Effective Integration of Sophisticated Operators in Isogeometric Analysis with `igatools`.

Nicola Cavallini, Oliver Weeger, M. Sebastian Pauletti, Massimiliano Martinelli, Pablo Antolín

Abstract `igatools` is a newly released library for operators assembly in isogeometric analysis. The library, which is object oriented designed and written in `C++11`, is general purpose, therefore it is not devoted to any specific application. In this paper we show that such a design makes `igatools` an effective tool in assembling isogeometric discretizations of sophisticated differential operators. This effectiveness will be demonstrated by showing code snippets relating one-to-one with the operators written on paper. To embrace a wide audience, applications from nonlinear incompressible solid and fluid mechanics will be addressed. In both cases we are going to deal with mixed isogeometric formulations. The applicative nature of this paper will be stressed solving industrially relevant tests cases.

Key words: isogeometric analysis, nonlinear incompressible elasticity, computational fluid dynamics

1 Introduction

Scientific computing is an area where expertise from several backgrounds such as computer science, engineering, mathematics, and physics come together. Contributions to this scientific field range from most theoretical to the most applied ones. With this work we address the numerical solution

N. Cavallini
Scuola Internazionale Superiore di Studi Avanzati, Via Bonomea 265, Trieste, Italy, e-mail: `nicola.cavalli@sissa.it`
O. Weeger
TU Kaiserslautern, Faculty of Mathematics, P.O. Box 3049, 67653 Kaiserslautern, Germany, e-mail: `weeger@rhrk.uni-kl.de`
M. S. Pauletti
Instituto de Matemática Aplicada del Litoral (IMAL), Consejo Nacional de Investigaciones científicas y técnicas (CONICET), Santa Fe, Argentinia.
M. Martinelli
Istituto di Matematica Applicata e Tecnologie Informatiche (IMATI), Consiglio Nazionale delle Ricerche (CNR), Pavia, Italy
P. Antolín
Università degli Studi di Pavia, Dipartimento di Ingegneria Civile ed Architettura, Via Ferrata 3, 27100 Pavia, Italy

of nonlinear partial differential equations arising from industrial applications in solid and fluid mechanics. The aim of the paper is to demonstrate that the newly released software library `igatools` allows an effective implementation of isogeometric finite element discretizations of rather complex differential operators, here in particular mixed formulations of nonlinear incompressible elasticity and Navier-Stokes equations, defined on complex domains.

As the software prefix `iga` suggests, we are dealing with isogeometric type of spaces. Isogeometric analysis has been introduced in [25] with the aim of bridging scientific computing and computational geometry. The basic idea here, is to use the shape functions that describe the geometry, as basis functions for a Galerkin method. Several aspects in computational modeling benefit from this linking. In particular, an appealing aspect of this approach is the higher inter element continuity that characterizes the shape functions describing the geometry and numerical solution. Among other contributions we would mention [14] for structural vibration, [28] for shell structures and [36] in non-linear vibration analysis. In fluid dynamics we would like to mention [6, 7] for a variational multi scale modeling approach to turbulent flows and [20] for divergence conforming B-splines.

`igatools` is an object oriented library for discretization of partial differential equations, using isogeometric type of spaces. It has been first presented in [30] and it is available under GPL conditions at www.igatools.org. The idea that underlays the software design is encapsulating mathematical concepts used in isogeometric method into objects, namely classes. Mapping the interaction between classes we get methods used in the actual integration. The software is implemented in `C++11` [8, 27] and it makes an extensive use of generic programming, templates in `C++`. In this context, generic programming is particularly useful to obtain dimension independent code, a very interesting feature of the software.

This paper is devoted to the community of scientists curious about numerical solution implementation of partial differential equations. In this respect we thought at two model readers. Those who might be skeptic about object oriented programming in scientific computing, and to those who already have an acquaintance with this programming style. We aim at involving the first class of readers showing that a carefully designed software can help in implementing non elementary operators. In the second case we would like to capture their attention in a collaborative way. To pursue this twofold objective, we think there is no better way rather than presenting a simple way of implementing sophisticated operators. The implementation ease will be demonstrated providing code snippets, that prove a one-to-one correspondence between the way we write the operators on paper and the way these operators are coded in assembly loops.

Validation of the software will be provided with respect to literature test cases and, industrial applications will be presented to prove feasibility of the software in applicative contexts. In order to embrace the widest possible audience, we present two systems of partial differential equations, one arising from solid mechanics and one from fluid dynamics. In both cases the problems are nonlinear. Though this paper is not focused on numerical treatment of nonlinear systems, we would like to remark that the nonlinearities treatment is a matter of linear algebra strategies, that do not interfere with operators assembly.

One of the goals of this paper is to show that `igatools` can be an effective tool in industrially relevant applications. `igatools` development has been supported by several institutions and projects. In the context of this paper, a special mention goes to the TERRIFIC project, as the presented applications are partial fulfillment of the project itself. TERRIFIC (Towards Enhanced Integration of Design and Production in the Factory of the Future through Isogeometric Technologies) is a project within the seventh framework program of the European Union, that lasted from 1st of September 2011 to the 31st of August 2014. The aim of the project was to inject isogeometric

analysis in an industrial context. The geometries presented are courtesy of ALENIA aeronautica, in particular the authors are grateful to L. Morrone and G. Mirra. The other supporting institutions and projects will be listed in the acknowledgments.

In Section 2 we continue with a brief summary of igatools design. As the applications we are going to deal with require mixed formulations, we recall the isogeometric formulation of Taylor-Hood elements in Section 3. In Section 4 we detail our computational mechanics applications and in Section 5 our fluid dynamics applications are addressed. Finally, in Section 6 we conclude the paper.

2 igatools Design Description

The software prefix iga restricts the discussion to isogeometric methods. Isogeometric methods commonly address Galerkin approximations, characterized by high inter element continuity of basis functions. Originally, in historical sense, basis functions for isogeometric methods are B-Splines and NURBS, the interested reader can refer to [31, 32] for a precise definition, and to [30] for igatools related notation. Other than these, new reference spaces have been introduced lately. The reader can refer to [5, 16, 12, 33] for T-splines, to [15] for locally refined splines, and to [21, 23, 35] for hierarchical splines. In this paper, we are going to use B-splines. B-splines of degree d and regularity r ($0 \leq r < d$) will be addressed as Splines S_r^d. In this section we briefly review the objects, or classes, that build the backbone of the software. As extensively explained in [30], these objects are designed to resemble the key mathematical concepts in isogeometric methods. A visual sketch of the major players is outlined in Figure 1.

Reference Space (BSplineSpace and NURBSSpace)

The classes BSPlineSpace and NURBSSpace represent the shape functions defined on the parametric domain (usually the unit square in two dimensions and the unit cube in three dimensions). These spaces can be non-homogeneous, in the sense that they can be characterized by different degrees and regularities in each dimension.

Mapping (Mapping)

The mathematical mapping is the object that maps the parametric domain into the physical one. Notice that, in principle, there is no reason why this mapping should be restricted to isogeometric type of maps. Moreover, in the desired case of an isogeometric type of map, Mapping is not supposed to be refined together with the reference space to achieve convergence.

Push Forward (PushForward)

The PushForward object combines the Mapping together with the TransformationType. The transformation type defines how to transform functions. Depending on the different transformations,

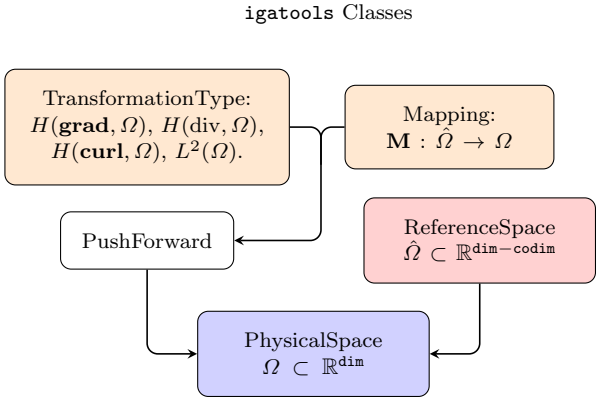


Fig. 1 In this sketch we picture the major object we isolated in isogeometric analysis, together with their mathematical symbols. We outlined that in general the dimension of the reference space can differ from the physical space, this difference is addressed as codimension `codim`.

different operators can be preserved throughout the transformation itself. Example transformations are $H(\mathbf{grad}, \Omega)$, $H(\mathrm{div}, \Omega)$, $H(\mathbf{curl}, \Omega)$ and $L^2(\Omega)$.

Physical Space (`PhysicalSpace`)

The combination of the `ReferenceSpace` with the `PushForward` gives the `PhysicalSpace`. This class contains all the information necessary to recover point values of functions and derivatives on the physical domain. This point values are the ones used to assemble the desired operators.

Element Iterator (`ElementIterator`)

One of the similarities between standard finite elements and the isogeometric method, is the decomposition of the domain in a collection of elements where a small number of functions have support. The global matrix, is built *iterating* through all the elements in the grid. In object oriented programming the mechanism that points to an element, and modifies itself to point to the next element is called iterator. In [30] we pointed out that we consider the previously mentioned classes as *grid-like* containers, this why the element level information is accessed via the `ElementIterator` class.

For an immediate flavor on how a simple $\int \nabla u \nabla v$ operator is assembled, we sketch a sort of "Hello World" program, in Listing 1. In the first four lines we define the presented classes. We start with a `CartesianGrid`, namely the knots without repetitions. This class is used to build the corresponding `BSplineSpace`. The transformation type (`h_grad`, in this case) couples with a `Mapping` to build the `PushForward`. Finally it only remains to instantiate the `PhysicalSpace` as the combination of `BSplineSpace` and `PushForward`. The mechanism that implies the initialization of the cache `elem->init_values()` and its filling is detailed in [30]. In this paragraph we limit ourselves to emphasize that we access the shape function values, using `PhysicalSpace` as a grid-like container, to assemble the local matrix.

```
auto grid = CartesianGrid<dim>::create(box, n_knots);
auto ref_space = BSplineSpace<dim>::create(grid, deg);
map       = BallMapping<dim>::create(grid);
space     = PhysicalSpace<BSplineSpace<dim>, PushForward<h_grad, dim>>::create(
    ref_space, PushForward<h_grad, dim>::create(map));

auto elem = space->begin(); const auto elem_end = space->end();
ValueFlags fill_flags = ...
elem->init_values(fill_flags, elem_quad);

for (; elem != elem_end; ++elem){
    elem->fill_values();
    for (int i = 0; i < n_basis; ++i){
        for (int j = 0; j < n_basis; ++j){
            for (int qp = 0; qp < n_qp; ++qp){
                loc_mat(i,j) += scalar_product(
                    elem->get_basis_gradient(i,qp),
                    elem->get_basis_gradient(j,qp))* w_meas[qp];}}}
```

Listing 1 A first "Hello World" like code. Intended to assemble a simple $\int \nabla u \, \nabla v$ operator.

3 Mixed methods with isogeometric finite elements

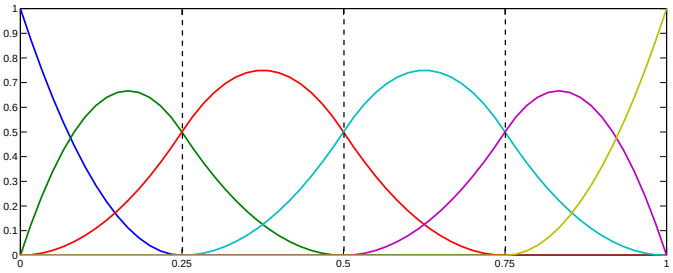
The mathematical description of physical phenomena involves several mechanisms interplaying at the same time. It often happens that different independent variables are necessary to describe a given mechanism. When such independent variables are discretized using different solution spaces we have a mixed method.

A typical case is an equation of motion coupled with an incompressibility constrain. In the applications we are going to address, namely incompressible elasticity (Section 4) and incompressible Navier-Stokes (Section 5), the independent variables are two. The first one will be referred to as $\mathbf{u}$ and will be used to describe velocities or displacements. The second one is going to be p that represents the Lagrangian multiplier of the incompressibility condition. In both mechanical and fluid dynamical contexts the discretization and linearization of the continuous operators will lead to a block linear system of the form:

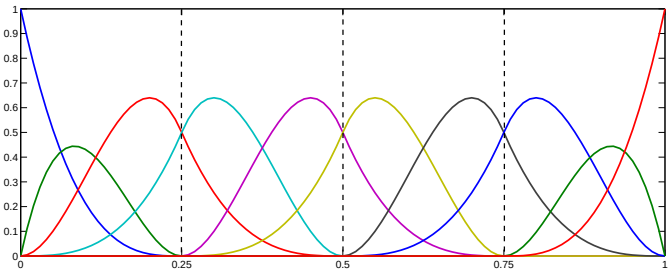
$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}.$$

If we consider the two discrete spaces $\mathcal{V}_h$ and $\mathcal{Q}_h$ for $\mathbf{u}$ and p respectively, it is well known that these two spaces have to satisfy the inf-sup condition, see [1, 9, 10]. In standard finite elements, an established choice is the Taylor-Hood element. Here the discrete spaces for $\mathbf{u}$ and p share the same triangulation and the inf-sup condition is fulfilled when the degree of the shape functions for $\mathbf{u}$ is $d+1$, being d the shape functions degree for p.

The isogeometric counterpart of standard Taylor-Hood elements is constructed choosing $\mathcal{V}_h = (S^{d+1}_{d-1})^{\mathtt{dim}}$ and $\mathcal{Q}_h = S^d_{d-1}$, being `dim` the dimension. This type of element has been first used in [4] and a stability proof can be found in [11]. For sake of clearness we represent in one dimension the shape functions corresponding to S^3_1/S^2_1 pair in Figure 2.



(a) Space $\mathcal{Q}_h = \mathcal{S}_1^2$ for the Lagrangian multiplier p



(b) Space $\mathcal{V}_h = \mathcal{S}_1^3$ of primal variable $\mathbf{u}$

Fig. 2 In this figure we sketch one dimensional shape functions for the Taylor-Hood type of spaces. In particular we represent S_1^3 splines for the velocity, and S_1^2 splines for the pressure. The corresponding knot vectors are: $\Xi_{\mathbf{u}} = \{0, 0, 0, 0, 0.25, 0.25, 0.5, 0.5, 0.75, 0.75, 1, 1, 1, 1\}$ for the velocity and $\Xi_p = \{0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1\}$ for the pressure.

As mentioned in Section 2, the software design should resemble the mathematical concepts needed to integrate the boundary value problem we are interested in, in a one-to-one correspondence. As different spaces are required to solve for $\mathbf{u}$ and p, the same situation will be reproduced in the code.

Listing 2 reports the code necessary to define $\mathcal{Q}_h$ and $\mathcal{V}_h$. We first focus on the solution space for p, namely `BSplineSpace<dim>`. As previously mentioned, one of the most appealing features of `igatools` is its code dimension independence. In the present case of a scalar space, we only need to define the first template parameter as `dim`, the other template parameters are default defined to give a scalar space. A bit different is the situation for the vector space $\mathcal{V}_h$. In this case mathematics is elegantly resembled by specifying the second template parameter, the equations range is the same as the problem dimension, this is why we get `BSplineSpace<dim,dim>`.

In code snippet in Listing 2 the reader can also notice how two different vectors, `mult_u` and `mult_p`, are defined to assign the right multiplicity to each knot. Following the Taylor-Hood element definition, the two spaces must have the same regularity and different degree, and we need to set the knots multiplicity to obtain compatible spaces.

```
using PreSpace = BSplineSpace<dim>;
using VelSpcae = BSplineSpace<dim, dim>;

deg_u = deg_p + 1;

shared_ptr<BSplineSpace<dim>> pre_space;
shared_ptr<BSplineSpace<dim, dim>> vel_space;

vector<int> mult_p(n_knots, deg_p - reg);
vector<int> mult_u(n_knots, deg_u - reg);
mult_p[0] = mult_p[n_knots-1] = deg_p + 1;
mult_u[0] = mult_u[n_knots-1] = deg_u + 1;

pre_mult.fill(mult_p);
vel_mult.fill(mult_u);

pre_deg.fill(deg_p);
vel_deg.fill(deg_u);

auto grid = CartesianGrid<dim>::create(n_knots);
pre_space = PreSpace::create(grid, pre_mult, pre_deg);
vel_space = VelSpace::create(grid, vel_mult, vel_deg);
```

Listing 2 Definition of to different spaces for the variables $\mathbf{u}$ and p. In this listing we detail the construction of spaces with different degrees and multiplicites.

4 Computational Mechanics: Nonlinear Incompressible Elasticity

As a first application for the `igatools` implementation of a mixed method, we present the isogeometric discretization of nonlinear elasticity, i.e. large deformation kinematics and a hyperelastic constitutive law, subject to an incompressibility constraint.

Incompressible elasticity has been studied already by a few researchers in the context of isogeometric formulations. Elguedj et al. investigated the use of $\bar{B}$- and $\bar{F}$-projection methods for linear and nonlinear incompressible elasticity. Taylor [34] and Mathisen et al. [29] employed a three-field mixed approach for nonlinear incompressible elasticity with independent approximation of $\mathbf{u}$, p and θ. Here we use a classical mixed method with independent variables $\mathbf{u}$ and p, as presented in the previous Section, and outline its implementation using `igatools`. The developed solver is validated using a well-known benchmark example and then applied to an industrial test case.

In this paper we employ the Langrangian or material description of the large deformation elasticity problem in $\texttt{dim} = 2, 3$ dimensions, as for instance introduced in [1, 37]. Thus, a body in its undeformed, Lagrangian configuration is given by its reference or material domain $\Omega \in \mathbb{R}^{\texttt{dim}}$ and its deformed, current configuration is expressed with the mapping $\hat{\varphi} : \Omega \to \mathbb{R}^{\texttt{dim}}$:

$$\hat{\varphi}(\mathbf{X}) = \mathbf{X} + \hat{\mathbf{u}}(\mathbf{X}). \tag{1}$$

Following the notation of [1] from the displacement field $\hat{\mathbf{u}}$ we can derive the kinematic quantities such as the deformation gradient

$$\hat{\mathbf{F}} = \mathbf{F}(\hat{\mathbf{u}}) = \mathbf{I} + \nabla\hat{\mathbf{u}}, \tag{2}$$

and the right Cauchy-Green strain tensor

$$\hat{\mathbf{C}} = \mathbf{C}(\hat{\mathbf{u}}) = \hat{\mathbf{F}}^T \hat{\mathbf{F}}. \tag{3}$$

$\hat{J} = \det \hat{\mathbf{F}}$ is the determinant of the deformation gradient and for a purely incompressible material it must hold $\hat{J} = 1$.

Using Lamé parameter μ and a pressure-like variable $\hat{p}$ we can introduce the elastic energy functional for a purely incompressible Neo-Hookean material:

$$\mathit{\Pi}_{elast}(\hat{\mathbf{u}}, \hat{p}) = \int_\Omega \left\{ \frac{1}{2}\mu \left[\mathbf{I} : \hat{\mathbf{C}} - d \right] - \mu \ln \hat{J} + \hat{p} \ln(\hat{J}) \right\} d\Omega - \mathcal{F}(\hat{\mathbf{u}}, \gamma). \tag{4}$$

Here $\mathcal{F}(\hat{\mathbf{u}}, \gamma)$ denotes the external energy functional and has the form

$$\mathcal{F}(\hat{\mathbf{u}}, \gamma) = \gamma \left(\int_\Omega \mathbf{f} \cdot \hat{\mathbf{u}} d\Omega + \int_{\mathit{\Gamma}_N} \mathbf{p} \cdot \hat{\mathbf{u}} d\Omega \right), \tag{5}$$

where $\mathbf{f}$ is a body load, $\mathbf{p}$ a traction force acting on the Neumann boundary of the domain $\mathit{\Gamma}_N \subset \delta\Omega$ and $\gamma \in \mathbb{R}$ a load parameter.

For the minimization of the energy functional (4) it follows from the calculus of variations that the total differential of $\mathit{\Pi}_{elast}$ must be zero,

$$\mathrm{d}\mathit{\Pi}_{elast}(\hat{\mathbf{u}}, \hat{p})[\mathbf{v}, q] = 0,$$

for all generic virtual displacement fields $\mathbf{v}$ and pressure fields q. Explicitly this reads:

$$\begin{cases} \mu \displaystyle\int_\Omega [\hat{\mathbf{F}} - \hat{\mathbf{F}}^{-T}] : \nabla \mathbf{v} + \int_\Omega \hat{p}\ \hat{\mathbf{F}}^{-T} : \nabla \mathbf{v} - \mathcal{F}(\mathbf{v}; \gamma) = 0 \quad \forall \mathbf{v}, \\ \displaystyle\int_\Omega \ln(\hat{J}) q = 0 \quad \forall q. \end{cases} \tag{6}$$

The nonlinear system of equations (6) is then discretized using isogeometric Taylor-Hood elements in order to obtain displacements $\hat{\mathbf{u}} \in \mathcal{V}_h$ and pressures $\hat{p} \in \mathcal{Q}_h$. We also need to evaluate the second derivative of the energy functional (4), in order to setup a Newton's method for our solver:

$$\mathrm{d}^2\mathit{\Pi}_{elast}(\hat{\mathbf{u}}, \hat{p})[(\mathbf{u}, p), (\mathbf{v}, q)] = a_\gamma(\mathbf{u}, \mathbf{v}) + b_\gamma(\mathbf{v}, p) + b_\gamma(\mathbf{u}, q), \tag{7}$$

where

$$\begin{cases} a_\gamma(\mathbf{u}, \mathbf{v}) := \mu \displaystyle\int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} + \int_\Omega [\mu - \hat{p}] \left(\hat{\mathbf{F}}^{-1} \nabla \mathbf{u}\right)^T : \hat{\mathbf{F}}^{-1} \nabla \mathbf{v}, \\ b_\gamma(\mathbf{v}, q) := \displaystyle\int_\Omega q\ \hat{\mathbf{F}}^{-T} : \nabla \mathbf{v}. \end{cases} \tag{8}$$

This formulation is a good example of what the authors intend as "sophisticated formulation". If we were to give a precise definition of "sophisticated" in this context, we would define it as an open scale, where grade zero is the classical $\int_\Omega \nabla u \cdot \nabla v$ arising from a Poisson's problem. Of course we do not aim grading every possible operator, but we think that a good measure of complexity is the number of manipulations beyond the classical grad – grad. It is in such cases that `igatools` can help.

In particular we focus on the term

$$\int_{\Omega} q\, \hat{\mathbf{F}}^{-T} : \nabla \mathbf{v}$$

from (7), which serves as a good example for the difficulties in this formulation. Its implementation is sketched in Listing 3. We first start by noticing that this is a mixed element. As we have two different spaces for $\mathbf{u}$ and p we need to iterate trough both of them and therefor we need two different iterators: the fist one is `defo_elem`, which iterates over the elements of the deformation space, and the second one is `prex_elem`, which iterates over the elements of the pressure like space. The single terms such as $\hat{\mathbf{F}}$, $\nabla \mathbf{v}$, or q are defined as tensors, and we refer to `igatools` documentation of the `Tensor` class for details [26]. Two classical for loops are used to evaluate $\nabla \mathbf{u}$ and p at quadrature points. Then $\hat{\mathbf{F}}^{-T}$ can be easily evaluated combing the inverse and the transpose functions. Once all the variables have been evaluated, they can be combined to obtain the local contribution to the tangent matrix evaluation. The remaining terms of the residual and tangent matrix are assembled in an analogous way. Once all the operators are in place, the setup of a Newton's method is a matter of linear algebra.

```
1  for (; defo_elem != defo_end ; ++defo_elem, ++prex_elem)
2  {
3      ...
4    for (Index q = 0; q < n_qp; ++q)
5    {
6      for(Index i = 0; i < defo_loc_ndofs; ++i)
7        defo_grad_q += defo_vec(dof) * grad_phi_q[i];
8
9      for(Index i = 0; i < prex_loc_ndofs; ++i)
10       prex_q += prex_vec(dof) * prex_phi_q[i];
11
12     defgrad_q = unit_defgrad + defo_grad_q;
13
14     inverse (defgrad_q,defgrad_inv_q);
15     defgrad_invT_q = transpose(defgrad_inv_q);
16
17     for (Index i = 0; i < defo_loc_ndofs; ++i)
18       defo_loc_res(i) += mat_mu * prex_q[0] *
19                   scalar_product(defgrad_invT_q, grad_phi_q[i]) *w_meas[q];
20     }
21 }
```

Listing 3 Assembly of the mixed term in the second variation of the energy functional.

For validation of the implementation we chose the well-known Cook's membrane problem as benchmark [13]. This is a well established test in the context of finite element and also isogeometric methods [17, 29]. It is used by a number of researchers to validate solvers and benchmark performance of discretizations. Cook's membrane is a 2D panel, clamped on its side and subject to a shear load on the right side, causing combined bending and shear deformation. Geometry, dimensions and values of parameters can be found in Figure 3. The quantity of interest is the horizontal

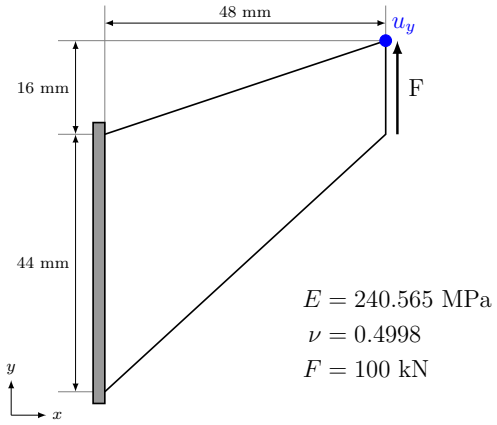


Fig. 3 Cook's membrane problem. A 2D panel clamped on the left side, subject to shear load on right side

deformation of the top right corner of the panel u_y, which is used to study p/k- and h-convergence and locking-free behavior of mixed elements.

In Figure 4 convergence behavior of u_y with respect to the total number of degrees of freedom is shown. We compare the isogeometric Taylor-Hood elements with pure displacement isogeometric formulations. The Poisson's ratio ν is set to 0.4998 that correspond to nearly incompressible material. The displacement-based show a rather slow convergence gradient due to locking. On the other hand, as expected, the Taylor-Hood elements converge quickly for p/k- and h-refinement. These results also match very well with the ones obtained by [17], where an isogeometric $\bar{F}$-method was used instead of a mixed formulation.

Even though the purely displacement formulation also converges towards the correct value of u_y for higher polynomial degree and number of degrees of freedom, the approximation of stresses exhibits spurious oscillations, which are typical for locking phenomena. In Figure 5 we plot the stress component σ_{xx}. The non mixed elements show a highly oscillatory behavior of volumetric stress components – a typical sign of volumetric locking – while the stress distributions for mixed elements are smooth.

The geometry of a pneumatic pipe, used in aircraft manufacturing, is one of the industrial applications within the TERRIFIC project. Using symmetry boundary conditions we can parametrize only half of the pipe as a B-Spline volume with degrees $d = (1, 2, 2)$ and $n = (2, 34, 3)$ control points, i.e. 204 control points of the volume. The first parameter direction is the thickness of the pipe, the second the circumferential direction and the third its length. For numerical computations we degree-elevate to $d = (2, 2, 2)$ and perform four steps of uniform h-refinement in the length-direction. Thus we have the following Taylor-Hood spaces for pressure and displacements:

$$\begin{aligned} d_p &= (2, 2, 2), & n_p &= (3, 34, 18), & N_p &= 1836, \\ d_{\mathbf{u}} &= (3, 3, 3), & n_{\mathbf{u}} &= (4, 66, 34), & N_{\mathbf{u}} &= 8976. \end{aligned} \tag{9}$$

here n stands for degrees of freedom per direction, while N indicates the total number of degrees of freedom. The material parameters of the pipe are:

$$E = 240.565 \text{ MPa}, \qquad \nu = 0.5, \qquad \rho = 1.0 \cdot 10^3 \text{ kg/m}^3. \tag{10}$$

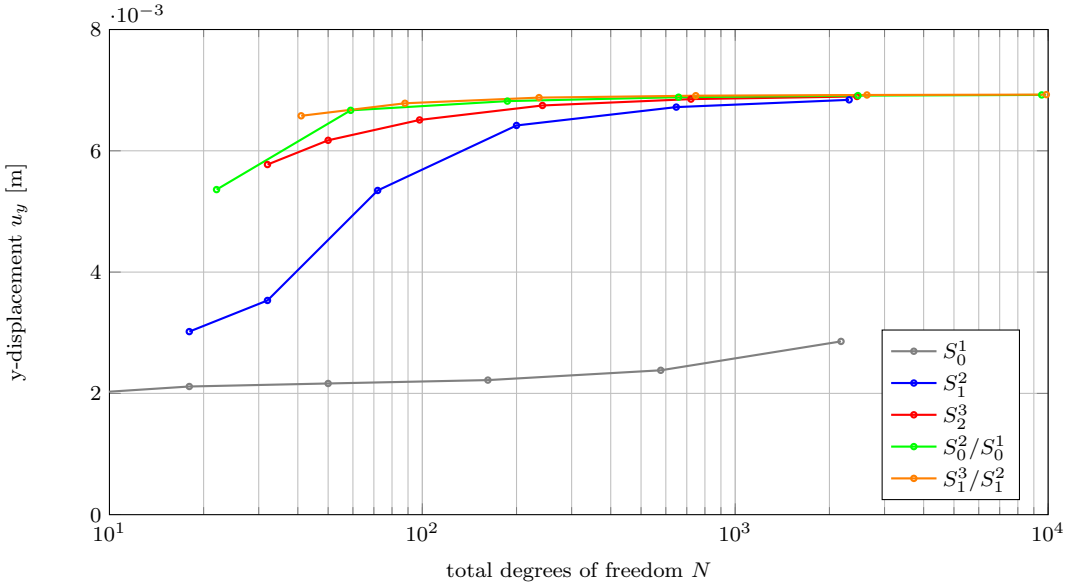


Fig. 4 Convergence test for the Cook's membrane. The reader can notice the fast convergence the Taylor Hood elements and a slower convergence for pure displacements formulation. In the following, the plots of the stresses, will demonstrate that volumetric locking is preventing from a fast convergence.

Boundary conditions are symmetry, zero displacement at long ends and an outwarded-directed Neumann load. We run 10 load steps increasing the magnitude of the surface load from 1.0 kN to 10.0 kN.

Figure 6 shows the deformed pipe after 10 load steps. Large displacements occur and no volume change is visible in the Figure. In Figure 7 the evolution of displacements u_x, u_y and pressure-like Lagrangian multiplier p, evaluated at the center point where maximum x-displacement and pressure occur, is plotted over the 10 load steps. The nonlinear behaviour can be noticed in the values of the displacements and pressure.

5 Computational Fluid Dynamics

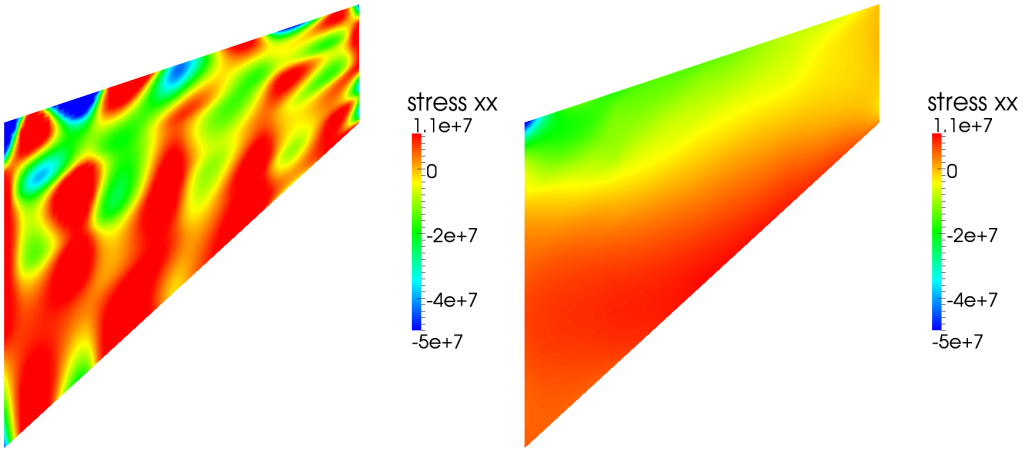
The laminar motion of an incompressible fluid is modeled by the the well known Navier-Stokes equations. We first write the momentum conservation equation:

$$\partial_t \mathbf{u} + (\nabla \mathbf{u})\, \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f},$$

then the mass preservation one:

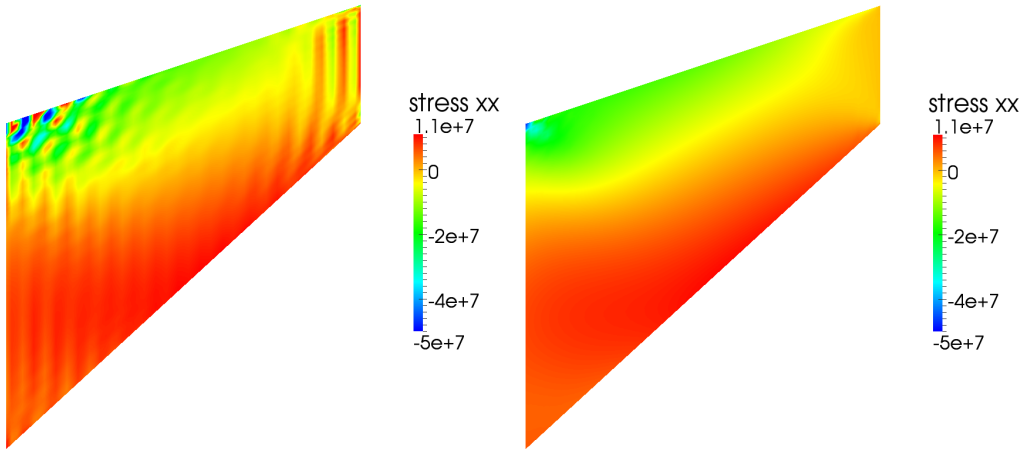
$$\nabla \cdot \mathbf{u} = 0.$$

Here $\mathbf{u}$ is the fluid velocity, p is the fluid pressure and ν is the kinematic viscosity. The external forces are denoted by $\mathbf{f}$. As a matter of clearness, since it is going to be useful for the method



(a) S_2^3 with 4 elements per side

(b) S_1^3/S_1^2 with 4 elements per side



(c) S_2^3 with 16 elements per side

(d) S_1^3/S_1^2 with 16 elements per side

Fig. 5 Stress σ_{xx} for purely displacement-based and mixed formulations. The reader can notice how purely displacement based formulations are affected from volumetric locking, while, as expected, mixed methods are not.

implementation, we explicitly write the component wise formulation of the momentum equation:

$$\partial_t \begin{pmatrix} u_x \\ u_y \end{pmatrix} + \begin{pmatrix} \partial_x u_x & \partial_y u_x \\ \partial_x u_y & \partial_y u_y \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix} = - \begin{pmatrix} \partial_x p \\ \partial_y p \end{pmatrix} + \nu \begin{pmatrix} \partial_{xx} u_x + \partial_{yy} u_x \\ \partial_{xx} u_y + \partial_{yy} u_y \end{pmatrix} + \begin{pmatrix} f_x \\ f_y \end{pmatrix}.$$

In the following, we focus on the stationary version of the problem:

$$(\nabla \mathbf{u})\,\mathbf{u} - \nu \nabla^2 \mathbf{u} + \nabla p = \mathbf{f}, \tag{11}$$

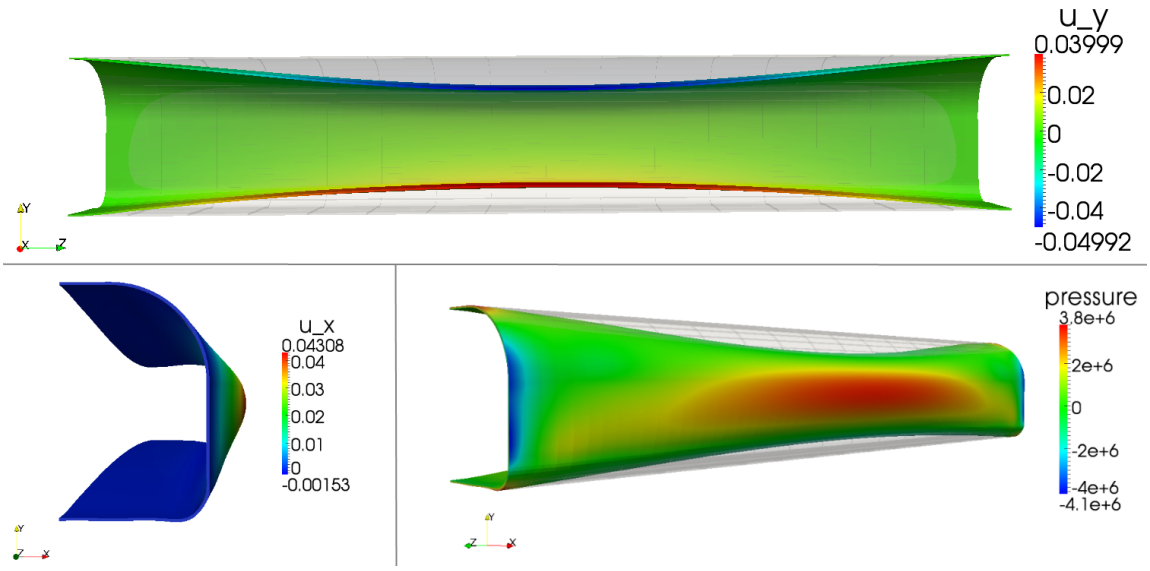
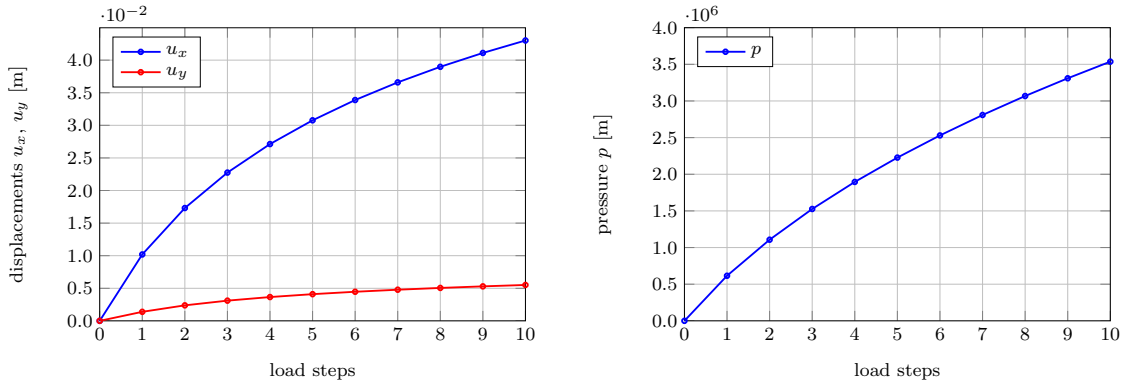


Fig. 6 Displacement and pressure of the pneumatic pipe after 10 load steps. Deformed pipe is colored by displacement resp. pressure, initial un-deformed configuration is shown in translucent grey.



(a) displacements u_x, u_y

(b) pressure-like Langrangian multiplier p

Fig. 7 Evaluation of point values over load steps for pneumatic pipe provided by ALENIA.

The spatial integration of equation (11) is again performed using the Galerkin approach. Given the computational domain $\Omega \subset \mathbb{R}^{\texttt{dim}}$, we consider $\mathcal{V}_h \subset (H^1(\Omega))^{\texttt{dim}}$ the solution space for the velocity, and $\mathcal{Q}_h \subset L^2(\Omega)$ the solution space for the pressure. We consider all the $\mathbf{v}_h \in \mathcal{V}_h$ the test functions for the velocity, then the assembled viscous term $\nabla^2\mathbf{u}$ correspond to the matrix A such that:

$$A_{ij} = \int_\Omega (\nabla \mathbf{u}_j) : (\nabla \mathbf{v}_i) = \int_\Omega \begin{pmatrix} \partial_x u_x & \partial_y u_x \\ \partial_x u_y & \partial_y u_y \end{pmatrix}_j : \begin{pmatrix} \partial_x v_x & \partial_y v_x \\ \partial_x v_y & \partial_y v_y \end{pmatrix}_i .$$

The convective term is a nonlinear one. For a given value of the velocity field we obtain the matrix N:

$$N_{ij} = \int_\Omega (\nabla \mathbf{u}_j)\, \mathbf{u} \cdot \mathbf{v}_i = \int_\Omega \begin{pmatrix} \partial_x u_x & \partial_y u_x \\ \partial_x u_y & \partial_y u_y \end{pmatrix}_j \begin{pmatrix} u_x \\ u_y \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}_i . \tag{12}$$

The operator that couples the momentum equation and the preservation of mass is B^T:

$$B^T_{ij} = \int_\Omega p_j \nabla \cdot \mathbf{v}_i = \int_\Omega p_j (\partial_x v_x + \partial_y v_y)_i .$$

These operators when combined together result in the following block nonlinear system:

$$\begin{pmatrix} A + N(\mathbf{u}) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}.$$

The nonlinearity is treated with a classical, residual-tangent strategy. Given an initial guess for the velocity $\mathbf{u}_0$ we first obtain the tentative solution $\mathbf{x}_k = (\mathbf{u}, p)_k$ by solving:

$$\begin{pmatrix} A + N(\mathbf{u}_0) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}_k = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}.$$

For $k > 0$ we need to form the residual $\mathbf{F}(\mathbf{x}_k)$:

$$\mathbf{F}(\mathbf{x}_k) = \begin{pmatrix} A + N(\mathbf{u}_k) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}_k - \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix},$$

and correct the tentative solution with an approximation of the tangent. The correction at the k-th step will be denoted as $(\delta\mathbf{u}, \delta p)_k$. The most popular strategies to approximate the tangent are the fixed point and the Newton approximation. In the first case we approximate the tangent with the function itself:

$$\begin{pmatrix} A + N(\mathbf{u}_k) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta\mathbf{u} \\ \delta p \end{pmatrix}_k = -\mathbf{F}(\mathbf{x}_k).$$

In the second case we consider a first order expansion of the operator:

$$\begin{pmatrix} A + N(\mathbf{u}_k) + D(\mathbf{u}_k) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta\mathbf{u} \\ \delta p \end{pmatrix}_k = -\mathbf{F}(\mathbf{x}_k).$$

In this case we need to assemble the function Jacobian as well:

$$D_{ij} = \int_\Omega (\nabla \mathbf{u})\, \mathbf{u}_j \cdot \mathbf{v}_i = \int_\Omega \begin{pmatrix} \partial_x u_x & \partial_y u_x \\ \partial_x u_y & \partial_y u_y \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix}_j \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}_i . \tag{13}$$

Finally, the solution is updated with its correction:

$$\begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}_{k+1} = \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}_k + \begin{pmatrix} \delta\mathbf{u} \\ \delta p \end{pmatrix}_k ,$$

the procedure stops when the residual satisfies a given tolerance.

Now we intend to focus on the two terms involved in the nonlinearity treatment. Looking at the convective term, we have the velocity gradient applied to the fluid velocity. This mathematical formulation has a pretty simple implementation in igatools. In Listing 4 we have the gradient of the trial function that acts, through the function action, on a point evaluation of the velocity vel_q. The result of this action is scalar multiplied with the test functions, by using a scalar_product function. In this way we obtain the local contribution to the global matrix. Almost the same procedure applies to the evaluation of the operator (13). In this case the quantity that has to be evaluated is the velocity gradient grad_vel_q. We evaluate the action of the velocity gradient on the trial functions, and then we perform the scalar_product with the test functions to get the local contribution to the global matrix, see Listing 5. Both vel_q and grad_vel_q are considered as tensors. We refer to the Tensor's class documentation for further details.

```
for (dof_index q = 0; q < quad.get_num_points(); q++)
{
   for (dof_index i = 0; i <local_ndofs; i ++){
      auto phi = element->get_value(i,q);
      vel_q = vel_q+vel[local_dofs[i]]*phi;}

   for (dof_index i = 0; i <local_ndofs; i ++){
      for (dof_index j = 0; j <local_ndofs; j ++){
         adv_ij = scalar_product(action(element->get_gradient(j,q),vel_q),
                                 element->get_value(i,q) )*
                                 element->get_w_measures()[q];]}}
}
```

Listing 4 Code snippet for the advection operator assembly. In this case the reader can notice the point evaluation of the velocity, the action of the velocity gradient on the velocity value, and the scalar product with the velocity test functions.

```
for (dof_index q = 0; q < quad.get_num_points(); q++)
{

   for (dof_index i = 0; i <local_ndofs; i++){
      auto grad_phi = element->get_gradient(i,q);
      grad_vel_q = grad_vel_q + vel[local_dofs[i]] * grad_phi;}

   for (dof_index i = 0; i <local_ndofs; i++){
      for (dof_index j = 0; j <local_ndofs; j++){
         jac_ij = scalar_product(action(grad_vel_q,element->get_value(j,q)),
                                 element->get_value(i,q) )*
                                 element->get_w_measures()[q];}}
}
```

Listing 5 Code snippet for the Jacobian assembly. In this case we have the point evaluation of the velocity gradient. Then the velocity gradient acts on the trial functions. The result of this multiplication is scalar multiplied with the test functions, to get the local contribution to the global matrix.

The solver is tested using the cavity flow test case. This is a typical test case for stationary Navier-Stokes equations [22, 24]. The horizontal velocity u_x is imposed at the top of a square domain. Its value is:

$$u_x(x) = \begin{cases} \sin(\pi x/2a) & \text{if } 0 \leq x \leq a \\ 1 & \text{if } a \leq x \leq 1-a \\ \sin(\pi(1-x)/2a) & \text{if } 1-a \leq x \leq 1 \end{cases}$$

with $a = 1/32$. No-slip boundary conditions are imposed elsewhere. It is known that isogeometric shape functions are not interpolatory to the degrees of freedom. It is then necessary to define values for the imposed degrees of freedom. In `igatools`, Listing 6, this is done defining a `CavityVel` class that inherits form the base class `Function` and providing implementation for the virtual function function `evaluate`. At this point the user is let to define an instance of his own class `CavityVel` and `igatools` will take care of evaluating an L^2 projection of the boundary function on the velocity trace space. The resulting `dof_values` will be applied to global matrix, right hand side and solution.

```
1  class CavityVel : public Function
2  {
3  public:
4      CavityVel () : Function() {}
5
6      void evaluate(const ValueVector & points,
7                    ValueVector & values)       const;
8  }
9  ...
10 CavityVel cv;
11 project_boundary_values(cv,velocity_space,face_q,direchelet_id,dof_values);
12 apply_boundary_values(dof_values,matrix,rhs,solution)
```

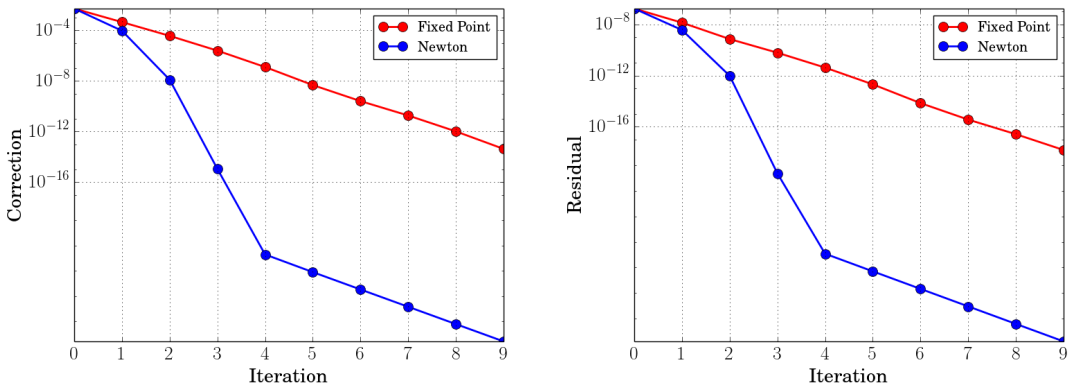
Listing 6 In this code snippet we present how to impose non constant boundary conditions for our problem. The user needs to create a derived class that represents the function to be evaluated. Implementation for the `evaluate` virtual function has to be provided. `igatools` will take care of L^2 projecting the function on the velocity boundary trace space. The resulting values will be applied to the global matrix, right hand side and solution.

Once the solver is implemented we start with the experiments. In Figure 8 we represent the residual and the correction for Reynolds's number Re $= 100$ for each iteration of the nonlinear solver. As expected, the residual and correction drop linearly with the fixed point type of iterations, and quadratically with the Newton's solver.

In the present case of the cavity flow experiment we used S_1^3 splines for the velocity and S_1^2 splines for the pressure with mesh size $h = 1/32$. Efficient linear algebra solvers for this system is still an active research area, the reader can refer to [18, 19] as example references. In this experiment and in the ones that follow case we set for a standard direct solver.

In a second set of experiments we consider increasing Re numbers. The solution space is still S_3^1/S_2^1, but the mesh size is 64×64 elements. In Figure 9 we present the velocity streamlines. The colormap pictures the velocity magnitude that ranges from zero to one. The agreement with results from [22] is good and the solver can be considered as validated.

The industrial test case we are considering is a ventilation outlet of an aircraft cabin, see Figure 10(a) for the geometry. Dirichlet inlet boundary conditions and Neumann boundary conditions at the outlet are applied with Re $= 10,000$. Shape functions spaces are S_1^3 for the velocity and S_1^2 for



(a) Solution correction. (b) Residual.

Fig. 8 Correction and residual for the cavity flow test case with Re = 100.

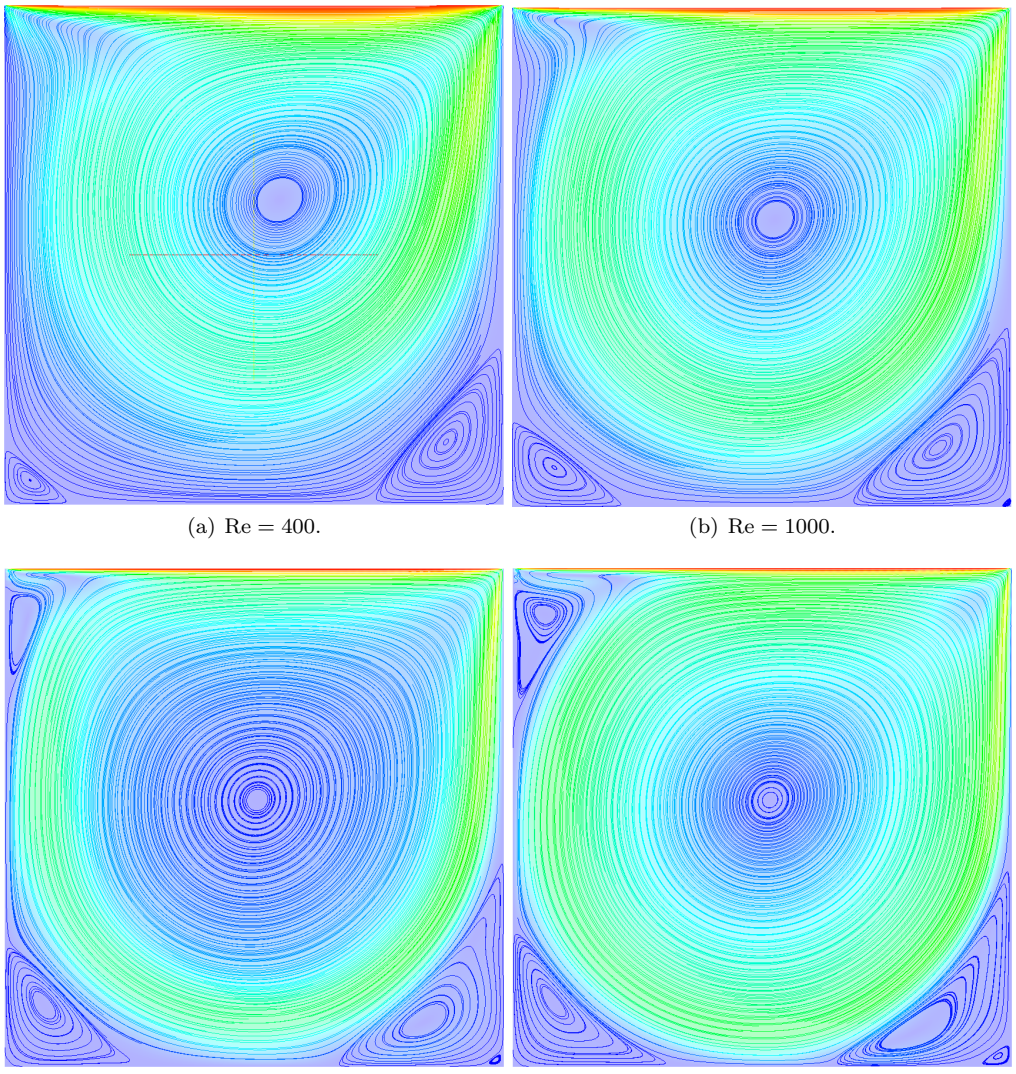
the pressure. The mesh is $16 \times 32 \times 8$ resulting in approximately 200,000 degrees of freedom for the velocity. The final results are visualized in Figure 10, where Figure 10(a) shows the pressure map and the velocity is represented in Figure 10(b). At this Reynolds number the fluid velocity does not diffuse uniformly along the outlet section and higher velocities are concentrated in the middle of the section.

6 Conclusions

In this paper we showed how `igatools` is an effective tool for assembling isogeometric discretizations of sophisticated operators. `igatools` has been recently released and its design has been presented in [30]. Only time will say if its design will be capable to attract new users and developers. We intended to demonstrate that we are on a good track in this direction. We picked sophisticated operators arising from industrial applications, and showed that a one to one relationship exists between how we write operators on paper, and how we implement them using `igatools`. We believe this attempt has been successful both in computational mechanics and computational fluid dynamics. In order to demonstrate a realistic effectiveness of the software we solved industrially relevant applications.

Acknowledgements

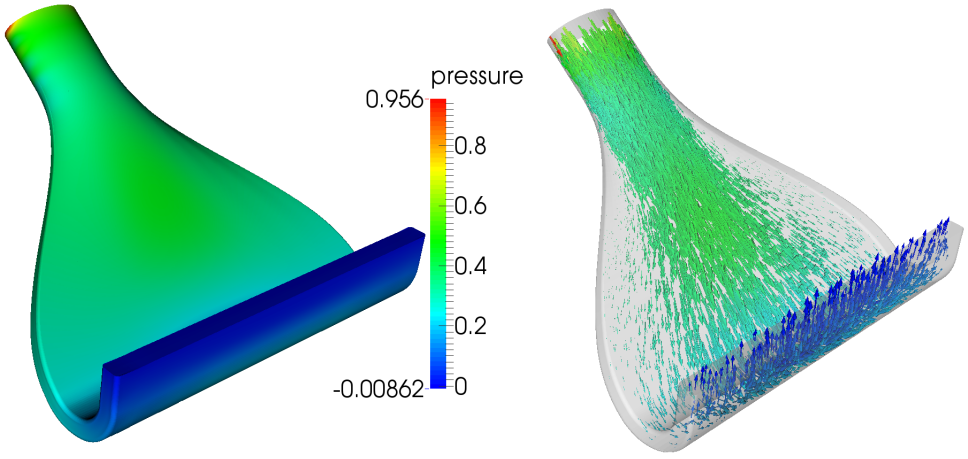
The authors are grateful to C. Lovadina for the useful discussion. N. Cavallini and O. Weeger have been supported by the TERRIFIC project, European Community's Seventh Framework Programme, Grant Agreement 284981 Call FP7-2011-NMP-ICT-FoF. In all the experiments in this paper we used linear algebra packages from `deal.II` [2, 3].



(a) Re = 400. (b) Re = 1000.

(c) Re = 3200. (d) Re = 5000.

Fig. 9 Cavity flow test case for the Navier-Stokes solver. Colormap represents the velocity magnitude in a scale the ranges from zero to one.



(a) Pressure map. (b) Velocity quiver.

Fig. 10 Pressure map and velocity quiver for the aircraft cabin ventilation outlet. In this simulation Re = 10000, shape functions are S_1^3/S_1^2, the number of knots, without repetitions, is $17 \times 33 \times 9$. Inlet velocity is imposed, stress free on the outlet, and no slip at the boundary.

References

1. Ferdinando Auricchio, Loureno Beiro da Veiga, Carlo Lovadina, Alessandro Reali, RobertL. Taylor, and Peter Wriggers. Approximation of incompressible large deformation elastic problems: some unresolved issues. *Computational Mechanics*, 52(5):1153–1167, 2013.
2. 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.
3. W. Bangerth, T. Heister, and G. Kanschat. `deal.II` *Differential Equations Analysis Library, Technical Reference.* `http://www.dealii.org`.
4. Y. Bazilevs, L. Beirão da Veiga, J. A. Cottrell, T. J. R. Hughes, and G. Sangalli. Isogeometric analysis: Approximation, stability and error estimates for h-refined meshes. *Math. Mod. Meth. Appl. S.*, 16(07):1031–1090, 2006.
5. Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg. Isogeometric analysis using T-splines. *Comput. Methods Appl. Mech. Engrg.*, 199(5-8):229–263, 2010.
6. Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. *Comput. Methods Appl. Mech. Engrg.*, 199(13–16):780–790, 2010.
7. Y. Bazilevs, Christian Michler, V. M. Calo, and T. J. R. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. *Comput. Methods Appl. Mech. Engrg.*, 199(13-16):780–790, 2010.
8. Pete Becker. Working draft, standard for programming language C++. Technical Report N3242=11-0012, ISO/IEC JTC 1, Information Technology, Subcommittee SC 22, Programming Language C++, February 2011.
9. D. Boffi, F. Brezzi, and M. Fortin. *Mixed Finite Element Methods and Applications.* Springer Series in Computational Mathematics. Springer London, Limited, 2013.
10. Daniele Boffi and Carlo Lovadina. Analysis of new augmented lagrangian formulations for mixed finite element schemes. *Numerische Mathematik*, 75(4):405–419, 1997.
11. Andrea Bressan and Giancarlo Sangalli. Isogeometric discretizations of the Stokes problem: stability analysis by the macroelement technique. *IMA J. Numer. Anal.*, 2012.
12. A. Buffa, D. Cho, and G. Sangalli. Linear independence of the T-spline blending functions associated with some particular T-meshes. *Comput. Methods Appl. Mech. Engrg.*, 199(23–24):1437–1445, 2010.

13. R.D. Cook. Improved two-dimensional finite element. *Journal of the Structural Division*, 100:1851–1863, 1974.
14. J. A. Cottrell, A. Reali, Y. Bazilevs, and T. J. R. Hughes. Isogeometric analysis of structural vibrations. *Comput. Methods Appl. Mech. Engrg.*, 195(41–43):5257–5296, 2006.
15. Tor Dokken, Tom Lyche, and Kjell Fredrik Pettersen. Locally refinable splines over box-partitions. Technical report, SINTEF, February 2012.
16. M. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with T-splines. *Comput. Methods Appl. Mech. Engrg.*, 199(5-8):264–275, 2009.
17. T. Elguedj, Y. Bazilevs, V. M. Calo, and T. J. R. Hughes. B-bar and f-bar projection methods for nearly incompressible linear and non-linear elasticity and plasticity based on higher-order nurbs elements. *Computer Methods in Applied Mechanics and Engineering*, 197:2732–2762, 2008.
18. H.C. Elman, D.J. Silvester, and A.J. Wathen. *Finite Elements and Fast Iterative Solvers : with Applications in Incompressible Fluid Dynamics: with Applications in Incompressible Fluid Dynamics*. Numerical Mathematics and Scientific Computation. OUP Oxford, 2005.
19. Howard C. Elman, David J. Silvester, Andrew, and Andrew J. Wathen. Performance and analysis of saddle point preconditioners for the discrete steady-state navier-stokes equations. *Numer. Math*, 90:665–688, 2000.
20. John A. Evans and Thomas J. R. Hughes. Isogeometric divergence-conforming B-spline for the steady Navier–Stokes equations. *Math. Mod. Meth. Appl. S.*, 23(08):1421–1478, 2013.
21. D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. In *SIGGRAPH '88 Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 205–212, 1988.
22. U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, December 1982.
23. C. Giannelli, B. Jüttler, and H. Speleers. THB–splines: The truncated basis for hierarchical splines. *Compute. Aided Geometric D.*, 29:485–498, 2012.
24. R. Glowinski, P. G. Ciarlet, and J. L. Lions. *Numerical Methods for Fluids*, volume 3 of *Handbook of numerical analysis*. Elsevier, 2002.
25. T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Engrg.*, 194(39-41):4135–4195, 2005.
26. igatools 0.3.0. An isogeometric analisys tool library - documentation and manual, October 2014.
27. ISO/IEC 14882:2011 – Information technology – programming languages – C++, 2011.
28. J. Kiendl, Y. Bazilevs, M.-C. Hsu, R. Wüchner, and K.-U. Bletzinger. Kirchhoff-Love shell structures comprised of multiple patches. *Comput. Methods Appl. Mech. Engrg.*, 199:2403–2416, 2010.
29. K. M. Mathisen, K. M. Okstad, T. Kvamsdal, and S. B. Raknes. Isogeometric analysis of finite deformation nearly incompressible solids. *Rakenteiden Mekaniikka (Journal of Structural Mechanics)*, 44(3):260–278, 2011.
30. M. S. Pauletti, M. Martinelli, N. Cavallini, and P. Antolin. Igatools: An isogeometric analysis library. *I.M.A.T.I.-C.N.R.*, pages 1–27, 2014.
31. L. A. Piegl and W. Tiller. *The NURBs Book*. Monographs in Visual Communication Series. Springer-Verlag GmbH, 1997.
32. Larry L. Schumaker. *Spline functions: basic theory*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, third edition, 2007.
33. M. A. Scott, X. Li, T. W. Sederberg, and T. J. R. Hughes. Local refinement of analysis-suitable T-splines. *Comput. Methods Appl. Mech. Engrg.*, 213216(0):206 – 222, 2012.
34. R. Taylor. Isogeometric analysis of nearly incompressible solids. *Int. J. Numer. Meth. Engng.*, 87(1-5):273288, 2010.
35. A.-V. Vuong, C. Giannelli, B. Jüttler, and B. Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Comput. Methods Appl. Mech. Engrg.*, 200:3554–3567, 2011 Dec.
36. Oliver Weeger, Utz Wever, and Bernd Simeon. Isogeometric analysis of nonlinear Euler Bernoulli beam vibrations. *Nonlinear Dynam.*, 72(4):813–835, 2013.
37. P. Wriggers. *Nonlinear Finite Element Methods*. Springer, 2008.