The deal.II Library, Version 9.0

Giovanni Alzetta[1], Daniel Arndt[2], Wolfgang Bangerth[3], Vishal Boddu[4], Benjamin Brands[4], Denis Davydov[4], Rene Gassmöller[5], Timo Heister[6], Luca Heltai[1], Katharina Kormann[7], Martin Kronbichler[8], Matthias Maier[9], Jean-Paul Pelteret[4], Bruno Turcksin*[10], and David Wells[11]

[1]SISSA, International School for Advanced Studies, Via Bonomea 265, 34136, Trieste, Italy. {galzetta,luca.heltai}@sissa.it
[2]Interdisciplinary Center for Scientific Computing, Heidelberg University, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany. daniel.arndt@iwr.uni-heidelberg.de
[3]Department of Mathematics, Colorado State University, Fort Collins, CO 80523-1874, USA. bangerth@colostate.edu
[4]Chair of Applied Mechanics, Friedrich-Alexander-Universität Erlangen-Nürnberg, Egerlandstr. 5, 91058 Erlangen, Germany. {vishal.boddu,benjamin.brands,denis.davydov,jean-paul.pelteret}@fau.de
[5]Department of Earth and Planetary Sciences, University of California Davis, One Shields Avenue, CA-95616 Davis, USA. rgassmoeller@ucdavis.edu
[6]Mathematical Sciences, O-110 Martin Hall, Clemson University, Clemson, SC 29634, USA. heister@clemson.edu
[7]Max Planck Institute for Plasma Physics, Boltzmannstr. 2, 85748 Garching, Germany. katharina.kormann@ipp.mpg.de
[8]Institute for Computational Mechanics, Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany. kronbichler@lnm.mw.tum.de
[9]School of Mathematics, University of Minnesota, 127 Vincent Hall, 206 Church Street SE, Minneapolis, MN 55455, USA. msmaier@umn.edu
[10]Computational Engineering and Energy Sciences Group, Computional Sciences and Engineering Division, Oak Ridge National Laboratory, 1 Bethel Valley Rd., TN 37831, USA. turcksinbr@ornl.gov
[11]Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, USA. wellsd2@rpi.edu

Abstract: This paper provides an overview of the new features of the finite element library deal.II version 9.0.

1 Overview

deal.II version 9.0.0 was released May 11, 2018.

This paper provides an overview of the new features of this major release and serves as a citable reference for the deal.II software library version 9.0. 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 GNU Lesser General Public License (LGPL) from the deal.II homepage at http://www.dealii.org/.

The major changes of this release are:

– Improved support for curved geometries;

– Support for particle-in-cell methods;

– Dedicated support for automatic differentiation;

– Interfaces to more external libraries and programs;

– C++11 is now both required and used;

– Support for GPU computations;

– Support for face integrals and significant improvements of the matrix-free framework;

These will all be discussed in more detail in the following section. In addition, this release contains the following changes:

– deal.II has made extensive use of both the Clang-Tidy [47] and Coverity Scan [23] static analysis tools for detecting bugs and other issues in the code. For example, around 260 issues were detected and fixed using the latter tool.

– LinearOperator, a flexible template class that implements the action of a linear operator (see [49]), now supports computations with Trilinos, Schur complements, and linear constraints. This class is, as of this release, the official replacement for about half a dozen similar (but less general) classes, such as FilteredMatrix, IterativeInverse, and PointerMatrix.

– New non-standard quadrature rules: A number of non-standard, special-purpose quadrature rules have been implemented. Among these are ones for (i) truncating standard formulas to simplical domains (QSimplex); (ii) singular transformations of the unit cell to the unit simplex (QDuffy); (iii) composition of simplical quadrature rules to a combined rule on the unit cell (QSplit); and (iv) transformation of the unit square to polar coordinates (QTrianglePolar). These quadrature rules greatly help when integrating singular functions or on singular domains. They are mainly used in Boundary Element Methods.

– Support for complex-valued vectors at the same level as real-valued vectors.

– A new python tutorial program tutorial-1; as well as updates to step-37. In addition, the separate code gallery of deal.II has gained a number of new entries.

– Improved support for user-defined run-time parameters: a new class for handling parameters, ParameterAcceptor, has been added to the library. The class is intended to be used as a base for any class that wants to handle parameters using the ParameterHandler class. If you derive all your classes from ParameterAcceptor, and declare your parameters either with parse/declare_parameters methods or via the ParameterAcceptor::add_parameter method, then both the declaration and the parsing of parameter files are automatically managed by ParameterAcceptor::initialize, greatly simplifying dealing with parameters in user codes.

– New caching mechanism for expensive grid computations: the new class GridTools::Cache stores computationally intensive information about a Triangulation, such as looking up a cell for a given vertex. This class allows the user to query some of the data structures constructed using functions in the GridTools namespace. This data is then computed only once, and then cached inside this class for faster access whenever the triangulation has not changed. The cache is marked for update by the Triangulation itself using signals. Many of the functions in GridTools now use this cache to speed up repeated calls to the same expensive methods.

– A new MeshWorker::mesh_loop function has been added that performs the same tasks of the MeshWorker::loop function without forcing the users to adhere to a specific interface.

Beyond these changes, the changelog lists more than 330 other features and bugfixes.

2 Significant changes to the library

This release of deal.II contains a number of large and significant changes that will be discussed in the following sections. It of course also contains a vast number of smaller changes and added functionality; the details of these can be found in the file that lists all changes for this release, see [46]. (The file is also linked to from the web site of each release as well as the release announcement.)

2.1 Improved support for curved geometries

deal.II has had the ability to attach *manifold descriptions* to all parts of a geometry since the 8.2 release. These descriptions are used to place new vertices during mesh refinement, to determine the mapping between the reference and real cells, and in a number of other contexts. These classes, inheriting from Manifold, describe coordinate transformations in a general way and completely supersede the older classes inheriting from Boundary. However, for historical reasons, manifold descriptors have used some of the same code paths as boundary indicators, which were only intended for marking what parts of the boundary correspond to what boundary conditions. Put another way: under certain circumstances a boundary indicator was also interpreted as a manifold indicator.

The current release severs this connection: Boundary indicators and manifold descriptions are now entirely separated. This means that boundary_ids are only used to set boundary conditions and manifold_ids are only used to set geometry descriptions. The old compatibility code for using boundary indicators as manifold indicators has been removed and all usages of the old-style Boundary objects (even with manifold ids) are now deprecated.

There are also numerous improvements to the available manifold descriptions. First, the manifold smoothing algorithms applied in the Triangulation class and MappingQGeneric have been changed from the old Laplace-style smoothing to a transfinite interpolation that linearly blends between the descriptions on the faces around a cell. The old transformation introduced boundary layers inside cells that prevented convergence rates from exceeding 3.5 in the global L^2 errors for typical settings. This change also considerably improves mesh quality in situations where curved descriptions are only applied to the boundary rather than the whole volume. This concept was also introduced as a new manifold class TransfiniteInterpolationManifold, which allows to apply this type of smoothing not only in the cells close to the boundary but over a full coarse (level 0) cell.

Finally, every function in the GridGenerator namespace now attaches a default manifold to the curved parts of the domain described by the generated mesh, and sets reasonable defaults for manifold indicators both in the domain and on the boundary.

2.2 Support for particle-in-cell methods

While `deal.II` is a package intended to solve problems with the finite element method – i.e., using continuous or discontinuous *fields* –, it is often convenient in fluid dynamics problems to couple the continuum description of phenomena with particles. These particles, advected along with the numerical approximation of the flow field, are then either used to visualize properties of the flow, or to advect material properties such as the viscosity of inhomogeneous mixtures of fluids. If each particle is associated with the cells of a mesh, these methods are often referred to as particle-in-cell (PIC).

`deal.II` now has a dedicated particles module. The module provides a base class `Particle` that represents a particle with position, an ID number and a variable number of properties. They are jointly represented by a `ParticleHandler` class that manages the storage and handling of all particles. In parallel simulations, this class also distributes the particles among the subdomains of the parallel process and supports efficient data transfer during mesh refinement and checkpoint/restart phases.

A much more detailed view of the underlying algorithms can be found in [29]. A longer report is at [28]. The implementation here originated in the `Aspect` code, see [42, 33].

2.3 Dedicated support for automatic differentiation

Automatic differentiation (AD) is often used to automatically derive residuals and their linearization from a stored energy functional, and to derive Jacobian matrices from residual vectors for simulations that use complicated material models. Examples of its application can be found widely within nonlinear solid mechanics, coupled multiphysics problems, as well as for nonlinear viscosity models in fluid flow.

`deal.II` has had a tutorial program (step-33) since 2007 that demonstrates this technique based on the Trilinos Sacado [19] package, but the functionality was not available pervasively throughout deal.II. This has changed with release 9.0 where support is given for differentiation using a selection of “white-listed” libraries (namely ADOL-C [32] and Sacado) and a subset of their supported number types. Currently, we offer support for the following cases:

– ADOL-C taped (n-differentiable),

– ADOL-C tapeless (once differentiable),

– Sacado dynamic forward (once differentiable),

– Sacado reverse (once differentiable),

– Sacado nested dynamic forward (twice differentiable), and

– Sacado nested reverse and dynamic forward (twice differentiable).

In practice, this support means that these ADOL-C and Sacado data types can be used in the `FEValues`, `FEValuesViews`, `Tensor`, `SymmetricTensor`, and related classes that are generally used to assemble linear systems and right hand sides. Given the updated capabilities of the library, there is now a dedicated module that presents the AD compatibility and capabilities of the `deal.II` libraries. Furthermore, the use of Sacado is now demonstrated in a much more simplified and transparent manner in an modernized version of an existing “code gallery” example [53].

To date it remains necessary for the user to manage the initialization of AD independent variables and the resultant calls to the AD dependent variables in order to initiate the computation of derivatives. In the next release we expect to provide a unified interface to these AD libraries, which will hide these library-dependent implementational details and facilitate switching between the supported libraries and AD number types based on the user’s requirements.

2.4 New interfaces to external libraries and programs

deal.II has always tried to leverage high-quality implementations of algorithms available through other open source software, rather than re-implementing their functionality. (A list of interfaces to other packages is given in Section 3.) As part of the current release, we have written several new interfaces as discussed in the following.

Assimp, the Open Asset Import Library. Assimp [57] can be used to read about 40 different 3D graphics formats. A subset of these formats can be now be read from within deal.II to generate two-dimensional meshes, possibly embedded in a three-dimensional space.

nanoflann, a library for building and querying k-d trees of datasets. Operations such as finding the vertex or cell closest to a given evaluation point occur frequently in many applications that use unstructured meshes. While the naive algorithm is linear in the number of vertices or cells, many such operations can be made significantly faster by building a k-d tree data structure that recursively subdivides a k dimensional space. The nanoflann library [21] provides such a data structure and allows querying it, either for closest points (e.g., when finding the closest vertex) or for searching the points that fall within a radius of a target point. This functionality is now available via deal.II interfaces.

ROL, a Rapid Optimization Library. ROL [56] is a package for large-scale optimization. deal.II can now use the state-of-the-art algorithms in ROL to solve unconstrained and constrained optimization problems as well as optimization problems under uncertainty. deal.II provides an interface to ROL's (abstract) vector class using the adapter software pattern. Through such an interface any vector class in deal.II following certain interface requirements can be used to define a ROL objective function.

ScaLAPACK, a parallel dense linear algebra library for distributed memory machines. ScaLAPACK [20] provides block-cyclic matrix distribution over 2D process grids. The functionality and interface of our wrappers is similar to the LAPACK [6] wrappers for serial dense linear algebra, namely matrix-matrix multiplication, Cholesky and LU factorizations, eigensolvers, SVD, least squares, pseudoinverses and save/load operations using either serial or parallel HDF5 [58]. All of this functionality is available even in cases where the number of MPI processes does not match the numbers of processes in the 2D process grid used to distribute a matrix.

As part of this effort, we have also improved LAPACK support: there are now methods to perform rank-1 updates/downdates, Cholesky factorizations, to compute the trace and determinant, as well as estimate the reciprocal condition number. We also now support configuration with 64-bit BLAS.

SUNDIALS, a SUite of Nonlinear and DIfferential/ALgebraic Equation Solvers. Solving nonlinear algebraic and differential equations is both a common task and one that often requires sophisticated globalization algorithms for efficiency and reliability. SUNDIALS [38] provides these in a widely used format, both sequentially and in parallel.

deal.II now has interfaces to SUNDIALS's ARKode, IDA, and KINSOL sub-packages. ARKode is a solver library that provides adaptive-step time integration. IDA is a package for the solution of differential-algebraic equations systems in the form $F(t, y, y') = 0$. KINSOL is solver for nonlinear algebraic systems.

2.5 Use of C++11

deal.II first offered support for a subset of C++11 features in version 6.2, released in 2009. The current release is the first to *require* a C++11 compiler.

Many parts of the code base have been rewritten to both support and use the new features of C++11. In particular, deal.II now makes extensive use of move semantics as well as range-based `for` loops with `auto` type deduction of iterator variables. We have also largely replaced `push_back()` by `emplace_back()` when adding elements to collections more efficiently.

Finally, we have changed the entire code base to avoid using raw pointers and instead use `std::unique_ptr` and `std::shared_ptr` where possible to make memory management more reliable. These changes include some minor incompatibilities: all `clone()` functions (such as `FiniteElement::clone()` and `Mapping::clone()`) now return `std::unique_ptr`s instead of C-style raw pointers. Indeed, nearly all interfaces throughout the library that return a pointer now return either a `std::shared_ptr` or a `std::unique_ptr`, thereby clarifying object ownership responsibilities and avoiding memory leaks.

2.6 Support for GPU computations

Heterogeneous computing is becoming more prevalent in supercomputing and this is a trend that is expected to continue in the future. In particular, the use of GPU has been increasing the last few years.

This release of `deal.II` adds support for GPU both for matrix-based and matrix-free applications. For matrix-based applications, we rely on cuSPARSE[2] and cuSOLVER[1] for operations on sparse matrices such as matrix-vector multiplication and for direct solvers. We have introduced a new type of sparse matrix, `CUDAWrappers::SparseMatrix`, which moves onto the device a `deal.II` `SparseMatrix` and changes the format of the underlying data to the appropriate CSR format used by cuSPARSE. We also have added wrappers for Cholesky and LU factorizations provided by cuSOLVER. In practice, a user would assemble the matrix associated to the system on the host and then move the matrix to the device. At this point, the system would be solved on the device and the solution would be moved back to the host.

We also have some support for matrix-free computation on GPU. For now, the evaluation of the operator is limited to mesh without hanging-nodes.

2.7 Extended matrix-free capabilities

The matrix-free infrastructure in `deal.II` was significantly overhauled for the current release. The major new contribution is the support of face integrals through a new class `FEFaceEvaluation`. The new class has a similar interface as the existing `FEEvaluation` class, and applies SIMD vectorization over several faces in analogy to the intra-cell vectorization in FEEvaluation. Discontinuous Galerkin operators are implemented defining two face functions, one for interior and one for boundary faces, in addition to the cell function. These kernels for the matrix-free operator evaluation are now collected in the new function `MatrixFree::loop`. The data structures have been particularly tuned for typical discontinuous Galerkin setups involving operators with first and second spatial derivatives. Both data access and computations have been thoroughly optimized and compared to the performance boundaries of the hardware. Furthermore, the support for AVX-512 instructions in the matrix-free framework was extended, adding new `gather` and `scatter` intrinsics for the indirect access to vector entries where appropriate.

To give an example of the algorithmic improvements, the computation of the values and gradients on all quadrature points for cell integrals has been significantly improved, yielding 10–20% better performance for cases where the kernels are compute bound. For the example of the reference cell gradient of a solution field $\mathbf{u}$ in three space dimensions, the new release applies the following change:

$$\text{previous: } \begin{bmatrix} D_1 \otimes S_2 \otimes S_3 \\ S_1 \otimes D_2 \otimes S_3 \\ S_1 \otimes S_2 \otimes D_3 \end{bmatrix} \mathbf{u} \quad \rightsquigarrow \quad \text{new: } \begin{bmatrix} D_1^{\text{co}} \otimes I_2 \otimes I_3 \\ I_1 \otimes D_2^{\text{co}} \otimes I_3 \\ I_1 \otimes I_2 \otimes D_3^{\text{co}} \end{bmatrix} \begin{bmatrix} S_1 \otimes S_2 \otimes S_3 \end{bmatrix} \mathbf{u}.$$

The matrices S_i contain the values of the one-dimensional shape functions in one-dimensional quadrature points and D_i their derivatives. When applied with the usual sum factorization implementation described, for example, in [43], the old kernels amounted to 9 partial summations – or rather 8 in the previous implementation of `deal.II` because the application of S_1 for the y and z components of the gradient can be merged. The new code performs a basis transformation to a related basis with derivative matrix $D_i = D_i^{\mathrm{co}} S_i$, which is the basis of Lagrange polynomials in the points of the quadrature. This change reduces the number of partial sums to only 6 for the gradient, as the action of the unit matrices I_i needs not be implemented. In isolation, this spectral element-like evaluation was previously available in `deal.II` for collocation between nodal points and quadrature, but not used for general bases. A more detailed description of the matrix-free modules is given in the preprint [44].

2.8 Tutorial and code gallery programs

`deal.II` gained several new tutorial programs:

- The step-59 tutorial program shows a matrix-free solver for the Poisson equation discretized with the symmetric interior penalty discontinuous Galerkin method.
- The step-60 tutorial program shows how to use the new `ParameterAcceptor` class as well as the new non-matching grid functionality.

`deal.II` has a separate "code gallery" that consists of programs shared by users as examples of what can be done with `deal.II`. While not part of the release process, it is nonetheless worth mentioning that the set of new programs since the last release covers the following topics:

- The multipoint flux mixed finite element method (MFMFE) applied to the Darcy problem of porous media flow;
- A linearized active skeletal muscle model with application to the simulation concentric contraction of the human biceps brachii;
- A parallel implementation of the Local Discontinuous Galerkin (LDG) method applied to the Poisson equation.

With these additions, the code gallery now contains 10 different applications.

2.9 Incompatible changes

The 9.0 release includes around 75 incompatible changes; see [46]. The majority of these changes should not be visible to typical user codes; some remove previously deprecated classes and functions, and the majority change internal interfaces that are not usually used in external applications. However, some are, such as changes to the interplay between meshes and manifolds, as well as the requirement to use a C++11 compiler (see Sections 2.1 and 2.5). In addition, the following incompatible changes are worth mentioning:

- The `BlockDiagonalMatrix`, `InverseMatrixRichardson`, `IterativeInverse`, `ProductMatrix`, `ProductSparseMatrix`, `TransposeMatrix`, `ScaledMatrix`, `SchurMatrix`, `ShiftedMatrix`, and `ShiftedMatrixGeneralized` classes have been removed. They are now generalized through the `LinearOperator` concept. Several other, similar classes have been deprecated.
- The default partitioner for the `parallel::shared::Triangulation` is now the Trilinos package Zoltan. This functionality was previously provided by the METIS partitioner, but the METIS package has not been actively maintained for a long time, and moreover yields subdivisions that depend on system details such as the random number generator and sorting facilities provided by the operating system; consequently, the partition is not consistent across platforms.

- The class `FE_DGQHermite` now uses a more stable, "Hermite-like" polynomial basis. The change is highly beneficial because it significantly improves the accuracy (in terms of round-off) for this basis and also reduces iteration counts for some iterative solvers with simple preconditioners.
- Many functions that previously returned a raw, C-style pointer now return a `std::unique_ptr` and `std::shared_ptr` where possible to make memory management more reliable.

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 it.

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 bibtex snippets for this document see:

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

The original `deal.II` paper containing an overview of its architecture is [12]. If you rely on specific features of the library, please consider citing any of the following:

- For geometric multigrid: [40, 39];
- For distributed parallel computing: [10];
- For *hp* adaptivity: [18];
- For partition-of-unity (PUM) and enrichment methods of the finite element space: [25];
- For matrix-free and fast assembly techniques: [43];
- For computations on lower-dimensional manifolds: [26];
- For integration with CAD files and tools: [34];
- For Boundary Elements Computations: [31];
- For `LinearOperator` and `PackagedOperation` facilities: [48, 49].
- For uses of the `WorkStream` interface: [59].

`deal.II` can interface with many other libraries:

- ADOL-C [32, 60]
- ARPACK [45]
- Assimp [57]
- BLAS and LAPACK [6]
- cuSOLVER [1]
- cuSPARSE [2]
- Gmsh [30]
- GSL [27]
- HDF5 [58]
- METIS [41]
- MUMPS [3, 4, 5, 50]
- muparser [51]
- nanoflann [21]
- NetCDF [55]
- OpenCASCADE [52]
- p4est [22]
- PETSc [8, 9]
- ROL [56]
- ScaLAPACK [20]
- SLEPc [35]
- SUNDIALS [38]
- TBB [54]
- Trilinos [36, 37]
- UMFPACK [24]

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

Older releases of `deal.II` can be cited as [14, 15, 16, 13, 11, 7].

4 Acknowledgments

`deal.II` is a world-wide project with dozens of contributors around the globe. Other than the authors of this paper, the following people contributed code to this release:
Julian Andrej, Rajat Arora, Lucas Campos, Praveen Chandrashekar, Jie Cheng, Emma Cinatl, Conrad Clevenger, Ester Comellas, Sambit Das, Giovanni Di Ilio, Nivesh Dommaraju, Marc Fehling, Niklas Fehn, Menno Fraters, Anian Fuchs, Daniel Garcia-Sanchez, Nicola Giuliani, Anne Glerum, Christoph Goering, Alexander Grayver, Samuel Imfeld, Daniel Jodlbauer, Guido Kanschat, Vishal Kenchan, Andreas Kergassner, Eldar Khattatov, Ingo Kligge, Uwe Köcher, Joachim Kopp, Ross Kynch, Konstantin Ladutenko, Tulio Ligneul, Karl Ljungkvist, Santiago Ospina, Alexey Ozeritsky, Dirk Peschka, Simon Puchert, E. G. Puckett, Lei Qiao, Ce Qin, Jonathan Robey, Alberto Sartori, Daniel Shapero, Ben Shields, Simon Sticko, Oliver Sutton, Zhuoran Wang, Xiaoyu Wei, Michał Wichrowski, Julius Witte, Feimi Yu, Weixiong Zheng.

Their contributions are much appreciated!

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

D. Arndt, K. Kormann and M. Kronbichler were partially supported by the German Research Foundation (DFG) under the project "High-order discontinuous Galerkin for the exa-scale" (ExaDG) within the priority program "Software for Exascale Computing" (SPPEXA).

W. Bangerth and R. Gassmöller were partially supported by the National Science Foundation under award OCI-1148116 as part of the Software Infrastructure for Sustained Innovation (SI2) program; and by the Computational Infrastructure in Geodynamics initiative (CIG), through the National Science Foundation under Awards No. EAR-0949446 and EAR-1550901 and The University of California – Davis.

V. Boddu was supported by the German Research Foundation (DFG) under the research group project FOR 1509.

B. Brands was partially supported by the Indo-German exchange programm "Multiscale Modeling, Simulation and Optimization for Energy, Advanced Materials and Manufacturing" (MMSO) funded by DAAD (Germany) and UGC (India).

D. Davydov was supported by the German Research Foundation (DFG), grant DA 1664/2-1.

T. Heister was partially supported by the Computational Infrastructure in Geodynamics initiative (CIG), through the National Science Foundation (NSF) under Award No. EAR-0949446 and EAR-1550901 and The University of California – Davis, and NSF Grant DMS-1522191.

M. Maier was partially supported by ARO MURI Award No. W911NF-14-0247.

J-P. Pelteret was supported by the European Research Council (ERC) through the Advanced Grant 289049 MOCOPOLY.

B. Turcksin: This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC05-00OR22725.

D. Wells was supported by the National Science Foundation (NSF) through Grant DMS-1344962.

The Interdisciplinary Center for Scientific Computing (IWR) at Heidelberg University has provided hosting services for the `deal.II` web page.

References

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

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

[3] P. Amestoy, I. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods in Appl. Mech. Eng.*, 184:501–520, 2000.

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

[5] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.

[6] 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.

[7] 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` library, version 8.5. *Journal of Numerical Mathematics*, 25(3):137–146, 2017.

[8] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. May, L. C. McInnes, R. Mills, T. Munson, K. Rupp, P. S. B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.9, Argonne National Laboratory, 2018.

[9] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. May, L. C. McInnes, R. Mills, T. Munson, K. Rupp, P. S. B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Web page. http://www.mcs.anl.gov/petsc, 2018.

[10] W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler. Algorithms and data structures for massively parallel generic adaptive finite element codes. *ACM Trans. Math. Softw.*, 38:14/1–28, 2011.

[11] W. Bangerth, D. Davydov, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and D. Wells. The `deal.II` library, version 8.4. *Journal of Numerical Mathematics*, 24(3):135–141, 2016.

[12] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II — a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4), 2007.

[13] W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, and B. Turcksin. The `deal.II` library, version 8.3. *Archive of Numerical Software*, 4(100):1–11, 2016.

[14] W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and T. D. Young. The `deal.II` library, version 8.0. *arXiv preprint http://arxiv.org/abs/1312.2266v3*, 2013.

[15] W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and T. D. Young. The `deal.II` library, version 8.1. *arXiv preprint http://arxiv.org/abs/1312.2266v4*, 2013.

[16] W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and T. D. Young. The `deal.II` library, version 8.2. *Archive of Numerical Software*, 3, 2015.

[17] W. Bangerth and G. Kanschat. Concepts for object-oriented finite element software – the `deal.II` library. Preprint 1999-43, SFB 359, Heidelberg, 1999.

[18] W. Bangerth and O. Kayser-Herold. Data structures and requirements for *hp* finite element software. *ACM Trans. Math. Softw.*, 36(1):4/1–4/31, 2009.

[19] R. A. Bartlett, D. M. Gay, and E. T. Phipps. Automatic differentiation of c++ codes for large-scale scientific computing. In V. Alexandrov, G. van Albada, and J. Sloot, P.M.A. amd Dongarra, editors, *International Conference on Computational Science – ICCS 2006*, pages 525–532. Springer Berlin Heidelberg, 2006.

[20] 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.

[21] J. L. Blanco and P. K. Rai. nanoflann: aC++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees. `https://github.com/jlblancoc/nanoflann`, 2014.

[22] 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.

[23] Coverity Scan (Synopsys, Inc.). `https://scan.coverity.com`.

[24] T. A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30:196–199, 2004.

[25] 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.

[26] 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.

[27] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, F. Rossi, and R. Ulerich. Gnu scientific library reference manual (edition 2.3), 2016.

[28] R. Gassmöller, E. Heien, E. G. Puckett, and W. Bangerth. Flexible and scalable particle-in-cell methods for massively parallel computations. Technical report, arXiv:1612.03369, 2016.

[29] 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. 2018, submitted.

[30] 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.

[31] 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(March):39–58, 2018.

[32] 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 (TOMS)*, 22(2):131–167, 1996.

[33] T. Heister, J. Dannberg, R. Gassmöller, and W. Bangerth. High accuracy mantle convection simulation through modern numerical methods. II: Realistic models and problems. *Geophysics Journal International*, 210:833–851, 2017.

[34] 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. Submitted.

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

[36] 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 Trans. Math. Softw.*, 31:397–423, 2005.

[37] M. A. Heroux et al. Trilinos web page, 2018. `http://trilinos.org`.

[38] 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 (TOMS)*, 31(3):363–396, 2005.

[39] 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.

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

[41] 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.

[42] M. Kronbichler, T. Heister, and W. Bangerth. High accuracy mantle convection simulation through modern numerical methods. *Geophysics Journal International*, 191:12–29, 2012.

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

[44] M. Kronbichler and K. Kormann. Fast matrix-free evaluation of discontinuous Galerkin finite element operators. Technical report, arXiv:1711.03590, 2017.

[45] 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.

[46] List of changes for 9.0. `https://www.dealii.org/developer/doxygen/deal.II/changes_between_8_5_and_9_0.html`.

[47] LLVM. Clang-Tidy. `http://clang.llvm.org/extra/clang-tidy/`.

[48] 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.

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

[50] MUMPS: a MUltifrontal Massively Parallel sparse direct Solver. `http://graal.ens-lyon.fr/MUMPS/`.

[51] muparser: Fast Math Parser Library. `http://muparser.beltoforion.de/`.

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

[53] J.-P. V. Pelteret and A. McBride. The `deal.II` code gallery: Quasi-static finite-strain compressible elasticity, 2016. Accessed April 2018. doi: `10.5281/zenodo.1228964`.

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

[55] R. Rew and G. Davis. NetCDF: an interface for scientific data access. *Computer Graphics and Applications, IEEE*, 10(4):76–82, 1990.

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

[57] T. Schulze, A. Gessler, K. Kulling, D. Nadlinger, J. Klein, M. Sibly, and M. Gubisch. Open asset import library (assimp). *Computer Software, URL: https://github. com/assimp/assimp*, 2012.

[58] The HDF Group. Hierarchical Data Format, version 5, 1997-2018. http://www.hdfgroup.org/HDF5/.

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

[60] A. Walther and A. Griewank. Getting started with ADOL-C. In *Combinatorial Scientific Computing*, Chapman-Hall CRC Computational Science, pages 181–202. U. Naumann and O.Schenk, 2012.