CASTIEL2 Code of the Month:
deal.II – A Generic Finite Element Library

Martin Kronbichler

Ruhr University Bochum, Germany

Collaborators in the dealii-X CoE and the deal.II community

With funding from the:

Federal Ministry of Research, Technology and Space

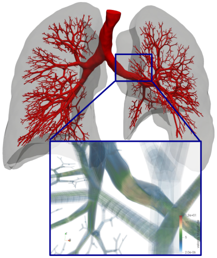
Funded by the European Union NextGenerationEU

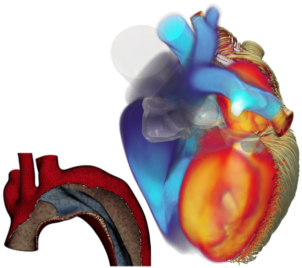
EuroHPC Joint Undertaking

July 23, 2025

dealii-X: an Exascale Framework for Digital Twins of the Human Body

- A scalable, high-performance computational platform
- Using the deal.II finite element library
- Create accurate digital twins of human organs
- Enable exascale computations of complex PDE models with deal.II





Project partners

- Ruhr University Bochum, DE
- Università di Pisa, IT
- Università degli Studi di Brescia, IT
- Leibniz Supercomputing Centre (LRZ), DE
- Scuola Internazionale Superiore degli Studi Avanzati, IT
- Università degli Studi di Roma Tor Vergata, IT
  - CNR
- Institut National Polytechnique de Toulouse, FR
  - Centre National de la Recherche Scientifique
- Technical University of Munich, DE
- Politecnico di Milano, IT
- Friedrich-Alexander University Erlangen-Nuremberg, DE
- Forschungsverbund Berlin eV, DE
- Exact Lab SRL, IT
  - Dualistic
- Virtual Physiological Human Institute, BE

Applications from our CoE dealii-X

The deal.II Finite Element Library

Matrix-free Operator Evaluation and Multigrid Solvers in deal.II

- Motivation
- Iterative Solvers

Advances in dealii-X CoE

Outline

Applications from our CoE dealii-X

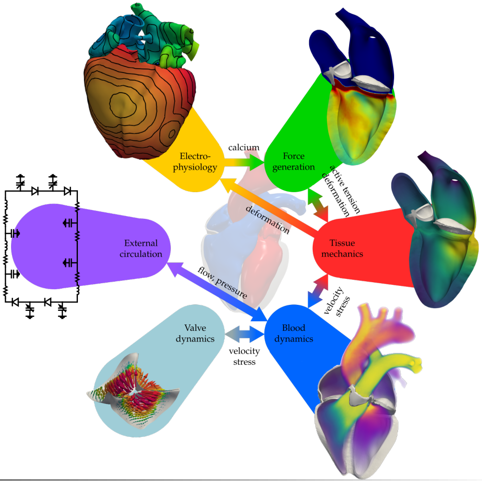
The deal.II Finite Element Library

Matrix-free Operator Evaluation and Multigrid Solvers in deal.II
- Motivation
- Iterative Solvers

Advances in dealii-X CoE

dealii-X Heart Modelling (Politecnico di Milano)

- **Goal:** an integrated, multiphysics simulator for the human heart, coupling several physical models describing the heart function
- This leads to large-scale, multiphysics, multiscale PDE systems
- Simulations use the in-house lifex library, based on deal.II (`https://lifex.gitlab.io`)
- We plan to leverage deal.II's matrix-free features to
  - improve the computational efficiency and scalability of standalone and multiphysics solvers
  - enhance support and efficiency for higher-order FEM, for improved accuracy
  - improve compatibility of heart simulators with HPC infrastructures

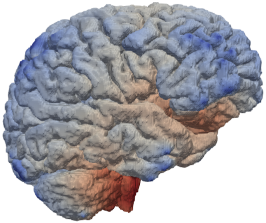


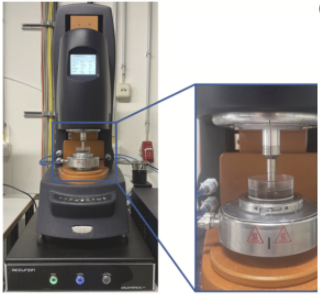
Poro-viscoelastic parameter identification of human brain tissue

by A. Greiner, S. Budday (FAU)

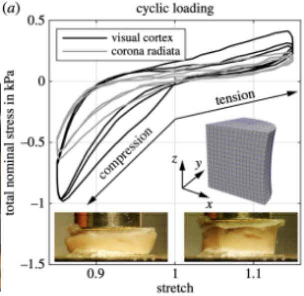
Highly complex biomechanical behavior requires region-specific poro-viscoelastic material parameters.

- Robust and efficient parameter identification algorithm.
- Exa-scale capabilities for full-brain simulations with high spatial and temporal resolution.

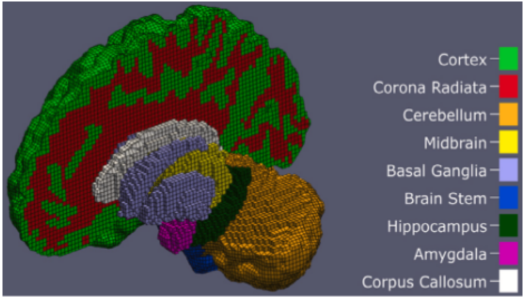




https://doi.org/10.1016/j.jmbbm.2024.106871



https://doi.org/10.1098/rsfs.2024.0026



https://doi.org/10.1016/j.euromechsol.2023.104910

Advanced modeling of the human liver

by A. Caiazzo, C. Belponer (WIAS Berlin), L. Heltai (Pisa)

Biomechanical properties of living tissues play an important role as biomarkers for diseases
Tissue imaging techniques (e.g., Elastography) are used for non-invasive estimation, based on medical imaging and computational models of tissue dynamics

Challenges

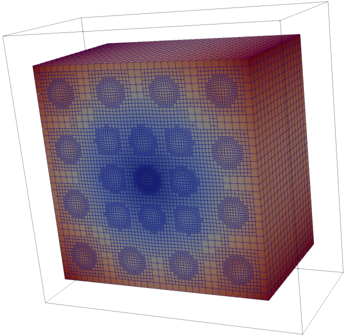
- Interplay of solid matrix and fluid vessels, on different scales
- Understanding the role of fluid properties is important for biomarkers related to hypertension – otherwise pressure can only be assessed invasively

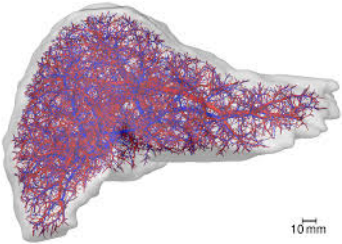
Objectives

- Efficient mathematical model of vascularized tissue, combining
  - immersed method (dimension reduction at the level of computational mesh)
  - reduced Lagrange multipliers (dimension reduction at the level of functional spaces)
  - multiscale modeling (dimension reduction at the level of the physical model, e.g., 3D-1D)

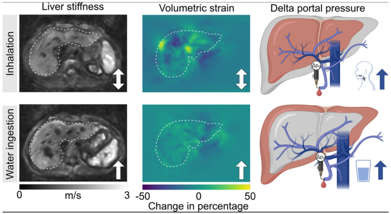
Liver: Subproject goals & steps (mathematical, computational, clinical)

- 3D-1D model of vascular tissue, coupled to one-dimensional hemodynamics
- Extension to realistic vasculature netwoks at different scales
- Application of the digital twin in inverse problems to characterize effective properties of liver tissues





Schwen et al. Representative Sinusoids for Hepatic Four-Scale Pharmacokinetics Simulations. PLOS One 10 (2015).



Jaitner et al. Noninvasive assessment of portal pressure by combined measurement of volumetric strain and stiffness ofÂ in vivoÂ human liver. Acta Biomater. 197 (2025)

Outline

Applications from our CoE dealii-X

The deal.II Finite Element Library

Matrix-free Operator Evaluation and Multigrid Solvers in deal.II

- Motivation
- Iterative Solvers

Advances in dealii-X CoE

What is deal.II?

- A C++ software library to ease the development of adaptive finite element codes on HPC systems
- Name origin: Differential Equations Analysis Library
- 2007 Wilkinson prize
- 2025 SIAM/ACM Prize in Computational Science and Engineering
- Homepage: `https://dealii.org`
- Code hosted on `https://github.com/dealii/dealii`
- Presently 464,000 lines of C++ code (`*.h, *.cc` files in core library), 9,000 lines of configuration code through `cmake` and template instantation files
- 70k lines of code and $>$ 100k lines of comments in tutorials, 17.6k tests in regression suite (controlled through 540,000 lines of code and 8 million lines of reference test output)
- 89 extensive tutorial programs

Arndt, Bangerth, Davydov, Heister, Heltai, Kronbichler, Maier, Pelteret, Turcksin and Wells: The deal.II finite element library: design, features, and insights. *Computers & Mathematics with Applications* 81:407–422, 2021 `doi:10.1016/j.camwa.2020.02.022`

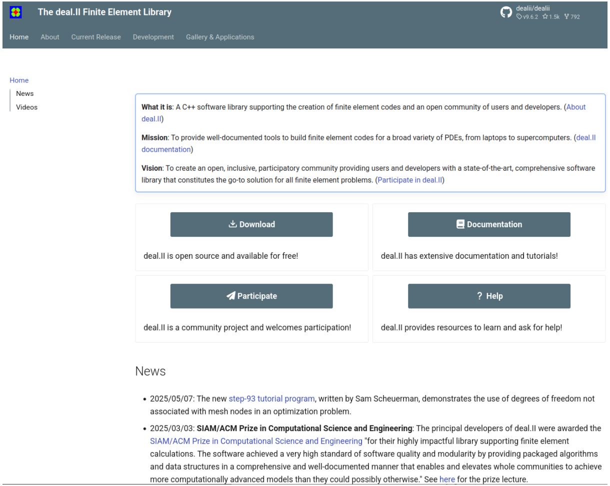
deal.II Community

- `deal.II` was started in 1997
  - Wolfgang Bangerth, Ralf Hartmann, Guido Kanschat @ University of Heidelberg
- Today 13 principal developers
  - Daniel Arndt (Oak Ridge), Wolfgang Bangerth (Colorado State), Bruno Blais (Montreal), Marc Fehling (Prague), Rene Gassmoeller (Kiel), Timo Heister (Clemson), Luca Heltai (Pisa), Martin Kronbichler (Bochum), Matthias Maier (Texas A&M), Peter Munch (Berlin), Jean-Paul Pelteret, Bruno Turcksin (Oak Ridge NL), David Wells (Chapel Hill/North Carolina)
- Contributions from 418 individuals
  - around 90 active developers last year
- Code contributions organized through github

Arndt, Bangerth, Davydov, Heister, Heltai, Kronbichler, Maier, Pelteret, Turcksin and Wells: The deal.II finite element library: design, features, and insights. *Computers & Mathematics with Applications* 81:407–422, 2021
`doi:10.1016/j.camwa.2020.02.022`

The deal.II homepage

The deal.II web page `www.dealii.org`



The deal.II Finite Element Library

dealii/dealii v9.6.2 1.5k 792

Home About Current Release Development Gallery & Applications

Home
News
Videos

What it is: A C++ software library supporting the creation of finite element codes and an open community of users and developers. (About deal.II)

Mission: To provide well-documented tools to build finite element codes for a broad variety of PDEs, from laptops to supercomputers. (deal.II documentation)

Vision: To create an open, inclusive, participatory community providing users and developers with a state-of-the-art, comprehensive software library that constitutes the go-to solution for all finite element problems. (Participate in deal.II)

Download

deal.II is open source and available for free!

Documentation

deal.II has extensive documentation and tutorials!

Participate

deal.II is a community project and welcomes participation!

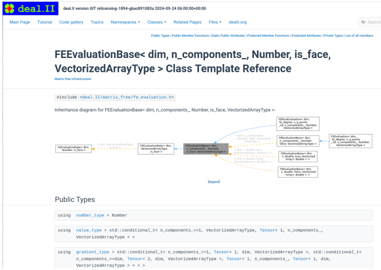
? Help

deal.II provides resources to learn and ask for help!

News

- 2025/05/07: The new step-93 tutorial program, written by Sam Scheuerman, demonstrates the use of degrees of freedom not associated with mesh nodes in an optimization problem.
- 2025/03/03: **SIAM/ACM Prize in Computational Science and Engineering**: The principal developers of deal.II were awarded the SIAM/ACM Prize in Computational Science and Engineering "for their highly impactful library supporting finite element calculations. The software achieved a very high standard of software quality and modularity by providing packaged algorithms and data structures in a comprehensive and well-documented manner that enables and elevates whole communities to achieve more computationally advanced models than they could possibly otherwise." See here for the prize lecture.

deal.II documentation



deal.II version GIT relicensing-1894-gbac891083a 2024-09-24 06:00:00+00:00

Main Page Tutorial Code gallery Topics Namespaces Classes Related Pages Files dealii.org Search

Public Types | Public Member Functions | Static Public Attributes | Protected Member Functions | Protected Attributes | Private Types | List of all members

FEEvaluationBase< dim, n_components_, Number, is_face, VectorizedArrayType > Class Template Reference

Matrix-free infrastructure

```
#include <deal.II/matrix_free/fe_evaluation.h>
```

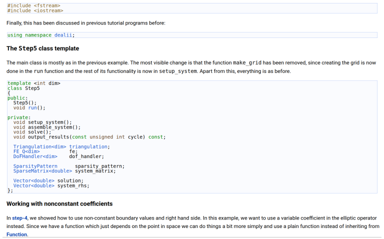
Inheritance diagram for FEEvaluationBase< dim, n_components_, Number, is_face, VectorizedArrayType >:

[legend]

Public Types

```
using number_type = Number
using value_type = std::conditional_t< n_components_==1, VectorizedArrayType, Tensor< 1, n_components_, VectorizedArrayType > >
using gradient_type = std::conditional_t< n_components_==1, Tensor< 1, dim, VectorizedArrayType >, std::conditional_t< n_components_==dim, Tensor< 2, dim, VectorizedArrayType >, Tensor< 1, n_components_, Tensor< 1, dim, VectorizedArrayType > > > >
```

Extensive tutorial programs



```
#include <fstream>
#include <iostream>
```

Finally, this has been discussed in previous tutorial programs before:

```
using namespace dealii;
```

The `Step5` class template

The main class is mostly as in the previous example. The most visible change is that the function `make_grid` has been removed, since creating the grid is now done in the `run` function and the rest of its functionality is now in `setup_system`. Apart from this, everything is as before.

```
template <int dim>
class Step5
{
public:
  Step5();
  void run();

private:
  void setup_system();
  void assemble_system();
  void solve();
  void output_results(const unsigned int cycle) const;

  Triangulation<dim> triangulation;
  FE_Q<dim>          fe;
  DoFHandler<dim>    dof_handler;

  SparsityPattern      sparsity_pattern;
  SparseMatrix<double> system_matrix;

  Vector<double> solution;
  Vector<double> system_rhs;
};
```

Working with nonconstant coefficients

In step-4, we showed how to use non-constant boundary values and right hand side. In this example, we want to use a variable coefficient in the elliptic operator instead. Since we have a function which just depends on the point in space we can do things a bit more simply and use a plain function instead of inheriting from Function.

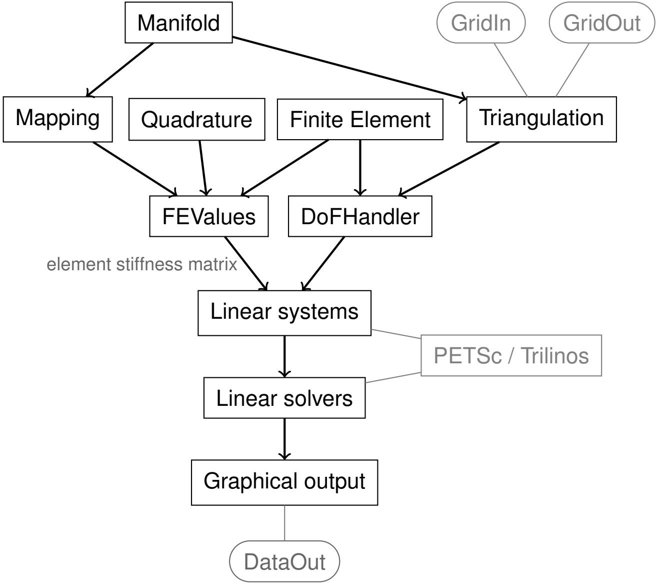
Concept of deal.II

- deal.II is a **library, not a solver**
  - Does not implement any specific equation by itself
  - Instead, provide the building blocks to easily build a solver
  - Applicable to almost any partial differential equation: Make it easy to state weak form in finite element problem
- Functionality for mathematical ingredients in a finite element problem
- Program in C++, with suitable abstractions
- Write basic prototypes in 100–200 lines of code
- Or start from one of the tutorials of deal.II

Main features of the deal.II library

- Lots of finite elements available
  - Continuous, discontinuous, H^{curl} and H^{div} conforming ones
  - Scalar or vector-valued problems, arbitrary combinations, *hp* adaptivity
  - Best support for line/quadrilateral/hexahedral elements, initial support for simplices and mixed meshes
- Pre- and post-processing
  - Can read most formats
  - Can write almost any visualization file format, VTK/VTU preferred
- Linear algebra in deal.II
  - Has its own sub-library for dense and sparse linear algebra
  - Interfaces to LAPACK, PETSc (including hypre and SLEPc), Trilinos, UMFPACK, SUNDIALS, etc.
- Parallelization
  - Uses threads and tasks on multicore machines
  - MPI parallelism used for production runs on 300,000 processors
  - Mesh distributed with METIS (small/medium scale) and p4est (fully distributed adaptive mesh storage)

How to solve a problem with deal.II

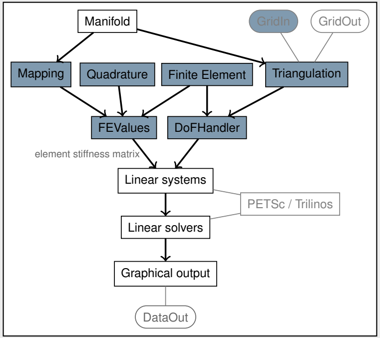


Features of FEM library

- Parallelization
- Mesh description
- Finite elements
- Quadrature rules
- Mapping to describe deformed elements
- Assembly of local matrices & vectors
- Linear solver

Example: Poisson problem

```cpp
const unsigned int dim = 2, degree = 3;

parallel::distributed::Triangulation<dim> tria(MPI_COMM_WORLD);
GridIn<dim> grid_in; grid_in.attach_triangulation(tria);
grid_in.read_ucd("my-grid.inp");
FE_Q<dim>       fe(degree);
QGauss<dim>     quad(degree + 1);
MappingQ<dim> mapping(1);

DoFHandler<dim> dof_handler(tria);
dof_handler.distribute_dofs(fe);

// deal with boundary conditions
AffineConstraints<double> constraints;
DoFTools::make_zero_boundary_constraints(dof_handler, constraints);
constraints.close();

// initialize vectors and system matrix
LinearAlgebra::distributed::Vector<double> x, b;
TrilinosWrappers::SparseMatrix             A;
util::initialize_dof_vector(dof_handler, x); util::initialize_dof_vector(dof_handler, b);
util::initialize_system_matrix(dof_handler, constraints, A);
```




```cpp
// assemble right-hand side and system matrix
FEValues<dim> fe_values(mapping, fe, quad, update_values | update_gradients | update_JxW_values);
FullMatrix<double>                   cell_matrix;
Vector<double>                       cell_rhs;
std::vector<types::global_dof_index> local_dof_indices;
```

Example: Assembly of Poisson problem

```
for (const auto &cell : dof_handler.active_cell_iterators()) // loop over all locally-owned cells
  {
    if (cell->is_locally_owned() == false) continue;
    fe_values.reinit(cell);

    const auto dofs_per_cell = cell->get_fe().n_dofs_per_cell(); // allocate memory for element matrix/vector
    cell_matrix.reinit(dofs_per_cell, dofs_per_cell);             //
    cell_rhs.reinit(dofs_per_cell);                               //


    for (const auto q : fe_values.quadrature_point_indices())   // compute element matrix/vector
      {                                                          //
        for (const auto i : fe_values.dof_indices())            //
          for (const auto j : fe_values.dof_indices())          //
            cell_matrix(i, j) += (fe_values.shape_grad(i, q) * //
                                  fe_values.shape_grad(j, q) * //
                                  fe_values.JxW(q));            //
                                                                 //
        for (const unsigned int i : fe_values.dof_indices())    //
          cell_rhs(i) += (fe_values.shape_value(i, q) *         //
                          1. *                                   //
                          fe_values.JxW(q));                     //
      }

    local_dof_indices.resize(cell->get_fe().dofs_per_cell);             // assembly
    cell->get_dof_indices(local_dof_indices);                           //
    constraints.distribute_local_to_global(cell_matrix, cell_rhs, A, b); //
  }

b.compress(VectorOperation::values::add); A.compress(VectorOperation::values::add);
```

$$(\nabla N, \nabla N)_{\Omega_e} \to \mathbf{K}^{(e)}$$

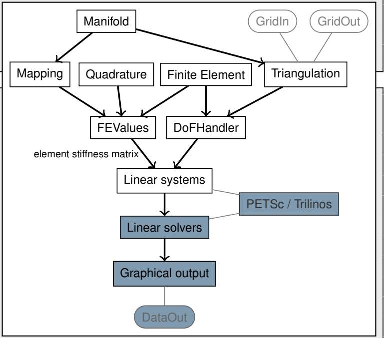
$$(N, f)_{\Omega_e} \to \mathbf{f}^{(e)}$$

$$A = \sum_e \mathbf{K}^{(e)}, b = \sum_e \mathbf{f}^{(e)}$$

Example: Solving system and post-processing

```
// solve linear equation system
ReductionControl                              reduction_control;
SolverCG<LinearAlgebra::distributed::Vector<double>> solver(reduction_control);
solver.solve(A, x, b, PreconditionIdentity());

if (Utilities::MPI::this_mpi_process(util::get_mpi_comm(tria)) == 0)
  printf("Solved in %d iterations.\n", reduction_control.last_step());

constraints.distribute(x);
```

```
// output results
DataOutBase::VtkFlags flags;
flags.write_higher_order_cells = true;

DataOut<dim> data_out;
data_out.set_flags(flags);
data_out.attach_dof_handler(dof_handler);
data_out.add_data_vector(dof_handler, x, "solution");
data_out.build_patches(mapping, degree + 1);

data_out.write_vtu_with_pvtu_record("./", "result", 0,
                                    MPI_COMM_WORLD);
```



Summary of coding interface

Compare deal.II code with other mathematical concepts

- Code generation (+ configuration) *. . . UFL in FEniCS, Firedrake, DUNE*

```
a = dot(grad(v), grad(u)) * dx
```

- Provide facilities to write C++ code directly easier *. . . deal.II style*

```
// Matrix-based code in deal.II
for (const auto q : fe_values.quadrature_point_indices())
  for (const auto i : fe_values.dof_indices())
    for (const auto j : fe_values.dof_indices())
      cell_matrix(i, j) += fe_values.shape_grad(i, q) * fe_values.shape_grad(j, q)
                           * fe_values.JxW(q);
```

```
// Matrix-free code in deal.II
phi.reinit(cell); phi.gather_evaluate(src, EvaluationFlags::gradients);
for (const auto q : phi.quadrature_point_indices())
  phi.submit_gradient(phi.get_gradient(q), q);
phi.integrate_scatter(EvaluationFlags::gradients, dst);
```

. . . . . . by providing easy-to-use helper classes

Working with deal.II: library approach

Code development in our CoE applications

- Implementation of physical equations
  - Based on appropriate data structures from deal.II library
- Simulation control based on library components
  - Iterative solvers, multigrid suited for equations at hand
  - Embeds equation evaluator into existing interfaces

Code development within deal.II finite element library

- Building blocks targeting particular hardware (SIMD, CUDA, SYCL)
- MPI parallelization of mesh
- Interface to linear algebra libraries like Trilinos/PETSc
- Move common functionality to deal.II library
  - Separation of concerns
  - Forces split of application-specific vs generic FEM toolbox
  - Reuse among many applications with more resources

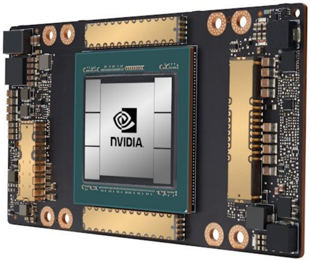


Image source: Nvidia Ampere Whitepaper

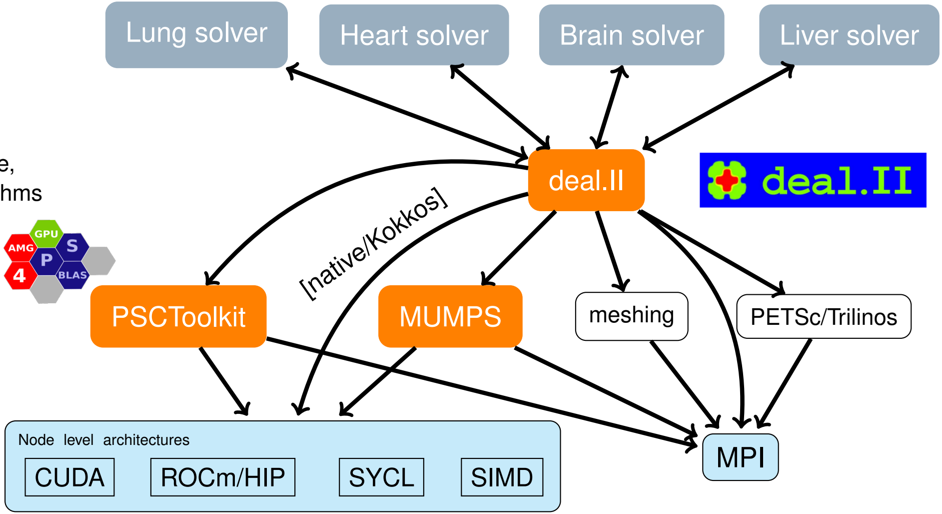
dealii-X: Composition of software libraries

Applications

FEM infrastructure,
matrix-free algorithms

Linear algebra,
preconditioning,
partitioning

Back-ends,
hardware



Implement in deal.II or use external libraries?

External libraries

- Matrix-based linear algebra and preconditioners: PETSc, Trilinos, PSCToolkit, MUMPS, several more direct/sparse solvers
- Mesh partitioning: p4est, METIS
- Time stepping and nonlinear solvers: SUNDIALS, PETSc-TS, ...

Programming frameworks & backends

- Distributed memory: MPI
- Threading
- Kokkos (and some CUDA) for GPUs
- Vectorization: via compiler (intrinsics) or C++ standard library

Implemented inside deal.II

- Algorithms for finite elements and
- Linear algebra infrastructure not available with adequate functionality or performance externally
- MPI-parallel vector `LinearAlgebra::distributed::Vector<Number>`
- SIMD abstraction class `VectorizedArray<Number>`
- Wrapper classes for unified interface to external libraries
- Help our users to concentrate on their application: separation of concerns

Outline

Applications from our CoE dealii-X

The deal.II Finite Element Library

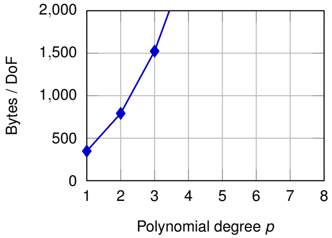
Matrix-free Operator Evaluation and Multigrid Solvers in deal.II

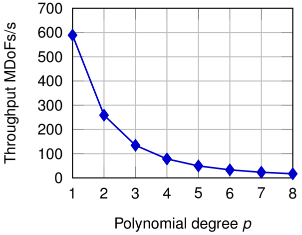
- Motivation
- Iterative Solvers

Advances in dealii-X CoE

Performance analysis

- Iterative solvers spend 60–95% of time in matrix-vector product or related substitutions (Gauss–Seidel, ILU)
- **Sparse matrix-vector product** with 0.16–0.25 Flop/Byte **memory bandwidth limited**
  - Bad already on 2005's hardware
  - Terrible in 2025 and in foreseeable future
  - Modern CPUs/GPUs provide 3–20 Flop/Byte
- Especially critical for higher order methods
  - Matrix becomes more densely populated
  - Degree 4 $10\times$ as expensive per unknown as degree 1
- Limited possibility for tuning (= reduce memory transfer)
  - Very hard to gain more than 30%
- Only way out: **matrix-free** algorithm with **action** of matrix entries on vector on the fly
  - Stencil-based (low order, meshes with structure)
  - Fast computation of FEM integrals $\to$ my choice





Sparse matrix-vector product on 1 node of SuperMUC-NG

Matrix-free operator evaluation by integration

Matrix-vector product

matrix-based:

$$\begin{cases} \boldsymbol{A} = \sum_{e=1}^{N_{\mathrm{el}}} \boldsymbol{P}_e^{\mathrm{T}} \boldsymbol{A}_e \boldsymbol{P}_e & \text{(assembly)} \\ \boldsymbol{v} = \boldsymbol{A}\boldsymbol{u} & \text{(matrix-vector product within iterative solver)} \end{cases}$$

matrix-free:

$$\boldsymbol{v} = \sum_{e=1}^{N_{\mathrm{el}}} \boldsymbol{P}_e^{\mathrm{T}} \boldsymbol{A}_e (\boldsymbol{P}_e \boldsymbol{u})$$

implication: assembly facilities within iterative solvers

Matrix-free evaluation of FEM Laplacian

- $\boldsymbol{v} = \mathbf{0}$
- loop over elements $e = 1, \ldots, N_{\mathrm{el}}$
  - (i) Extract local vector values: $\boldsymbol{u}_e = \boldsymbol{P}_e \boldsymbol{u}$
  - (ii) Apply operation on element by integration: $\boldsymbol{v}_e = \boldsymbol{A}_e \boldsymbol{u}_e$ (**without** forming $\boldsymbol{A}_e$)
  - (iii) Sum results from (ii) into the global solution vector: $\boldsymbol{v} = \boldsymbol{v} + \boldsymbol{P}_e^{\mathrm{T}} \boldsymbol{v}_e$

Design goals of implementation in deal.II:

- Data locality for higher arithmetic intensity: single sweep through data
- Absolute performance in **unknowns per second** (DoFs/s), not maximal GFlop/s or GB/s

M. Kronbichler, K. Kormann: A generic interface for parallel finite element operator application. *Comput. Fluids* 63, 2012

M. Kronbichler, K. Kormann: Fast matrix-free evaluation of discontinuous Galerkin finite element operators. *ACM Trans Math Softw* 45(3), 29, 2019

Matrix-vector product on cell by integration

Contribution of element Ω_e to matrix-vector product for finite element Laplacian

$$(\boldsymbol{A}_e \boldsymbol{u}_e)_i = \int_{\Omega_e} \nabla_{\boldsymbol{x}} \phi_i \cdot \nabla_{\boldsymbol{x}} u^h d\boldsymbol{x} \approx \sum_q w_q \det J_q \nabla_{\boldsymbol{x}} \phi_i \cdot \nabla_{\boldsymbol{x}} u^h \Big|_{\boldsymbol{x}=\boldsymbol{x}_q}$$

$$= \sum_q \nabla_\xi \phi_i \boldsymbol{J}_q^{-1} (w_q \det \boldsymbol{J}_q) \boldsymbol{J}_q^{-\mathrm{T}} \sum_j \nabla_\xi \phi_j u_{e,j} \Big|_{\boldsymbol{x}=\boldsymbol{x}_q}, \quad i = 1, \ldots, \text{cell\_dofs}$$

(a) Compute unit cell gradients $\nabla_\xi u^h = \sum_j (\nabla_\xi \phi_j) u_{e,j}$ at all quadrature points

(b) At each quadrature point, apply geometry $\boldsymbol{J}_q^{-\mathrm{T}}$, multiply by quadrature weight and Jacobian determinant, apply geometry for test function $\boldsymbol{J}_q^{-1}$

(c) Test by unit cell gradients of all basis functions and sum over quadrature points

Matrix notation:

$$\boldsymbol{v}_e = \boldsymbol{A}_e \boldsymbol{u}_e$$
$$= \boldsymbol{S}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{S} \boldsymbol{u}_e$$

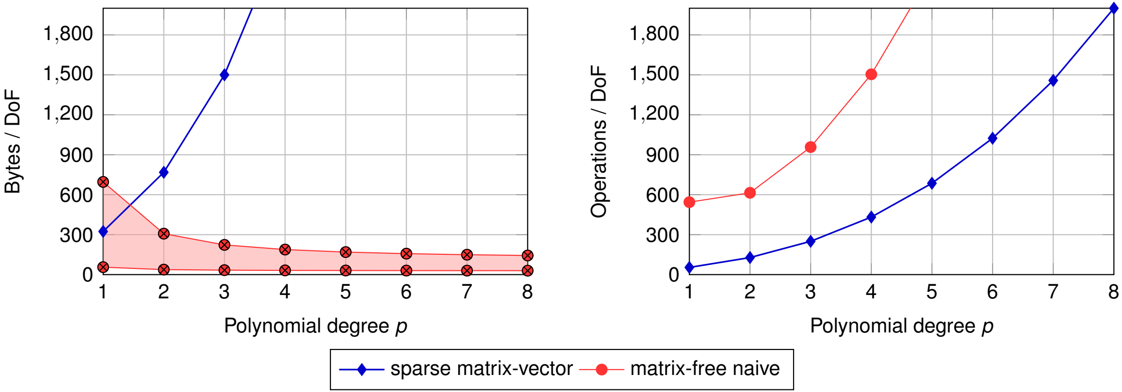
with

$\boldsymbol{S}_{qi} = \nabla_\xi \phi_i \big|_{\xi_q}$

$\boldsymbol{W}_{qq} = \boldsymbol{J}_q^{-1} (w_q \det \boldsymbol{J}_q) \boldsymbol{J}_q^{-\mathrm{T}}$

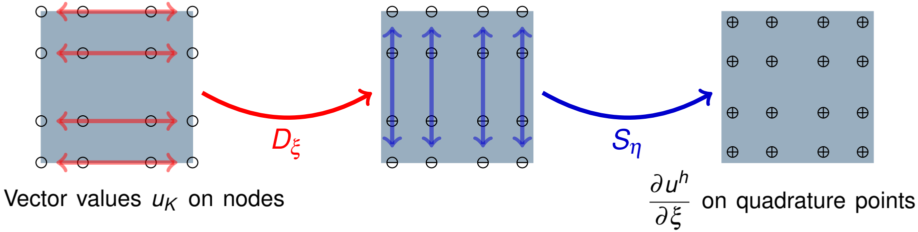
Matrix-free method with naive interpolation

- Matrix-free method trades memory transfer for additional computation
- Naive approach: Use dense interpolation/derivative matrices ***S***; same matrix on each element $\rightarrow$ data re-use by caches
- Not competitive in 2009, but has become surprisingly good in difficult cases[1]



[1] Sun, Mitchell, Kulkarni, Klöckner, Ham, Kelly: A study of vectorization for matrix-free finite element methods, *Int J High Perf Comput Appl* 34(6), 2020

- Efficient evaluation of S and S^T matrices with structure $S_{3\mathrm{D}} = \begin{bmatrix} S_\zeta \otimes S_\eta \otimes D_\xi \\ S_\zeta \otimes D_\eta \otimes S_\xi \\ D_\zeta \otimes S_\eta \otimes S_\xi \end{bmatrix}$
- Ideas from spectral elements (1980s) for tensor product shape functions and tensor product quadrature
- Visualization of interpolation of $\frac{\partial u}{\partial \xi}$ with $\mathcal{Q}_3$ element (Lagrange basis) $S_\eta \otimes D_\xi$: successively apply 1D kernels

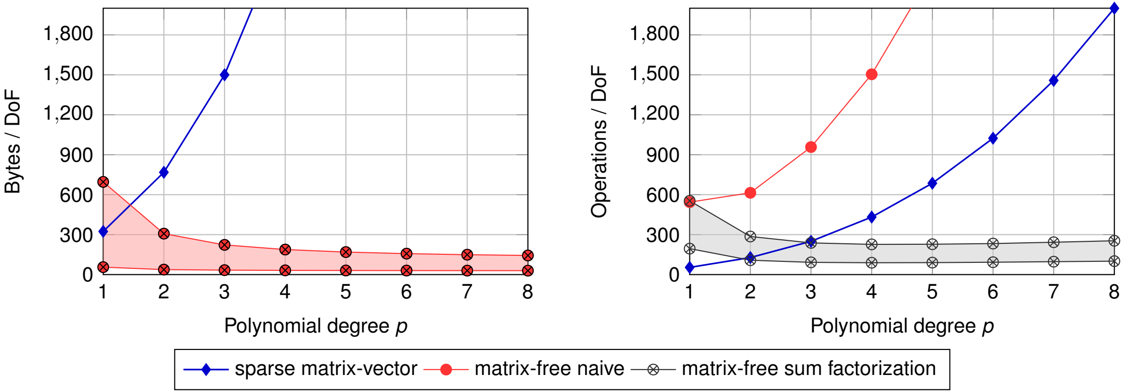


Tensor-based evaluation reduces evaluation cost from 4^4 to 2×4^3

In general for degree p and dimension d: $\mathcal{O}((p+1)^{2d})$ to $\mathcal{O}(d(p+1)^{d+1})$

Matrix-free methods with sum factorization

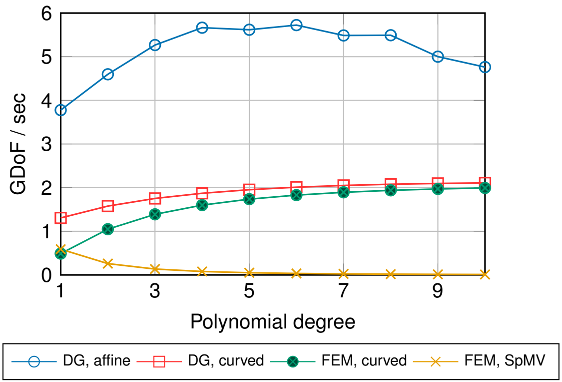
- Matrix-free method trades memory transfer for additional computation
- Naive approach: Use dense interpolation/derivative matrices ***S***
- Sum factorization: Lower complexity for higher degrees, only 1D interpolation matrices (matrix-matrix multiplication)



M. Kronbichler, K. Kormann: A generic interface for parallel finite element operator application. *Comput. Fluids* 63, 2012

- Performance of matrix-vector product essential for iterative solvers
- Sparse matrices unsuitable for higher orders $p \geq 2$ on modern hardware due to **memory-bandwidth** limit
- Matrix-free algorithm successful in trading computations for less memory transfer
  - Software: Specify operation at quadrature points
  - Combine with reference cell interpolation matrices
  - Indirect access into vector entries for continuous FEM

Throughput of matrix-vector product (billion unknowns processed per second) of 3D Laplacian



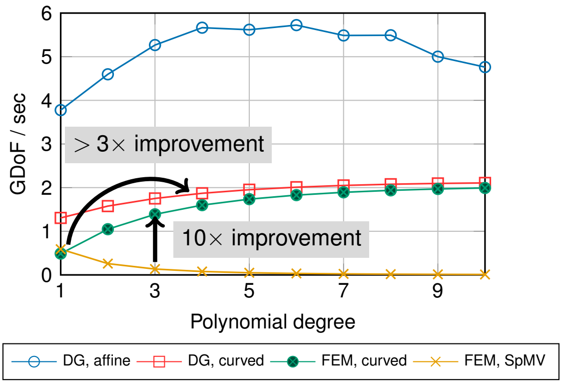
System: 1 node of 2×24 cores of Intel Xeon Platinum 8174 (Skylake)
Memory bw: 205 GB/s, arithmetic peak 3.5 TFlop/s

Kronbichler, Kormann: A generic interface for parallel cell-based finite element operator application. *Comput Fluids* 63:135–147, 2012
Kronbichler, Wall: A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers. *SISC* 40(5):A3423–48, 2018
Kronbichler, Kormann: Fast matrix-free evaluation of discontinuous Galerkin finite element operators. *ACM Trans Math Softw* 45(3):29/1–40, 2019

Matrix-free vs. matrix-based methods

- Performance of matrix-vector product essential for iterative solvers
- Sparse matrices unsuitable for higher orders $p \geq 2$ on modern hardware due to **memory-bandwidth** limit
- Matrix-free algorithm successful in trading computations for less memory transfer
  - Software: Specify operation at quadrature points
  - Combine with reference cell interpolation matrices
  - Indirect access into vector entries for continuous FEM

Throughput of matrix-vector product (billion unknowns processed per second) of 3D Laplacian

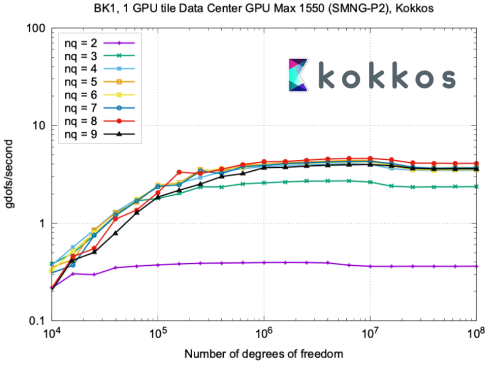


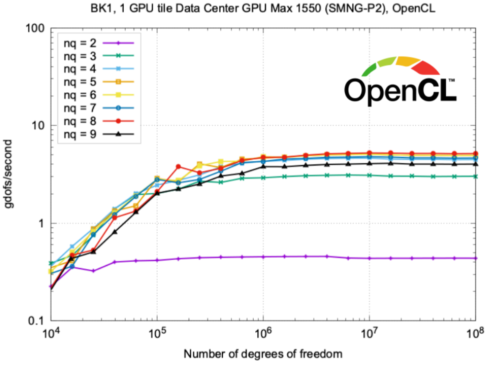
System: 1 node of 2×24 cores of Intel Xeon Platinum 8174 (Skylake)
Memory bw: 205 GB/s, arithmetic peak 3.5 TFlop/s

Kronbichler, Kormann: A generic interface for parallel cell-based finite element operator application. *Comput Fluids* 63:135–147, 2012
Kronbichler, Wall: A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers. *SISC* 40(5):A3423–48, 2018
Kronbichler, Kormann: Fast matrix-free evaluation of discontinuous Galerkin finite element operators. *ACM Trans Math Softw* 45(3):29/1–40, 2019

by E. Soydan (RUB), I. Pribec (LRZ)

- Find optimized kernels for all GPU vendors; CUDA + Kokkos kernels made good initial progress
- New results on Intel PVC (Intel Data Center GPU Max 1550, SMNG-P2)





Matrix-free methods within multigrid solvers

- Matrix-free implementation essential to reach good throughput for higher orders
- But **no matrix entries available** – options for multigrid solvers?

Low performance / impossible

- Substitutions on matrix entries such as (S)SOR, ILU $\rightarrow$ little benefit from using these smoothers
- Algebraic multigrid hierarchy generation
- AMG applied to high-order space leads to high iteration counts

High performance

- Polynomial smoothers (point-Jacobi within Chebyshev iteration)
- Overlapping Schwarz smoothers[2] [3]
- Multigrid transfer operations
- Algebraic multigrid on low-order refined discretization

- Select point-Jacobi with Chebyshev acceleration despite somewhat higher iteration counts than Schwarz methods due to better arithmetic optimizations

[2] Fischer, Lottes, Hybrid Schwarz-Multigrid Methods for the Spectral Element Method, 2004

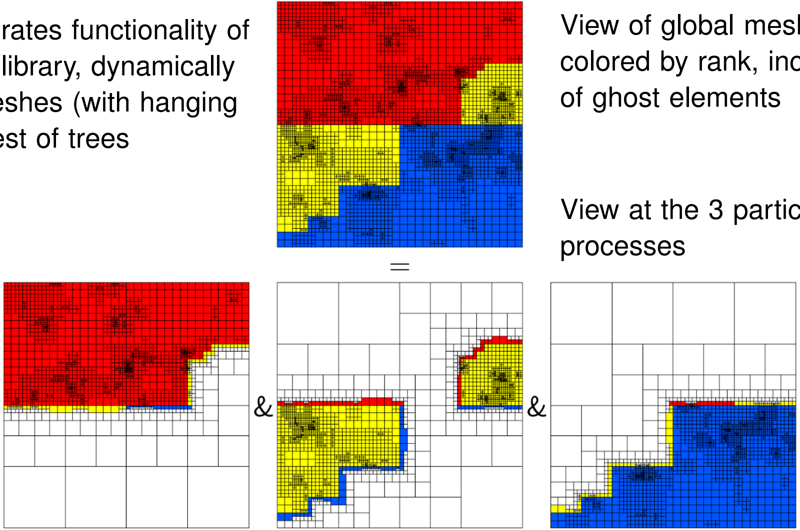
[3] Munch, Kronbichler, Cache-optimized and low-overhead implementations of additive Schwarz methods for high-order FEM multigrid computations, *Int. J. High Perf. Comput. Appl.* 38, 2024

Parallel meshes in deal.II: Schematic view

deal.II integrates functionality of the `p4est` library, dynamically adapted meshes (with hanging nodes), forest of trees

View of global mesh partitions, colored by rank, including 1 layer of ghost elements

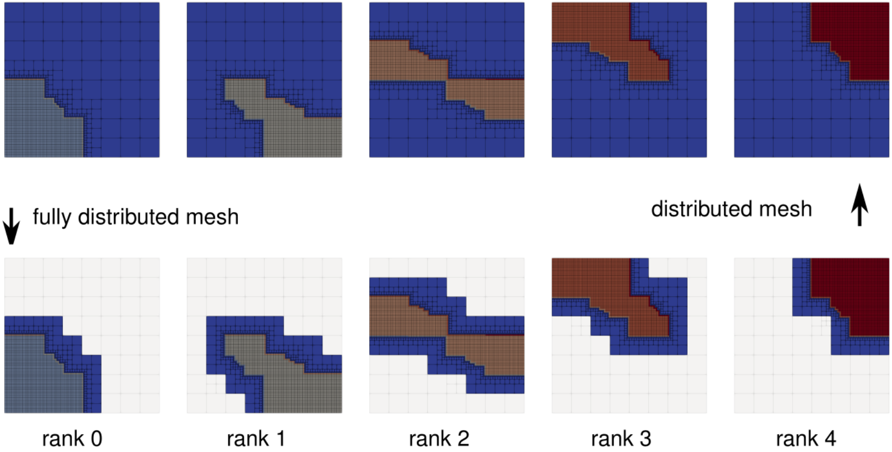
View at the 3 participating MPI processes



4 `www.p4est.org`

5 W. Bangerth, C. Burstedde, T. Heister, M. Kronbichler, *ACM TOMS* 38(2), 2011

Besides the "distributed" mesh via `p4est`, deal.II also supports externally partitioned meshes without all coarse cells ("fully distributed" mesh)

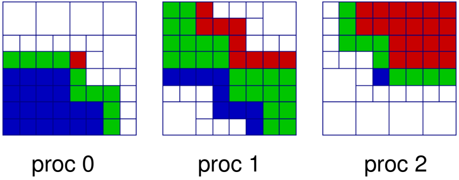


Joint work with P. Munch

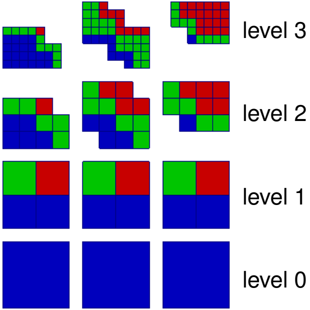
Parallel multigrid algorithms in the deal.II library

- Finite element library `deal.II`
- Each processor has its view of mesh (refined from coarse mesh)
- `deal.II` implements massively parallel geometric multigrid in 2D & 3D on both continuous and discontinuous elements[6]

View of active cells



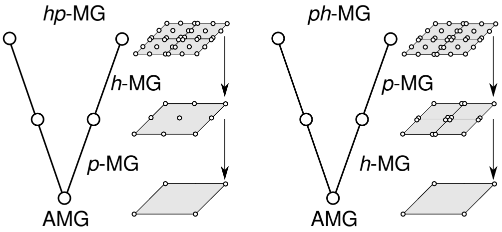
View of multigrid levels

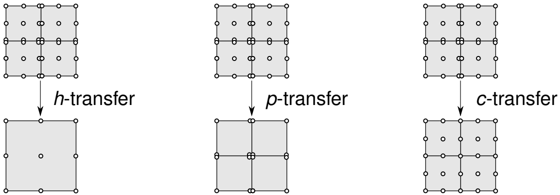


[6] C. Clevenger, T. Heister, G. Kanschat, M. Kronbichler, A flexible, parallel, adaptive geometric multigrid method for FEM, *ACM TOMS*, 47(1), 2020

Multigrid: Work on hierarchy of coarser representations

- Problem: Large coarse mesh + some adaptive refinements
- Question: How to create coarser problems needed for multigrid?
- Classical approach 1: reduce polynomial degree down to linear methods (*p* multigrid)
- Hybrid multigrid: after *p* coarsening, coarsen also mesh (*ph* MG)
- Innovation[7]: transfer DG $\rightarrow$ continuous elements as first step in hierarchy (*cph* coarsening)
- Combined with AMG on coarse level
- Multigrid V-cycle in **single precision**





[7] Fehn, Munch, Wall, Kronbichler, Hybrid multigrid methods for high-order discontinuous Galerkin discretizations, *J Comput Phys* 415:109538, 2020

Best multigrid coarsening strategy by # iterations

Solve Poisson equation on deformed cube, 8^3 elements, tolerance 10^{-10}, Chebyshev(5,5) with point-Jacobi smoother, coarse solver AMG V-cycle, report fractional iteration counts

- p multigrid with coarsening $k_{l-1} = \lfloor k_l/2 \rfloor$

MG type	Polynomial degree			
	1	3	5	9
h	3.4	15.3	18.5	25.1
ph	17.7	16.4	18.6	25.1
phc	17.7	16.4	14.1	25.1
cp	8.5	5.8	6.5	9.5
cph	8.5	5.9	6.5	9.5

- Robustness with respect to penalty parameter τ in SIPG
- Polynomial degree $k = 4$

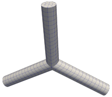
IP factor	*hp* MG	*ph* MG	*cph* MG
$10^0(k+1)^2/h$	12.2	12.0	4.9
$10^1(k+1)^2/h$	39.7	33.2	5.3
$10^2(k+1)^2/h$	83.7	52.2	5.4
$10^3(k+1)^2/h$	123	70.9	5.4

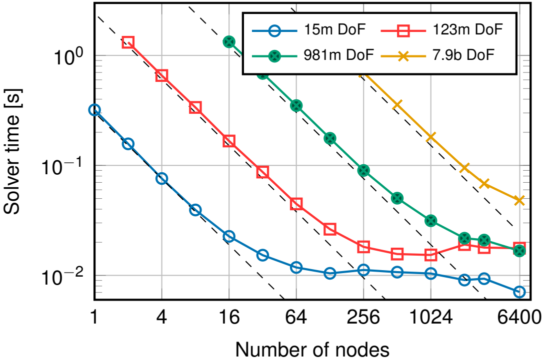
[7] Fehn, Munch, Wall, Kronbichler, Hybrid multigrid methods for high-order discontinuous Galerkin discretizations, *J Comput Phys* 415:109538, 2020

Scalability of multigrid solver on SuperMUC-NG (Intel Skylake CPU)

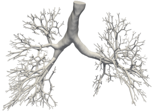
Point-Jacobi/Chebyshev (3,3) smoother, tolerance 10^{-3}

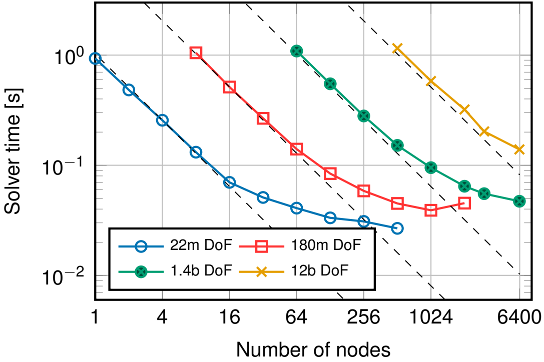
- Generic bifurcation
- Converges in 3 CG iterations





- Lung geometry
- Converges in 7 CG iterations





[8] Kronbichler, Fehn, Munch, Bergbauer, Wichmann, Geitner, Allalen, Schulz, Wall: A next-generation discontinuous Galerkin fluid dynamics solver with application to high-resolution lung airflow simulations. In *SC'21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, (2021).

Outline

Applications from our CoE dealii-X

The deal.II Finite Element Library

Matrix-free Operator Evaluation and Multigrid Solvers in deal.II

- Motivation
- Iterative Solvers

Advances in dealii-X CoE

Advances on linear solvers and preconditioners

- Not all problems work well with matrix-free algorithms $\rightarrow$ provide state-of-the-art matrix-based solvers using
  - Algebraic multigrid and related features through PSCToolkit
  - Sparse direct solvers with MUMPS
- Advances to geometric multigrid solvers in deal.II
  - Non-nested multigrid methods, based on totally flexible hierarchy of meshes[9]
  - Agglomeration-based multigrid with novel agglomeration strategies[10]

[9] Feder, Heltai, Kronbichler, Munch: Matrix-free implementation of the non-nested multigrid method. Submitted, 2025

[10] Feder, Cangiani, Heltai: R3MG: R-tree based agglomeration of polytopal grids with applications to multilevel methods. *J Comput Phys* 526, (2025).

Parallel Sparse Computation Toolkit – `psctoolkit.github.io`

Two central libraries PSBLAS and AMG4PSBLAS:

- Existing software standards:
  - MPI, OpenMP, CUDA
  - Serial sparse BLAS,
  - (Par)Metis,
  - AMD
- Attention to performance using **modern Fortran**;
- Research on new preconditioners;
- No need to delve in the data structures for the user;
- Tools for error and mesh handling beyond simple algebraic operations;
- **D**istributed **Sparse BLAS**;
- Standard **K**rylov **s**olvers: CG, FCG, (R)GMRES, BiCGStab, CGS, ...

Two central libraries PSBLAS and AMG4PSBLAS:

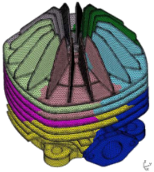
- Domain decomposition preconditioners
- **A**lgebraic **M**ulti**G**rid with aggregation schemes
  - Vaněk, Mandel, Brezina Aggregation
  - Matching Based
  - Smoothed Aggregation
- Parallel Smoothers (Block-Jacobi, Hybrid-GS/SGS/FBGS, ℓ_1 variants) that can be coupled with specialized block (approximate) solvers MUMPS, SuperLU, Incomplete Factorizations (AINV, INVK/L, ILU-type), and with Polynomial Accelerators (Chebyshev 1^{st}-kind, Chebyshev 4^{th}-kind)
- V-Cycle, W-Cycle, K-Cycle

Parallel Sparse Computation Toolkit – `psctoolkit.github.io`

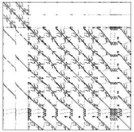
Two central libraries PSBLAS and AMG4PSBLAS.

- Freely available from: `https://psctoolkit.github.io`,
- Open Source with BSD 3 Clause License,
- Can be compiled/installed with either *Automake*/*CMake* or *Spack.io*: “`spack install psblas`”.
- See: D’Ambra, P., F. Durastante, and S. Filippone. “Parallel Sparse Computation Toolkit.” *Software Impacts* 15 (2023): 100463; for a **description of the architecture**.

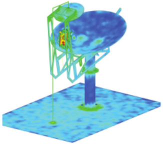
Sparse direct solvers



Code Aster (EDF)



Wide range of applications (e.g. medical diagnostics, geoscience, electromagnetism, circuit simulation, structural analysis ...)



FEKO-EM (Altair)

Sparse direct linear solvers

Factor $\mathbf{A} = \mathbf{LU}$; Solve: $\mathbf{LY} = \mathbf{B}$, then $\mathbf{UX} = \mathbf{Y}$

Method of choice for its accuracy and robustness

MUMPS: a MUltifrontal Massively Parallel Solver

MUMPS

A robust package using a direct method for solving

$$\mathbf{A}\mathbf{X} = \mathbf{B},$$

where **A** is a large sparse matrix, and $\mathbf{X}, \mathbf{B}$ are dense or sparse

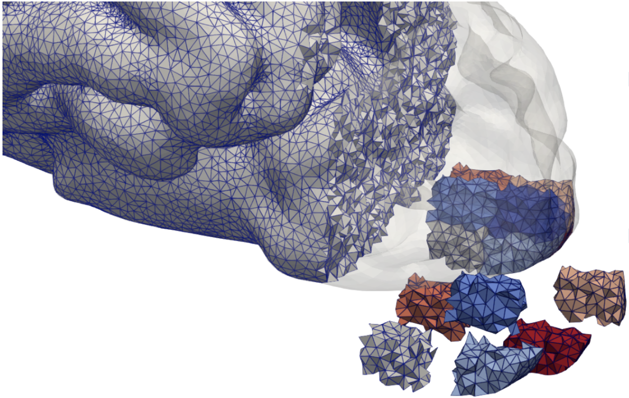
A free software distributed under CeCILL-C license (LGPL like), co-developped by Bordeaux Univ., CERFACS, CNRS, ENS Lyon, INPT, Inria, Mumps Tech, and Sorbonne Univ.

- Recent improvements cover low-rank approximations and mixed precision[11]
- Relies on data sparsity to reduce complexity

[11] Amestoy, Boiteau, Buttari, Gerest, Jézéquel, L'Excellent, Mary: Mixed precision low-rank approximations and their application to BLR LU factorization. *IMA J. Numer. Anal.* (2023)

Polytopic discontinuous Galerkin discretizations & geometric multigrid

by A. Cangiani (SISSA),
M. Feder, L. Heltai (Pisa)



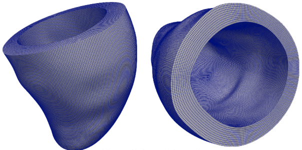
C++ deal.II library polyDEAL:
`https://github.com/fdrmrc/Polydeal`

- Agglomeration of standard mesh for
  - Upscaling of complicated geometries
  - Generation of coarse grids in view of geometric multigrid
- Based on
  - R-tree algorithm $\rightarrow$ `boost.org`
  - Our own implementation of hierarchy traversal within deal.II
- Allows for
  - Generation of Nested hierarchy of meshes
  - Optimal load-balancing
  - Constant wall-clock times independent of extraction (agglomeration) level

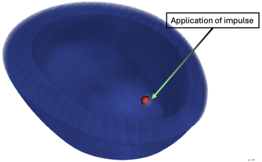
Cardiac electrophysiology monodomain model

by P. Africa (SISSA) & M. Feder (Pisa)

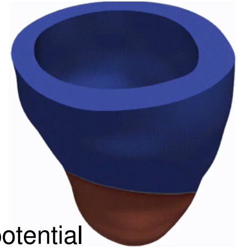
See also [Hoermann et al. 2018], [Africa et al., 2023], [Antonietti et al., 2024]



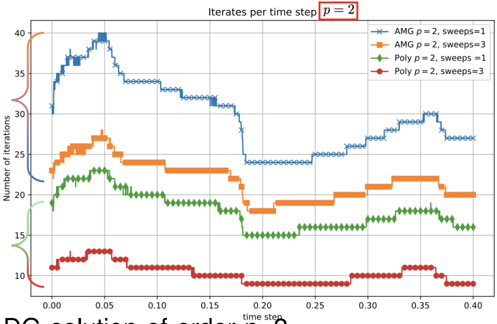
Views of left ventricle mesh



Realistic mesh and pysiological data of left ventricle



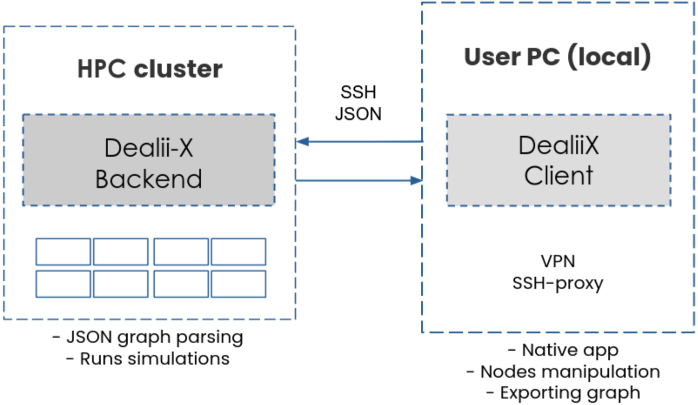
Transmembrane potential



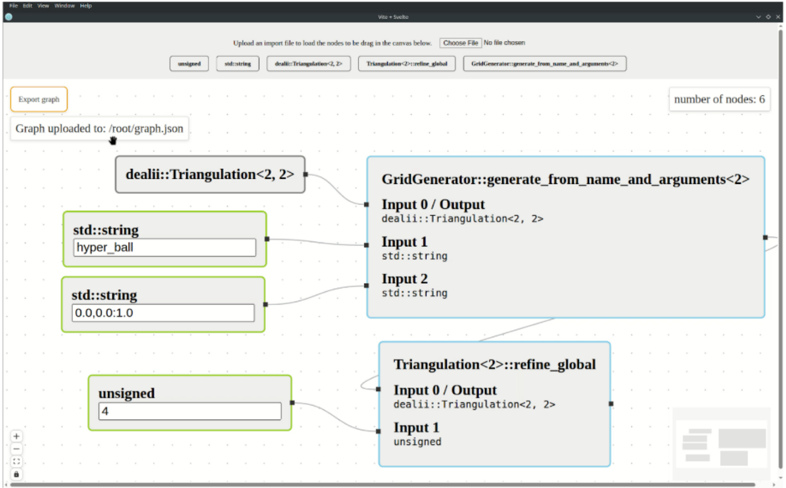
DG solution of order p=2
Conjugate Gradient with
V-cycle agglomerated geometric-algebraic
Multigrid vs standard AMG:
Number of iteration roughly halved.

by G. Brandino, F. De Giorgi (ExactLab, Dualistic)

- Develop GUI-based interface to make non-programmers productive with dealii-X
- Set up dedendencies in code by nodes in a graph
- Provide user intuitive handles to access various components of model
- Applicable on small systems and on big HPC clusters



Development of no-code / low-code interface: Sneak peak



Summary

- Development of deal.II library vibrant and active
- Target common mathematical formulations of partial differential equations, with focus on finite element problems
- The CoE dealii-X, started in Oct 2024, supports its developements
- Focus on robust solvers for coupled multi-physics problems
- Work on iterative solvers with matrix-free ingredients (very fast), sparse iterative solvers with robust preconditioners via PSCToolkit (fast, medium robust) and sparse direct solvers with MUMPS (slower, robust)
- Mixed precision, GPU portability, scalability to large node counts