evaluate the use of deal.II for their own projects nervous. Secondly, our previous choice to use an open source license with strong copyleft was motivated by our hope that we receive back code contributions from third parties. In practice, however, our observation over the last 10 years is that our choice of license has not helped achieve this goal: Copyleft only applies when a derived software is sold or published. But almost all of our code contributions are voluntary and come from individual contributors with an academic background, where these considerations do not apply.

We have thus decided to switch away from the LGPL v2.1 license towards a more permissible license. We settled on the Apache License 2.0 with LLVM-exception in large parts because it is (a) a permissible open source license with patent clause, (b) considered to be a verbose, well-written license, and (c) has been recently chosen by other large open source projects, notably LLVM and Kokkos. Our hope going forward is that the new license makes it easier for industry partners to use the library.

As a first step we now require all new code contributions to be dual licensed under the old and new licenses. As a second step we are in the process of contacting the more than 200 past contributors with copyrighted contributions to the library. As of September 2024 roughly 80% of the commits and 80% of the 1.88 million source lines of code have been relicensed.

2.7 Incompatible changes

The 9.6 release includes around 40 incompatible changes; see [51]. Many of these incompatibilities change internal interfaces that are not usually used in external applications. That said, the following are worth mentioning since they are more broadly visible:

- deal.II now requires compilers to support C++17, and has started to extensively use C++17 features.
- The CUDAWrappers namespace and its contents – notably things that enable the usage of cuSPARSE algorithms – have been deprecated and will be removed in the next release. Kokkos is now used for device-specific optimizations.

3 How to cite deal.II

In order to justify the work the developers of deal.II put into this software, we ask that papers using the library reference one of the deal.II papers. This helps us justify the effort we put into this library.

There are various ways to reference deal.II. To acknowledge the use of the current version of the library, **please reference the present document**. For up-to-date information and a bibtex entry see

https://www.dealii.org/publications.html

The original deal.II paper containing an overview of its architecture is [14], and a more recent publication documenting deal.II's design decisions is available as [8]. If you rely on specific features of the library, please consider citing any of the following:

- For geometric multigrid: [43, 42, 22, 55];
- For distributed parallel computing: [13];
- For *hp*-adaptivity: [15, 28];
- For partition-of-unity (PUM) and finite element enrichment methods: [26];
- For matrix-free and fast assembly techniques: [46, 47];
- For computations on lower-dimensional manifolds: [27];
- For curved geometry representations and manifolds: [36];
- For integration with CAD files and tools: [37];
- For boundary element computations: [32];
- For the `LinearOperator` and `PackagedOperation` facilities: [53, 54];
- For uses of the `WorkStream` interface: [67];
- For uses of the `ParameterAcceptor` concept, the `MeshWorker::ScratchData` base class, and the `ParsedConvergenceTable` class: [60];
- For uses of the particle functionality in deal.II: [30].

deal.II can interface with many other libraries:

- ADOL-C [33]
- ArborX [48]
- ARPACK [49]
- Assimp [62]
- BLAS and LAPACK [3]
- Boost [19]
- CGAL [64]
- cuSOLVER [23]
- cuSPARSE [24]
- Gmsh [31]
- GSL [29, 34]
- Ginkgo [4, 5]
- HDF5 [65]
- METIS [44]
- MUMPS [2, 1]
- muparser [56]
- OpenCASCADE [57]
- p4est [21]
- PETSc [11, 12]
- ROL [59]
- ScaLAPACK [17]
- SLEPc [38]
- SUNDIALS [40]
- SymEngine [63]
- Taskflow [41]
- TBB [58]
- Trilinos [39, 66]
- UMFPACK [25]

Please consider citing the appropriate references if you use interfaces to these libraries.

The two previous releases of deal.II can be cited as [9, 7].

4 Acknowledgments

deal.II is a worldwide project with dozens of contributors around the globe. Other than the authors of this paper, the following people contributed code to this release:

Laryssa Abdala, Mathias Anselmann, Abbas Ballout, Maximilian Bergbauer, Julian Brotz, Marco Feder, Niklas Fehn, Menno Fraters, Quang Hoang, Vladimir Ivannikov, Tao Jin, Yimin Jin, Paras Kumar, Sébastien Loriot, Nils Much, Abdullah Mujahid, Bob Myhill, Paul A. Patience, Luz Paz, Laura Prieto Saavedra, Sebastian Proell, Hendrik Ranocha, Johannes Resch, Andreas Ritthaler, Malik Scheifinger, David Schneider, Richard Schussnig, Nils Schween, Kyle Schwiebert, Simranjeet Singh, Simon Sticko, Dominik Still, Thierry Thomas, Vinayak Vijay, Ivy Weber, Simon Wiesheier, Chengjiang Yin, Stefano Zampini.

Their contributions are much appreciated!

`deal.II` and its developers are financially supported through a variety of funding sources:

P. C. Africa was partially supported by the consortium iNEST (Interconnected North-East Innovation Ecosystem), Piano Nazionale di Ripresa e Resilienza (PNRR) - Missione 4 Componente 2, Investimento 1.5 - D.D. 1058 23/06/2022, ECS00000043, supported by the European Union's NextGenerationEU program.

D. Arndt and B. Turcksin: Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy.

W. Bangerth and M. Fehling were partially supported by Award OAC-1835673 as part of the Cyberinfrastructure for Sustained Scientific Innovation (CSSI) program.

W. Bangerth, T. Heister, and R. Gassmöller were partially supported by the Computational Infrastructure for Geodynamics initiative (CIG), through the National Science Foundation (NSF) under Award No. EAR-2149126 via The University of California – Davis.

W. Bangerth was also partially supported by Award EAR-1925595.

B. Blais was supported by the National Science and Engineering Research Council of Canada (NSERC) through the RGPIN-2020-04510 Discovery Grant and the MMIAOW Canada Research Level 2 in Computer-Assisted Design and Scale-up of Alternative Energy Vectors for Sustainable Chemical Processes.

M. Fehling was also partially supported by the ERC-CZ grant LL2105 CONTACT, funded by the Czech Ministry of Education, Youth and Sports.

R. Gassmöller was also partially supported by NSF Awards EAR-1925677 and EAR-2054605.

T. Heister and V. Yushutin were also partiallty supported by NSF Awards OAC-2015848 and EAR-1925575.

T. Heister was also partially supported by NSF OAC-2410848.

L. Heltai was partially supported by the Italian Ministry of University and Research (MUR), under the grant MUR PRIN 2022 No. 2022WKWZA8 "Immersed methods for multiscale and multiphysics problems (IMMEDIATE)".

S. Kinnewig was supported by the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122, Project ID 390833453)

M. Kronbichler and P. Munch were partially supported by the German Ministry of Education and Research, project "PDExa: Optimized software methods for solving partial differential equations on exascale supercomputers" and the Bayerisches Kompetenznetzwerk für Technisch-Wissenschaftliches Hoch- und Höchstleistungsrechnen (KONWIHR), project "Fast and scalable finite element algorithms for coupled multiphysics problems and non-matching grids".

M. Maier was partially supported by NSF Award DMS-2045636 and and by the Air Force Office of Scientific Research under grant/contract number FA9550-23-1-0007.

P. Munch acknowledges the funding by the Swedish Research Council (VR) under grant 2021-04620 and the strategic funding from the IT department of Uppsala University.

M. Schreter-Fleischhacker was supported by the Austrian Science Fund (FWF) Schrödinger Fellowship (project number: J4577) and by the European Research Council through the ERC Starting Grant ExcelAM (project number: 101117579).

D. Wells was supported by NSF Award OAC-1931516.

Clemson University is acknowledged for generous allotment of compute time on the Palmetto cluster.

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. http://www.tacc.utexas.edu

This work used the *Expanse* HPC system at the San Diego Supercomputer Center (SDSC) at UC San Diego through the CIG Science Gateway and Community Codes for the Geodynamics Community MCA08X011 allocation from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296. See [18].

References

[1] 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(1):2/1–26, 2019.

[2] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[3] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[4] H. Anzt, T. Cojean, Y.-C. Chen, G. Flegar, F. Göbel, T. Grützmacher, P. Nayak, T. Ribizel, and Y.-H. Tsai. Ginkgo: A high performance numerical linear algebra library. *Journal of Open Source Software*, 5(52):2260, 2020.

[5] H. Anzt, T. Cojean, G. Flegar, F. Göbel, T. Grützmacher, P. Nayak, T. Ribizel, Y. M. Tsai, and E. S. Quintana-Ortí. Ginkgo: A modern linear operator algebra framework for high performance computing. *ACM Transactions on Mathematical Software*, 48(1):2/1–33, 2022.

[6] Apache License 2.0. https://spdx.org/licenses/Apache-2.0.html.

[7] D. Arndt, W. Bangerth, M. Bergbauer, M. Feder, M. Fehling, J. Heinz, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, B. Turcksin, D. Wells, and S. Zampini. The `deal.II` library, version 9.5. *Journal of Numerical Mathematics*, 31(3):231–246, 2023.

[8] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells. The deal.II finite element library: Design, features, and insights. *Computers & Mathematics with Applications*, 81:407–422, 2021.

[9] D. Arndt, W. Bangerth, M. Feder, M. Fehling, R. Gassmöller, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, S. Sticko, B. Turcksin, and D. Wells. The `deal.II` library, version 9.4. *Journal of Numerical Mathematics*, 30(3):231–246, 2022.

[10] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang. Multigrid smoothers for ultraparallel computing. *SIAM Journal on Scientific Computing*, 33(5):2864–2887, 2011.

[11] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, and J. Zhang. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.17, Argonne National Laboratory, 2022.

[12] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, and J. Zhang. PETSc Web page. https://petsc.org/, 2022.

[13] W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler. Algorithms and data structures for massively parallel generic adaptive finite element codes. *ACM Transactions on Mathematical Software*, 38(2):14/1–28, 2012.

[14] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II — a general purpose object oriented finite element library. *ACM Transactions on Mathematical Software*, 33(4):24–es, 2007.

[15] W. Bangerth and O. Kayser-Herold. Data structures and requirements for *hp* finite element software. *ACM Transactions on Mathematical Software*, 36(1):4/1–31, 2009.

[16] D. Bielich, J. Langou, S. Thomas, K. Świrydowicz, I. Yamazaki, and E. G. Boman. Low-synch Gram–Schmidt with delayed reorthogonalization for Krylov solvers. *Parallel Computing*, 112:102940, 2022.

[17] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[18] T. J. Boerner, S. Deems, T. R. Furlani, S. L. Knuth, and J. Towns. ACCESS: Advancing Innovation: NSF's Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support. In *Practice and Experience in Advanced Research Computing*, PEARC '23. ACM, July 2023.

[19] Boost C++ Libraries. http://www.boost.org/.

[20] J. Brown, A. Abdelfattah, V. Barra, N. Beams, J.-S. Camier, V. Dobrev, Y. Dudouit, L. Ghaffari, T. Kolev, D. Medina, et al. libceed: Fast algebra for high-order element-based discretizations. *Journal of Open Source Software*, 6(63):2945, 2021.

[21] C. Burstedde, L. C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM J. Sci. Comput.*, 33(3):1103–1133, 2011.

[22] T. C. Clevenger, T. Heister, G. Kanschat, and M. Kronbichler. A flexible, parallel, adaptive geometric multigrid method for FEM. *ACM Transactions on Mathematical Software*, 47(1):7/1–27, 2021.

[23] cuSOLVER Library. https://docs.nvidia.com/cuda/cusolver/index.html.

[24] cuSPARSE Library. https://docs.nvidia.com/cuda/cusparse/index.html.

[25] T. A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30:196–199, 2004.

[26] D. Davydov, T. Gerasimov, J.-P. Pelteret, and P. Steinmann. Convergence study of the *h*-adaptive PUM and the *hp*-adaptive FEM applied to eigenvalue problems in quantum mechanics. *Advanced Modeling and Simulation in Engineering Sciences*, 4(1):7, Dec 2017.

[27] A. DeSimone, L. Heltai, and C. Manigrasso. Tools for the solution of PDEs defined on curved manifolds with deal.II. Technical Report 42/2009/M, SISSA, 2009.

[28] M. Fehling and W. Bangerth. Algorithms for parallel generic *hp*-adaptive finite element software. *ACM Transactions on Mathematical Software*, 49(3):25/1–26, 2023.

[29] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi. *GNU Scientific Library Reference Manual*. Network Theory Ltd., 3rd edition, 2009.

[30] R. Gassmöller, H. Lokavarapu, E. Heien, E. G. Puckett, and W. Bangerth. Flexible and scalable particle-in-cell methods with adaptive mesh refinement for geodynamic computations. *Geochemistry, Geophysics, Geosystems*, 19(9):3596–3604, 2018.

[31] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

[32] N. Giuliani, A. Mola, and L. Heltai. π-BEM: A flexible parallel implementation for adaptive, geometry aware, and high order boundary element methods. *Advances in Engineering Software*, 121:39–58, July 2018.

[33] A. Griewank, D. Juedes, and J. Utke. Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software*, 22(2):131–167, 1996.

[34] GSL: GNU Scientific Library. http://www.gnu.org/software/gsl.

[35] J. Heinz, P. Munch, and M. Kaltenbacher. High-order non-conforming discontinuous Galerkin methods for the acoustic conservation equations. *International Journal for Numerical Methods in Engineering*, 124(9):2034–2049, 2023.

[36] L. Heltai, W. Bangerth, M. Kronbichler, and A. Mola. Propagating geometry information to finite element computations. *ACM Transactions on Mathematical Software*, 47(4):32/1–30, 2021.

[37] L. Heltai and A. Mola. Towards the Integration of CAD and FEM using open source libraries: a Collection of deal.II Manifold Wrappers for the OpenCASCADE Library. Technical report, SISSA, 2015.

[38] V. Hernandez, J. E. Roman, and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software*, 31(3):351–362, 2005.

[39] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley. An overview of the Trilinos project. *ACM Transactions on Mathematical Software*, 31:397–423, 2005.

[40] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31(3):363–396, 2005.

[41] T.-W. Huang, D.-L. Lin, C.-X. Lin, and Y. Lin. Taskflow: A lightweight parallel and heterogeneous task graph computing system. *IEEE Transactions on Parallel and Distributed Systems*, 33(6):1303–1320, 2021.

[42] B. Janssen and G. Kanschat. Adaptive multilevel methods with local smoothing for H^1- and H^{curl}-conforming high order finite element methods. *SIAM J. Sci. Comput.*, 33(4):2095–2114, 2011.

[43] G. Kanschat. Multi-level methods for discontinuous Galerkin FEM on locally refined meshes. *Comput. & Struct.*, 82(28):2437–2445, 2004.

[44] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[45] S. Kinnewig, T. Wick, and S. Beuchler. Algorithmic realization of the solution to the sign conflict problem for hanging nodes on hp-hexahedral nédélec elements, 2024.

[46] M. Kronbichler and K. Kormann. A generic interface for parallel cell-based finite element operator application. *Comput. Fluids*, 63:135–147, 2012.

[47] M. Kronbichler and K. Kormann. Fast matrix-free evaluation of discontinuous Galerkin finite element operators. *ACM Transactions on Mathematical Software*, 45(3):29/1–40, 2019.

[48] D. Lebrun-Grandié, A. Prokopenko, B. Turcksin, and S. R. Slattery. ArborX: A performance portable geometric search library. *ACM Transactions on Mathematical Software*, 47(1):2/1–15, 2020.

[49] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, Philadelphia, 1998.

[50] GNU Lesser Public License v2.1 or later. https://spdx.org/licenses/LGPL-2.1-or-later.htmll.

[51] List of changes for `deal.II` release 9.6. https://dealii.org/developer/doxygen/deal.II/changes_between_9_5_2_and_9_6_0.html.

[52] LLVM Exception. https://spdx.org/licenses/LLVM-exception.html.

[53] M. Maier, M. Bardelloni, and L. Heltai. LinearOperator – a generic, high-level expression syntax for linear algebra. *Computers and Mathematics with Applications*, 72(1):1–24, 2016.

[54] M. Maier, M. Bardelloni, and L. Heltai. LinearOperator Benchmarks, Version 1.0.0, 2016.

[55] P. Munch, T. Heister, L. Prieto Saavedra, and M. Kronbichler. Efficient distributed matrix-free multigrid methods on locally refined meshes for FEM computations. *ACM Transactions on Parallel Computing*, 10(1):3/1–38, 2023.

[56] muparser: Fast Math Parser Library. https://beltoforion.de/en/muparser.

[57] OpenCASCADE: Open CASCADE Technology, 3D modeling & numerical simulation. http://www.opencascade.org/.

[58] J. Reinders. *Intel Threading Building Blocks*. O'Reilly, 2007.

[59] D. Ridzal and D. P. Kouri. Rapid optimization library. Technical report, Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), 2014.

[60] A. Sartori, N. Giuliani, M. Bardelloni, and L. Heltai. deal2lkit: A toolkit library for high performance programming in deal.II. *SoftwareX*, 7:318–327, 2018.

[61] M. Schreter-Fleischhacker and P. Munch. The deal.II tutorial step-87: evaluation of finite element solutions at arbitrary points within a distributed mesh with application to two-phase flow, oct 2023.

[62] T. Schulze, A. Gessler, K. Kulling, D. Nadlinger, J. Klein, M. Sibly, and M. Gubisch. Open asset import library (assimp). https://github.com/assimp/assimp, 2021.

[63] SymEngine: fast symbolic manipulation library, written in C++. https://symengine.org/.

[64] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.4.1 edition, 2022. https://doc.cgal.org/5.4.1/Manual/packages.html.

[65] The HDF Group. Hierarchical Data Format, version 5, 2022. http://www.hdfgroup.org/HDF5/.

[66] The Trilinos Project Team. *The Trilinos Project Website*. https://trilinos.github.io/.

[67] B. Turcksin, M. Kronbichler, and W. Bangerth. *WorkStream* – a design pattern for multicore-enabled finite element computations. *ACM Transactions on Mathematical Software*, 43(1):2/1–29, 2016.

[68] S. Zaglmayr. *High Order Finite Element Methods for Electromagnetic Field Computation*. PhD thesis, Johannes Kepler University, Linz, Austria, 2006.