Archive of Numerical Software
vol. 5, no. 2, 2017, pages 1–14
DOI: 10.11588/ans.2017.1.11815

`DOpElib`: Differential Equations and Optimization Environment; A Goal Oriented Software Library for Solving PDEs and Optimization Problems with PDEs

Christian Goll[*1], Thomas Wick[†2], and Winnifried Wollner[‡3]

[1]d-fine GmbH, An der Hauptwache 7, 60313 Frankfurt, Germany
[2]Centre de Mathémathiques Appliquées, École Polytechnique, Université Paris-Saclay, 91128 Palaiseau, France
[3]Technische Universität Darmstadt, Fachbereich Mathematik, Dolivostr. 15, 64293 Darmstadt, Germany

Received: Dec 12th, 2013; **final revision:** Apr 26th, 2017; **published:** July 27th, 2017.

Abstract: In this article, we describe the *Differential Equations and Optimization Environment* (`DOpElib`). `DOpElib` is a software library that provides a unified interface to high level algorithms such as time-stepping methods, nonlinear solvers and optimization routines. This structure ensures that, first of all, the user is only required to write those sections of code that are specific to the considered problem. Second, the exchange of parts of the used routines is possible with only a few lines of code to change instead of large reimplementations. The article illustrates the design principles, their reasoning, and various features of `DOpElib` and provides some numerical results as demonstration for the versatility of the software.

1 Introduction

Today, there exists a broad variety of software libraries that provide numerical tools for solving partial differential equations (PDEs). Prominent examples are `deal.II` [Arndt et al., 2017], dune [DUNE], FreeFem++ [Hecht, 2012], OpenFOAM [openfoam], FEniCS [FEniCS], PETSc [Balay et al., 2013], Trilinos [Heroux et al., 2003], Gascoigne [Gascoigne], DAKOTA [Adams et al., 2009], and many more. In addition, a lot of work is done in Matlab. In fact, all these libraries provide very useful tools for many applications.

Typically, all resources that are required to implement and solve PDEs are provided. Depending on the design decisions, some of these tools can be exchanged easily, while others can not. These can be, for instance, linear solvers or the finite element (FE) shape functions under consideration. Recently, there is an increasing demand to couple such PDE solvers with optimization routines to solve PDE-constrained optimization problems, as it is demonstrated for instance by PETSc [Balay

[*]hello@goll.me
[†]thomas.wick@polytechnique.edu
[‡]wollner@mathematik.tu-darmstadt.de

et al., 2013] including its TAO project [Munson et al., 2012], the Trilinos [Heroux et al., 2003] rapid optimization library (ROL) or FEniCS [FEniCS] together with its Dolfin library [Logg and Wells, 2010] in the Dolfin-Adjoint project [Farrell et al., 2012, Funke and Farrell, 2013], in which the discretized optimization problem is solved. Alternatively, the infinite-dimensional optimization problem can be approximated, as it is proposed, for instance in [Kirchner et al., 2011, Clever et al., 2012]. Such implementations are realized for example in FreeFem++ [Hecht, 2012], or in the optimization toolkit RoDoBo [RoDoBo] based on Gascoigne [Gascoigne], from which the authors obtained the initial idea for `DOpElib`.

The presented toolkit `DOpElib` has started with the aim to provide such optimization algorithms for PDE constrained optimization problems based upon a flexible finite element toolkit. To allow switching of optimization routines as well as finite elements and PDE-solvers, `DOpElib` provides both, modules for the solution of optimization problems, as well as modules providing tools for the solution of PDEs discretized using `deal.II`.

Of course other software libraries or combinations of such libraries are also capable of providing the desired main goal, to solve PDE-constrained optimization problems. However, they are all of very recent origin leaving a lot of room for competing implementations. Second, when starting from the capability to solve a PDE using any of the available toolkits, the efficient coupling with the desired optimization routines requires additional work, unique to the chosen PDE toolkit, in any case.

From our perspective the power of `DOpElib` is three-fold:

- Structured interface for high-level optimization routines integrated with tools to solve PDEs;
- Link to `deal.II` that is used around the world and provides a powerful basis;
- A collection of many challenging numerical benchmark examples.

Moreover, `DOpElib` features:

- An extensive online documentation (see also Section 2.4);
- A well-documented source code;
- Regression tests to check the library after modifications.

Our article is organized as follows. In Section 2, we formulate the goals of our software package and its main features. In Section 3, we provide three representative numerical examples illustrating the versatile features of `DOpElib`. In Section 4, we provide a short description that explains how to obtain `DOpElib`.

2 Goals and Features of `DOpElib`

As our guiding design principle, we note that in many numerical studies, the main task is the computation of some target functional, e.g., error norms, point values, mean values or technical quantities such as drag or lift coefficients. As a second motivating principle, there is interest in solving optimization problems in which a PDE is included as a constraint. Such optimization problems comprise optimal control, parameter estimation, inverse problems, or optimal experimental design. To do this in complex applications requires tremendous work already on the PDE itself before considering optimization of the processes modeled by this PDE. Thus it is advantageous to have an environment that allows to focus on the development of an appropriate code for the PDE at first but ensures, due to the required interface, that the code written for the PDE is then reusable for the application of state-of-the-art optimization routines.

It is the observation of the authors that in many master or dissertation projects much work is first spent on implementing a solver for a particular, complicated, PDE and then once this

is done very basic optimization routines are used since the code does not allow for an easy enough implementation of the needed routines for the optimization constrained by this PDE. The *Differential Equations and Optimization Environment* `DOpElib` DOpElib solves this problem by providing a well structured interface for high-level routines integrated with tools to solve PDEs. At the present stage, we provide an interface to the finite element library `deal.II` and a collection of algorithms to solve both pure PDE problems as well as optimization problems constrained by a PDE.

2.1 Global Features

The main feature of `DOpElib` is to provide a unified interface to high level algorithms such as time-stepping methods, nonlinear solvers, and optimization routines. `DOpElib` is designed in such a way that the user only needs to write those parts of the code that are problem dependent while all invariant parts of the algorithms are reusable without any need for further coding. In particular, the user is enabled to switch between various different algorithms without the need to rewrite the problem dependent code, though obviously he or she will have to replace the algorithm object with another one. This replacement can be done by replacing the appropriate object at only one point in the code. For instance, if the user wishes to change the time-stepping scheme, only one line of the code, identifying the scheme, needs to be changed in the main file without adjusting the specific PDE implementation. Similarly, if within an optimization problem the optimization solver should be changed, this can again be done by changing one line of the main file, plus possibly, additional include files. Furthermore, the exchange of many `deal.II` objects, e.g., different vector and matrix types (e.g., `Vector` and `BlockVector`) and finite elements, can be done with one line of code.

In addition to the finite element code provided by `deal.II` (which at present is the FE-toolkit to which we provide an interface) the presented library `DOpElib` is user-focused by delivering prefabricated tools which require adjustments by the user only for parts connected to his specific problem. This is in contrast to `deal.II`, which leaves the implementation of many high-level algorithms to the user, either by writing the code themselves or by writing an interface to appropriate libraries.

To accomplish these goals, the interfaces of the library are chosen in such a way that certain features (e.g. time stepping schemes, linear solvers, switching between stationary and nonstationary code versions) can be done by changing a few lines of code. This modularity is achieved by using a template-based approach, since methods of these objects tend to be called on each element of the finite element subdivision. The alternative choice would have been the use of a class hierarchy, i.e., the usage of virtual interface-classes instead of template parameters. However, we opt to utilize templates for two reasons mainly. First, and most important, the use of template parameters instead of base-classes provides a possible advantage in runtime at the cost of a longer compile time. In a typical program solving a sophisticated PDE or optimization problem, the runtime is magnitudes larger than compile time, thus aforementioned trade off looks attractive. Second, the prerequisites for an interface class are not always fulfilled. We have, for example, a template parameter PROBLEM, which describes the problem we try to solve. This can be a pure stationary or nonstationary partial differential equation, or an optimization problem, that can also be stationary or nonstationary. Using the template approach rather than class inheritance allows stationary problems to have significantly fewer methods since several functions are not required for the solution of stationary problems.

2.2 Specific Features

To achieve the previously formulated goals, in coupling the finite element toolkit `deal.II` with state of the art optimization routines as provided for instance by, SNOPT or IPOPT [Wächter and Biegler, 2006], we provide an interface that is capable of separating the following components into independent modules:

- Time discretization (i.e., different time-stepping schemes);
- Nonlinear solvers for PDEs;
- Linear solvers;
- Optimization solvers;
- Handling of unknowns in space and time;
- Using different meshes and finite elements for different variables;
- Description of the problem (e.g., the PDE, goal functional evaluations, and the objective functional for optimization).

A change in any of these modules requires very little effort for the user. In particular, the modules are independent of the specific implementation done in any of the other modules. On the other hand, adaptation of each module to the specific needs of the given problem is possible.

To provide a more detailed overview, the following key features are currently supported by the library. First, we provide wrappers and classes encapsulating several of the standard tasks in the PDE solution using `deal.II`. These are

- Solvers for nonlinear stationary and nonstationary PDEs in any dimension (1, 2, and 3 are as well supported by `deal.II`).
- Various, easily exchangeable, time-stepping schemes (based on finite differences), such as forward Euler, backward Euler, Crank-Nicolson, shifted Crank-Nicolson, and Fractional-Step-Θ scheme.
- Integration routines that allow the user to implement the PDE by simply writing the element or face contributions of the weak formulation.
- Use of all finite elements from `deal.II` including hp-support.
- Goal-oriented mesh adaptivity with the Dual Weighted Residual (DWR) method, see Becker and Rannacher [1996, 2001] or Bangerth and Rannacher [2003], and standard residual based error estimators (also for optimization problems).
- Several examples showing the numerical solution of PDEs including Poisson, Navier-Stokes, plasticity, fluid-structure interaction problems, the coupled Biot-Lamé-Navier system in subsurface modeling, and the Black-Scholes equations from financial mathematics.

Second, we have added several features useful for the solution of optimization problems subject to PDEs:

- Line search and trust region Newton algorithms for the solution of optimization problems with PDE constraints by elimination of the state variable, see, e.g., Becker et al. [2007]. In contrast to standard optimization routines, these are aware of the problem dependent choice of the topology.
- Interfaces to SNOPT and IPOPT for the solution of optimization problems with PDEs and additional control and state constraints.
- Several examples showing how to solve various kinds of optimization problems involving stationary and nonstationary PDE constraints.
- Different spatial triangulations for control and state variables without the need to reimplement the user dependent code when switching from one common triangulation to different ones.

2.3 Numerous Numerical Tests and Benchmarks

To date, two highlights of `DOpElib` are, firstly, implementations of several challenging benchmarks; and secondly, more than 30 tests in total that cover various aspects of stationary and nonstationary PDEs, linear and nonlinear solvers, 2D and 3D, discontinuous Galerkin, hp-elements, a posteriori error estimation, and different optimization settings with different controls as well as different cost functionals.

The underlying configurations of the benchmarks and numerical tests come from different application areas such as the Poisson problem, Navier-Stokes, elasticity, plasticity, fluid-structure interaction problems, porous media, and finally from financial mathematics the Black-Scholes equations.

The most important PDE examples are:

- Laplace equations in 2D and 3D
- Stationary elasticity benchmark Ramm et al. [2002]
- Stationary plasticity benchmark Ramm et al. [2002]
- Stationary Stokes equations with hp-finite elements
- Stationary fluid-structure interaction (FSI)
- Nonstationary Navier-Stokes benchmarks of Schaefer and Turek Turek and Schäfer [1996]
- Nonstationary fluid-structure interaction benchmarks FSI 1, 2, 3 of Hron and Turek Hron and Turek [2006], Bungartz and Schäfer [2006]
- Black-Scholes equation
- Heat equations
- Biot-Lamé-Navier problem from porous media with the Mandel-Cryer effect
- Isothermal Euler equations

The most important optimization examples are:

- Distributed and parameter control of linear elliptic PDEs
- Control in Dirichlet boundary values
- Distributed control with different meshes for control and state
- Distributed control showing how to interface external third party optimization libraries
- Topology optimization
- Parameter control with stationary Navier-Stokes equations
- Parameter control with stationary fluid-structure interaction
- Control of a semi-linear heat equation

2.4 Documentation / Manual

`DOpElib` comes with an extensive online documentation (pdf manual and doxygen) at `http://www.dopelib.net/`. The manual describes the features of this section in more detail. Moreover, detailed installation instructions and license information including FAQ are provided. Next, all examples are well-documented therein, including links to library parts that are unique to each specific example. Finally, an explanation of the purpose and usage of the regression tests is given.

3 Applications

In this section, we present three representative numerical tests demonstrating different features of `DOpElib`. For further details on the implementation, we provide the references to the corresponding examples of `DOpElib` which can be seen in the libraries manual:

- In the first numerical example, we consider goal-oriented mesh refinement with the help of the DWR method for the stationary Navier-Stokes equations. The code is a simple combination of the stationary Navier-Stokes problem, which can be easily obtained from `Examples/PDE/InstatPDE/Example1` and the DWR method whose usage is described in `Examples/PDE/StatPDE/Example5` for the Laplace equation.
- In the second application, we consider flow in and deformation of a porous medium. Here, numerical results of the augmented Mandel's configuration Girault et al. [2011] using the Biot equations (a PDE system that couples stationary linearized elasticity and a parabolic pressure equation) are presented. The code of this example can be found in the library under `Examples/PDE/InstatPDE/Example6`.
- In the third numerical test, an optimization problem for structural mechanics is discussed. The code of this example can be found in the library under `Examples/OPT/StatPDE/Example8`.

3.1 Goal-Oriented Mesh Refinement for Navier-Stokes

In this example, we consider a laminar flow around a cylinder in 2d. To be more precise, we compute the benchmark 2D-1 from Turek and Schäfer [1996]. The equation is discretized with the Taylor-Hood element. We want to compute the drag force on the enclosed cylinder

$$J(u) = \frac{1}{20} \int_{\Gamma_{circ}} (\nu \partial_n v_1 - p n_1) \, \mathrm{d}s, \tag{1}$$

with the solution $u = (v, p)$, the normal n and the viscosity $\nu = 0.001$.

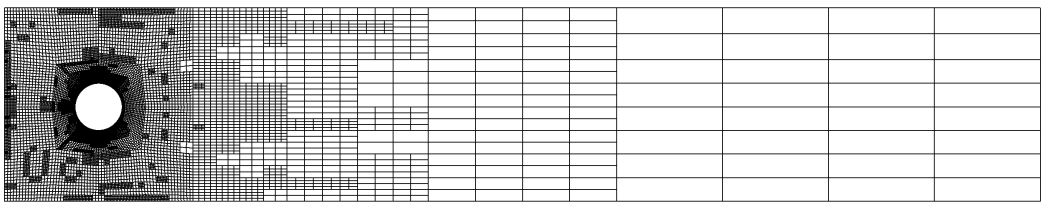

Figure 1: Locally adapted grid after eight refinement steps.

To achieve an efficient computation of the quantity J, we employ goal-oriented mesh refinement with the help of the DWR-method. The method delivers error indicators and by this way computational grids tailored to minimize the error in the previously given functional J. The price we have to pay is the additional computation of a (linear) dual problem. In Fig. 1 we show a resulting triangulation after eight refinement steps with roughly 320 000 degrees of freedom.

In Fig. 2 we compare the errors in the functional obtained by using uniform and local mesh refinement. We see clearly that local mesh refinement allows for substantial savings with respect to degrees of freedom.

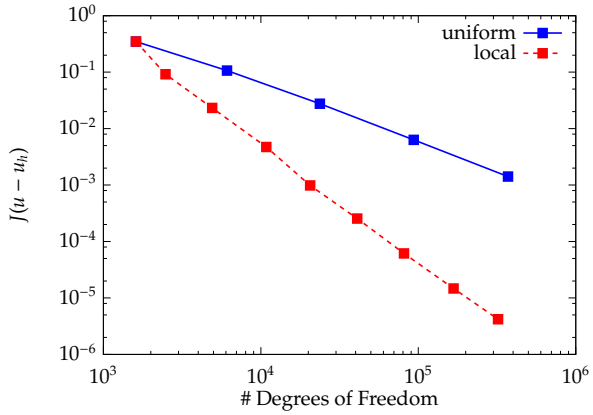

Figure 2: Comparison of uniform and local mesh refinement with respect to the error in the drag-functional J.

3.1.0.1 What needs to be implemented? In order to realize the above configuration, the user needs to provide the following problem-specific data:

- A file describing the initial mesh, readable by `deal.II`'s `GridIn` class.
- A `deal.II` function describing the inflow profile.
- A class implementing the local element and face contributions required for the PDE. A template that can be adjusted is available within `DOpElib`. For details, we refer to the manual.
- A class implementing the local face contributions of the drag functional. The implementation is quite similar to the PDE.
- Optionally, a parameter file for setting run time values, e.g., steering the nonlinear and linear solvers, can be given.

With these data, the user can initialize the methods from `DOpElib`. To this end, the user has to:

- Select the spatial discretization (finite elements) and quadrature rules.
- Attach the Dirichlet values to appropriate sections of the boundary.
- Set the functional in which the error should be estimated.
- Select the nonlinear and linear solvers to be used.

With these preparations, the solution of the benchmark, can be obtained by calling a method named `ComputeReducedFunctionals` from the appropriate solver object. To obtain the DWR error indicators, the same objects provides a method named `ComputeRefinementIndicators`, that internally solves the required adjoint problem and assembles the element-wise error indicators. To adjust the mesh according to these indicators, the DoF-handling object of `DOpElib`, provides a method `RefineSpace`.

It would be very simple to change the previous stationary problem to a nonstationary version, the only change needed is that the user also needs to:

- Select the time-stepping scheme.
- Select how the space-time unknowns should be stored, e.g., all in main memory or partly on disk.
- Replace some of the stationary objects by nonstationary ones as described in the manual.

3.2 Augmented Mandel's benchmark with Mandel-Cryer effect

In this example, a porous media zone is coupled with an overburden solid zone. The porous medium is described with the help of the Biot equations Biot [1941a,b, 1955]. The nonstationary coupled system for the state is formulated within a variational monolithically-coupled framework.

Let d be the dimension and $V_P := \{\varphi^p \in H^1(\Omega_B) | f = p^D \text{on } \Gamma_p\}$ and $V_S := \{\varphi^u \in [H^1(\Omega_B \cup \Omega_S)]^d | g = u^D \text{on } \Gamma_u\}$ be Hilbert spaces and p^D and u^D extensions of Dirichlet data.

Formulation 3.1 (Biot-Lamé-Navier Problem) *Find* $\{p, u\} \in \{p^D + V_P\} \times \{u^D + V_S\}$, *such that* $p(0) = p^0$, *for almost all times* $t \in I$, *and*

$$\begin{aligned} c_B(\partial_t p, \varphi^p) + \alpha_B(\nabla \cdot u, \varphi^p) + \frac{K}{\nu_F}(\nabla p, \nabla \varphi^p) & \\ -\rho_F(g, \varphi^p) - (q, \varphi^p) = 0 \quad & \forall \varphi^p \in V_P, \\ (\sigma_B, \nabla \varphi^u) - \alpha_B(pI, \nabla \varphi^u) - (f_B, \varphi^u) = 0 \quad & \forall \varphi^u \in V_S, \\ (\sigma_S, \nabla \varphi^u) - (f_S, \varphi^u) = 0 \quad & \forall \varphi^u \in V_S. \end{aligned}$$

We first discretize in time and then in space. Temporal discretization is based on the backward Euler scheme. We simply employ a CG finite element Galerkin discretization in space for both subdomains. Specifically, in the pay-zone (porous medium), we use an inf-sup stable element Q_1^c/Q_2^c (continuous bi-linear pressures and continuous bi-quadratic displacements) quadrilaterals Girault and Raviart [1986]. However, equal order elements Liu [2004] or mixed finite element methods could have been used as well. The non-pay zone, the solid overburden, is discretized with Q_2^c elements.

The computational domain is given by $[0, 100m] \times [0, 40m]$ and is sketched in Figure 3.

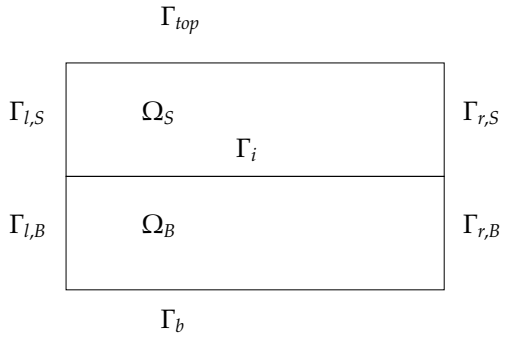

Figure 3: Configuration of augmented Mandel's problem.

The first delicate issues is the prescription of the interface conditions on Γ_i between the porous medium and the solid domain. Secondly, the pressure variable only lives in the pay-zone Ω_B. Here, the `FE Nothing` element provided by `deal.II` is very useful.

In more detail, the interface conditions on Γ_i are:

$$\begin{aligned} u_B &= u_S, \\ \sigma_B(u_B) n_B - \sigma_S(u_S) n_S &= \alpha p_B n_B, \\ -\frac{1}{\eta_f} K(\nabla p_B - \rho_f g) \cdot n_S &= 0. \end{aligned} \tag{2}$$

It is the second condition in (2), which requires careful implementation on the interface. As outer boundary conditions, we choose:

$$\begin{aligned} \sigma_B n &= \bar{t} \quad \text{on } \Gamma_{top}, \\ \sigma_B n &= 0 \quad \text{on } \Gamma_{r,S}, \\ p &= 0 \quad \text{on } \Gamma_{r,B}, \\ u_y &= 0 \quad \text{on } \Gamma_b, \\ u_x &= 0 \quad \text{on } \Gamma_{l,B} \cup \Gamma_{l,S}. \end{aligned}$$

As parameters, we choose: $M_B = 2.5 \cdot 10^{12}\,\text{Pa}$, $c_B = \frac{1}{M_B\,\text{Pa}}$, $\alpha_B = 1.0$, $\nu_F = 1.0 \cdot 10^{-3}\frac{\text{m}^2}{\text{s}}$, $K_B = 100\,\text{m}\,d = 10^{-13}\,\text{m}^2$ (where d is the unit for Darcy. It holds $1d = 9.87 \times 10^{-13}\,\text{m}^2$.), $\rho_F = 1.0\frac{\text{kg}}{\text{m}^3}$, $\bar{t} = F = 1.0 \cdot 10^7\,\text{Pa}\cdot\text{m}$. As elasticity parameters in Biot's model, we use $\mu_S = 10^8\,\text{Pa}$, $\nu_S = 0.2$. In addition, we use in the pure elastic zone the same Lamé coefficients, such that $\mu_B = \mu_S$ and $\lambda_B = \lambda_S$. The time step is chosen as $k = 1000s$. The final time is $5 \cdot 10^6 s$ (corresponds to computing 5000 time steps).

We observe the pressure and the horizontal displacements for different time steps $t_1 = 1\times10^3, t_2 = 5\times10^3, t_3 = 1\times10^4, t_4 = 1\times10^5, t_5 = 5\times10^5$ and $t_{10} = 5\times10^6[s]$ (the numeration of time steps is taken from Gai [2004]).

This problem does also show the well-known Mandel-Cryer effect (first increasing pressure, later decreasing pressure; for detailed references, we refer to Chapter 5 in Gai [2004]) as illustrated in Figure 4. Graphical solutions of the surfaces of pressure distribution are shown in Figure 5. Here, we detect the typical pressure behavior on the x-axis.

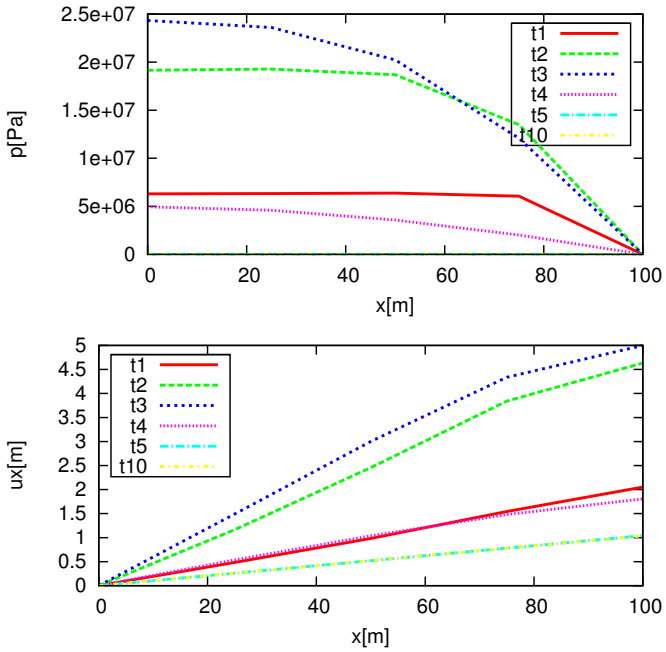

Figure 4: Augmented Mandel's problem: Pressure trace and x-displacement on the x-axis (when $y = 0$). The last two pressure curves t_5 and t_{10} are nearly invisible because they coincide with the x-axis (the pressure being nearly 0). The first increasing and then decreasing pressure is known as Mandel-Cryer effect.

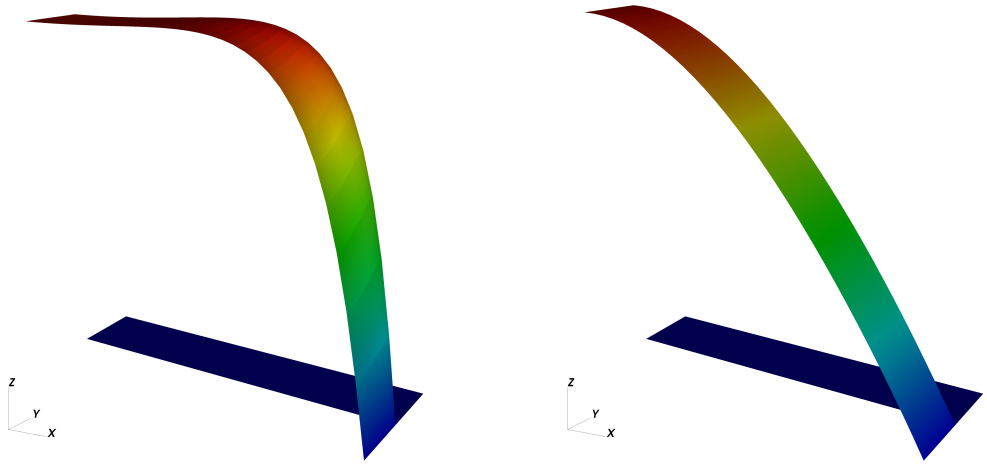

Figure 5: Augmented Mandel's problem: Pressure surface for two time steps t_1 and t_4. The pressure is only present in the lower half of the domain because the top is the overburden in which only elasticity is solved.

3.2.0.2 What needs to be implemented? Similarly to the first example, the user needs to provide the following problem-specific data:

- A class implementing the local element and face contributions required for the PDE. A template that can be adjusted is available within `DOpElib`. For details, we refer to the manual and in particular to the implementation of the example `PDE/InstatPDE/Example6`. Specifically, the interface terms (2) are implemented here.
- A class implementing the evaluation of the point values.
- A class indicating which finite elements should be used based upon the subdomain, distinguished by `deal.II`'s `material_id()`.
- Optionally, a parameter file for setting run time values, e.g., steering the nonlinear and linear solvers, can be given.

With these data, the user can initialize the methods from `DOpElib`. To this end, the user has to:

- Select the spatial and temporal discretization (finite elements and time-stepping scheme, respectively) and quadrature rules.
- Create a mesh describing the rectangular domain including the colorization according to the subdomains. Attach this mesh to the space-time DoF-handler object of `DOpElib`, which automatically create the tensor-product structure in space-time.
- Attach the Dirichlet and Neumann values to appropriate sections of the boundary.
- Select the nonlinear and linear solvers to be used.

With these preparations, the solution of the problem, can be obtained by calling a method named `ComputeReducedFunctionals` from the appropriate solver object.

3.3 Compliance Minimization

In this example, we consider the compliance minimization of a standard MBB beam (Messerschmidt-Bölkow-Blohm beam), see, e.g., Bendsøe and Sigmund [2003]. In order to enforce thickness values to be either 1 or 0 we employ the so-called SIMP-model (Solid Isotropic Material with Penalization), see Bendsøe and Sigmund [2003]. The discretization is done using Q_2 elements for the displacements and discontinuous P_0 elements for the thickness.

Here, we consider the domain $\Omega = (0,2) \times (0,1)$ in $\mathbb{R}^2$. On the upper boundary $\Gamma_1 = (0, 0.25) \times \{1\}$ we consider the constant force $f = (0, -1)^T$. For a given parameter $p > 0$, the discretized minimum compliance problem is to find a thickness ρ, and a corresponding displacement

$$u \in V_h := \{v = (v_1, v_2)^T \in Q_2 \,|\, v_1 = 0 \text{ on } \{0\} \times (0,1), v_2(2,0) = 0.\}$$

solving

$$\min_{\rho,u} \int_{\Gamma_1} f \cdot u \, \mathrm{d}s$$
$$\text{s.t.} \begin{cases} \int_\Omega \rho^p \nabla u : \nabla \varphi \, \mathrm{d}x = \int_{\Gamma_1} f \cdot \varphi \, \mathrm{d}s \quad \forall \varphi \in V_h, \\ \int_\Omega \rho \, \mathrm{d}x \leq 0.5, \\ 10^{-4} \leq \rho \leq 1. \end{cases}$$

If we pick the SIMP parameter $p = 1$ the thickness variable is allowed to take intermediate values other than 0 and 1, i.e., we consider a variable thickness sheet. A typical thickness distribution is

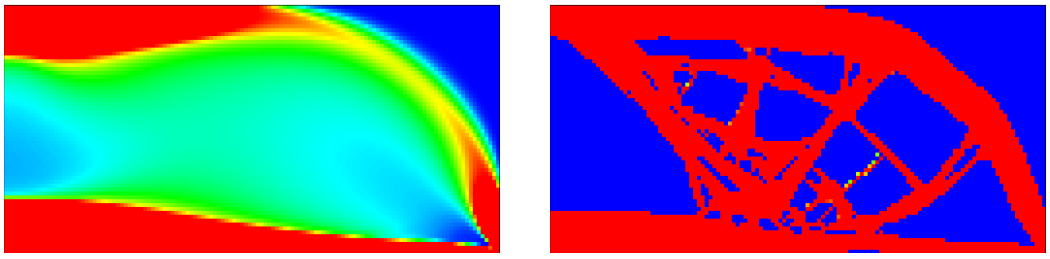

Figure 6: Thickness distribution for the MBB-beam minimum compliance problem using SIMP parameter $p = 1$ (left) and $p = 4$ right.

shown in the left image in Fig. 6. To obtain thickness values that are, mostly, either 1 or 0 we need to pick a larger value for p. The distribution given in the right image in Fig. 6 has been obtained using $p = 4$. Both results are in good agreement with the available literature, see, e.g., Bendsøe and Sigmund [2003], noting that no filter was applied for $p = 4$.

3.3.0.3 What needs to be implemented? In this third example, the user needs to provide the following problem-specific data:

- A class implementing the local element contributions required for the PDE. A template that can be adjusted is available within `DOpElib`. For details, we refer to the manual and in particular to the implementation of the example `OPT/StatPDE/Example8`.
- A class implementing the cost functional.
- An analogous class implementing the additional constraints on the mean value and the point value of the density.

- Optionally, a parameter file for setting run time values, e.g., steering the nonlinear and linear solvers, can be given.

With these data, the user can initialize the methods from `DOpElib`. To this end, the user has to:

- Select the spatial discretization (finite elements) and quadrature rules.
- Create a mesh describing the rectangular domain.
- Attach the Dirichlet and Neumann values to appropriate sections of the boundary. In particular the vertex $(0, 2)$ is constrained to a Dirichlet value.
- Select the nonlinear and linear solvers for the solution of the PDE.
- Select an appropriate optimization algorithm.

With these preparations, the solution of the problem, can be obtained by calling a method named `Solve` from the appropriate (optimization-) solver object.

4 How to get `DOpElib`

1. For all releases, tar-balls are provided at the web-site `http://www.dopelib.net`.
2. Alternatively, the current developer sources, including all features, bugs, and adjustments to, e.g., newer `deal.II` versions, can be obtained via git:

   ```
   git clone git://git.mathematik.tu-darmstadt.de/dopelib
   ```

Before installing `DOpElib`, the user is asked to install `deal.II` as the finite element tool. A detailed description for this process can be obtained on the projects web-site `http://www.dealii.org/`.

For the installation of `DOpElib`, we refer to the pdf documentation which can be found on `http://www.dopelib.net/`. The installation of optional third party libraries such as SNOPT and IPOPT, for the solution of constrained optimization problems, is described in the documentation as well. For the freely available library IPOPT an installation script is provided.

If you have made modifications to the code that you would like to share you can contact the maintainers by sending an E-Mail to `dope@dopelib.net`.

Acknowledgments

The `DOpElib` project makes use of various finite elements taken from the `deal.II`(version 8.5.0) Arndt et al. [2017] finite element library, which has been developed initially by W. Bangerth, R. Hartmann, and G. Kanschat Bangerth et al. [2007]. The authors acknowledge their past experience as well as discussions on modularization of algorithms with the authors of the libraries Gascoigne, initiated by Malte Braack and Roland Becker Gascoigne and RoDoBo, which were initiated by Roland Becker, Dominik Meidner, and Boris Vexler RoDoBo. Last, but not least, we would like to express our gratitude to Bernhard Endtmayer, Michael Geiger, Masoud Ghaderi, Daniel Jodlbauer, Uwe Köcher, Francesco Ludovici, Matthias Maier for their respective contributions to the library.

References

B. Adams, W. Bohnhoff, K. Dalbey, J. Eddy, M. Eldred, D. Gay, K. Haskell, P. Hough, and L. Swiler. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.0 user's manual. Technical Report SAND2010-2183, Sandia Technical Report, 2009. URL http://dakota.sandia.gov/.

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. *J. Numer. Math.*, 2017. doi: 10.1515/jnma-2016-1045.

S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc Web page, 2013. URL http://www.mcs.anl.gov/petsc.

W. Bangerth and R. Rannacher. *Adaptive Finite Element Methods for Differential Equations*. Birkhäuser Verlag, 2003.

W. Bangerth, R. Hartmann, and G. Kanschat. deal.II – a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4):24/1–24/27, 2007.

R. Becker and R. Rannacher. A feed-back approach to error control in finite element methods: basic analysis and examples. *East-West J. Numer. Math.*, 4:237–264, 1996.

R. Becker and R. Rannacher. *An optimal control approach to error control and mesh adaptation in finite element methods*, pages 1–102. Acta Numerica 2001, Cambridge University Press, a. iserles edition, 2001.

R. Becker, D. Meidner, and B. Vexler. Efficient numerical solution of parabolic optimization problems by finite element methods. *Optim. Methods Softw.*, 22(5):813–833, 2007.

M. P. Bendsøe and O. Sigmund. *Topology Optimization: Theory, Methods and Applications*. Springer, 2003.

M. Biot. Consolidation settlement under a rectangular load distribution. *J. Appl. Phys.*, 12(5): 426–430, 1941a.

M. Biot. General theory of three-dimensional consolidation. *J. Appl. Phys.*, 12(2):155–164, 1941b.

M. Biot. Theory of elasticity and consolidation for a porous anisotropic solid. *J. Appl. Phys.*, 25(2): 182–185, 1955.

H.-J. Bungartz and M. Schäfer. *Fluid-Structure Interaction: Modelling, Simulation, Optimization*, volume 53 of *Lecture Notes in Computational Science and Engineering*. Springer, 2006.

D. Clever, J. Lang, S. Ulbrich, and J. C. Ziems. *Constrained Optimization and Optimal Control for Partial Differential Equations*, volume 160 of *International Series of Numerical Mathematics*, chapter Generalized Multilevel SQP-methods for PDAE-constrained Optimization Based on Space-Time Adaptive PDAE Solvers, pages 37–60. Springer Verlag, 2012.

DOpElib. The differential equation and optimization environment: DOpElib. URL http://www.dopelib.net.

DUNE. DUNE distributed and unified numerics environment. URL http://www.dune-project.org.

P. E. Farrell, D. A. Ham, S. W. Funke, and M. E. Rognes. Automated derivation of the adjoint of high-level transient finite element programs. Technical Report 1204.5577, arXiv, 2012.

FEniCS. FEniCS project. URL http://fenicsproject.org.

S. W. Funke and P. E. Farrell. A framework for automated pde-constrained optimisation. Technical Report 1302.389, arXiv, 2013.

X. Gai. *A coupled geomechanics and reservoir flow model on parallel computers*. PhD thesis, The University of Texas at Austin, 2004.

Gascoigne. The finite element toolkit Gascoigne. URL http://www.gascoigne.de.

V. Girault and P.-A. Raviart. *Finite Element method for the Navier-Stokes equations*. Number 5 in Computer Series in Computational Mathematics. Springer-Verlag, 1986.

V. Girault, G. Pencheva, M. F. Wheeler, and T. Wildey. Domain decomposition for poroelasticity and elasticity with dg jumps and mortars. *Mathematical Models and Methods in Applied Sciences*, 21(1):169–213, 2011.

F. Hecht. New development in freefem++. *J. Numer. Math.*, 20(3-4):251–265, 2012.

M. Heroux, R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

J. Hron and S. Turek. *Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow*, volume 53, pages 146–170. Springer-Verlag, 2006.

A. Kirchner, D. Meidner, and B. Vexler. Trust region methods with hierarchical finite element models for pde-constrained optimization. *Control Cybernet.*, 40(4):1019–1042, 2011.

R. Liu. *Discontinuous Galerkin Finite Element Solution for Poromechanics*. PhD thesis, The University of Texas at Austin, 2004.

A. Logg and G. N. Wells. DOLFIN:Automated finite element computing. *ACM Trans. Math. Softw.*, 37(2), 2010.

T. Munson, J. Sarich, S. Wild, S. Benson, and L. C. McInnes. Tao 2.0 users manual. Technical Report ANL/MCS-TM-322, Mathematics and Computer Science Division, Argonne National Laboratory, 2012. URL http://www.mcs.anl.gov/tao.

openfoam. Openfoam. URL http://www.openfoam.org/.

E. Ramm, E. Rank, R. Rannacher, K. Schweizerhof, E. Stein, W. Wendland, G. Wittum, P. Wriggers, and W. Wunderlich. *Error-controlled Adaptive Finite Elements in Solid Mechanics*. Wiley, 2002.

RoDoBo. RoDoBo: A C++ library for optimization with stationary and nonstationary PDEs. URL http://www.rodobo.org.

S. Turek and M. Schäfer. Benchmark computations of laminar flow around cylinder. In E. Hirschel, editor, *Flow Simulation with High-Performance Computers II*, volume 52 of *Notes on Numerical Fluid Mechanics*, pages 547–566. Vieweg, 1996.

A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(2):25–57, 2006.

