Struct Multidisc Optim (2016) 54:1345–1364
DOI 10.1007/s00158-014-1182-z

EDUCATIONAL ARTICLE

Fluid flow topology optimization in PolyTop: stability and computational implementation

Anderson Pereira · Cameron Talischi · Glaucio H. Paulino · Ivan F. M. Menezes · Marcio S. Carvalho

Received: 5 June 2013 / Revised: 21 June 2014 / Accepted: 22 June 2014 / Published online: 15 July 2016

Abstract We present a Matlab implementation of topology optimization for fluid flow problems in the educational computer code PolyTop (Talischi et al. 2012b). The underlying formulation is the well-established porosity approach of Borrvall and Petersson (2003), wherein a dissipative term is introduced to impede the flow in the solid (non-fluid) regions. Polygonal finite elements are used to obtain a stable low-order discretization of the governing Stokes equations for incompressible viscous flow. As a result, the same mesh represents the design field as well as the velocity and pressure fields that characterize its response. Owing to the modular structure of PolyTop, incorporating new physics, in this case modeling fluid flow, involves changes that are limited mainly to the analysis routine. We provide several numerical examples to illustrate the capabilities and use of the code. To illustrate the modularity of the present approach, we extend the implementation to accommodate alternative formulations and cost functions. These include topology optimization formulations where both viscosity and inverse permeability are functions of the design; and flow control where the velocity at a certain location in the domain is maximized in a prescribed direction.

Keywords Topology optimization · Polygonal finite elements · Matlab · Stokes flow

1 Introduction

This paper is a continuation of a series of educational computer codes written in Matlab for solving topology optimization problems (Talischi et al. 2012a, b). In Talischi et al. (2012b), we presented PolyTop which features a general finite element analysis routine using unstructured polygonal elements. The companion code PolyMesher (Talischi et al. 2012a) is a general-purpose Voronoi mesh generator that supplies polygonal meshes for use in PolyTop or in other applications. The general framework outlined in Talischi et al. (2012b) emphasizes a modular code structure where the analysis routine, including sensitivity calculations with respect to analysis parameters, and the optimization algorithm are kept separate from quantities defining the design field. This separation in turn permits changing the topology optimization formulation, including the choice of material interpolation scheme and the complexity control mechanism (e.g. filters and other manufacturing constraints), without the need for modifying the analysis function. In this paper, we illustrate how different physics and governing state equations can be implemented in PolyTop. This will highlight the benefits of adopting a modular structure because most of the changes in this case are related to the analysis function. For the problem of minimizing dissipated power in Stokes flow, which serves as

Electronic supplementary material The online version of this article (doi:10.1007/s00158-014-1182-z) contains supplementary material, which is available to authorized users.

A. Pereira · I. F. M. Menezes · M. S. Carvalho
Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rua Marques de Sao Vicente, 225, Rio de Janeiro, R.J. 22453, Brazil

C. Talischi · G. H. Paulino
Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign (UIUC), Urbana, IL 61801, USA

G. H. Paulino (✉)
School of School of Civil and Environmental Engineering, Georgia Institute of Technology, 790 Atlantic Drive, Atlanta, GA 30332, USA
e-mail: paulino@gatech.edu

the model problem here, the changes to the main `PolyTop` function, originally written for compliance minimization in elasticity, involve few lines of code. Since polygonal finite elements are again employed for the analysis, then the basis function construction and element integration routines also remain intact.

Another important feature of the present implementation is that the discretization of governing incompressible Stokes equations is carried out using a conceptually simple and computationally efficient low-order mixed finite element formulation, featuring piecewise constant approximations of the pressure field. The performance and stability of this formulation is addressed in a recent paper (Talischi et al. 2014) (see also, Beirão Da Veiga and Lipnikov 2010) where the Babuska-Brezzi conditions are shown to naturally hold for a large class of polygonal meshes where each vertex is at most incident on three edges. This topological requirement is met by "Voronoi" meshes generated from an initial random point set by `PolyMesher`.[1] It is well-known that low-order velocity and pressure pairs on triangular and quadrilateral meshes yield poor approximations of the velocity field (due to the so-called locking phenomenon) and pressure fields (due to the presence of spurious modes) and therefore may not be reliable for use in optimization. Intuitively, the stability of the corresponding polygonal discretizations can be attributed to the presence of more velocity degrees of freedom per pressure degree of freedom for elements with many sides. In this paper, we also employ a low-order discretization of the design field on the same polygonal mesh used to solve the governing boundary value problem. We note that previous works in the literature for fluid flow topology optimization have employed nested grids (Borrvall and Petersson 2003; Wiker et al. 2007), higher-order velocity and pressure pairs (Deng et al. 2011), and stabilization techniques (Guest and Prévost 2006; Challis and Guest 2009). The proposed discretization scheme produces fewer number of design and analysis unknowns for a given mesh resolution (such low-order discretizations are in general favored for topology optimization).

As mentioned above, the model problem considered here is that of minimization of viscous drag on an incompressible Newtonian fluid governed by the Stokes equations. We will focus on the topology optimization formulation proposed by Borrvall and Petersson (2003), sometimes known as the *porosity* approach, where a dissipative term is introduced to impede the flow in the solid (non-fluid) regions. This method has been previously used for topology optimization of stationary and transient Navier-Stokes flows (see Gersborg-Hansen et al. 2005; Deng et al. 2011; Kreissl et al. 2011b). There are alternative formulations in the literature in which viscosity is considered a function of the design in order to model a Darcy flow condition in the solid domain (Guest and Prévost 2006; Wiker et al. 2007). We will discuss the relatively minor changes to the code needed to accommodate this formulation and present a numerical example in this case. While we do not intend to give a complete survey of the literature in this brief introduction, we should mention level-set-based methods that address certain shortcomings of the porosity approach (e.g. pressure diffusion through the solid domain in high Reynold's number flow) (Kreissl et al. 2011a; Kreissl and Maute 2012) and use of Lattice Boltzmann theory to model the flow (Pingen et al. 2007).

The remainder of the paper is organized as follows: in the next section, we will discuss the model problem and its reformulation as a sizing optimization problem. The discretization of the state equation and design field is presented in Section 3. Next, we will detail the implementation of the resulting discrete problem in `PolyTop` in Section 4 and present several numerical results in Section 5. We conclude by discussing possible extensions of the code in Section 6.

2 Model problem

In this section, we will discuss the formulation of topology optimization in Stokes flow. As shown in Fig. 1, let $\Omega \subseteq \mathbb{R}^2$ denote the extended domain that contains all candidate designs and $\mathbf{g}$ a sufficiently regular velocity field on $\partial\Omega$ such that

$$\int_{\partial\Omega} \mathbf{g} \cdot \mathbf{n} ds = 0 \tag{1}$$

Moreover, let Γ denote the portion of $\partial\Omega$ where $\mathbf{g}$ is non-zero.

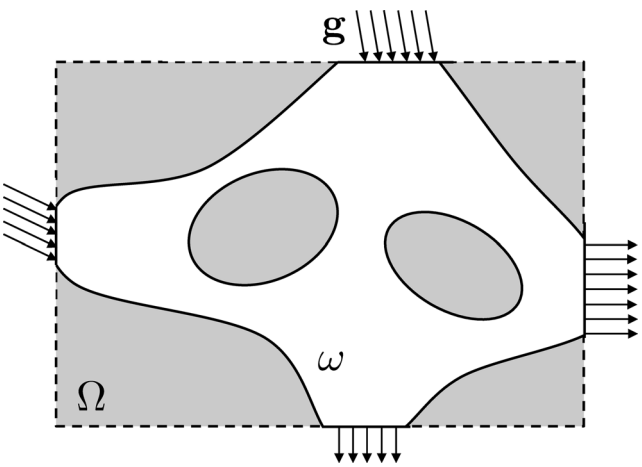


Fig. 1 Illustration of the extended domain Ω, prescribed boundary velocities $\mathbf{g}$ and an admissible shape ω

[1] Note, however, that not all Voronoi tessellations yield stable discretizations. For example, a structured quadrilateral mesh, corresponding to a Voronoi tessellation of a uniform grid of seeds, does not lead to a stable method.

We consider $\omega \subseteq \Omega$ to be an *admissible* shape if $\partial\omega \supseteq \Gamma$ and it is sufficiently smooth so that the incompressible Stokes equations

$$-2\text{div}\,[\mu\boldsymbol{\epsilon}(\mathbf{u})] + \nabla p = \mathbf{0} \quad \text{in } \omega \tag{2}$$

$$\text{div}\,\mathbf{u} = 0 \quad \text{in } \omega \tag{3}$$

$$\mathbf{u} = \mathbf{g} \quad \text{on } \partial\omega \cap \partial\Omega \tag{4}$$

$$\mathbf{u} = \mathbf{0} \quad \partial\omega \backslash \partial\Omega \tag{5}$$

is well-posed. In this expression, $\mathbf{u}$ and p are the velocity and pressure of the fluid, μ is its viscosity and $\boldsymbol{\epsilon}(\mathbf{u}) = \left(\nabla\mathbf{u} + \nabla^T\mathbf{u}\right)/2$ denotes the rate-of-strain tensor and we are assuming that the fluid has unit density. Observe that the fluid is subject to homogenous velocity boundary conditions on the "internal" boundary of ω, i.e., $\partial\omega\backslash\partial\Omega$. We will denote the unique solution of (2)–(5) by $(\mathbf{u}_\omega, p_\omega)$ to indicate its dependence on the shape ω.

A benchmark optimal shape problem consists of finding an admissible shape ω that minimizes the dissipated power subject to a constraint on its volume. In particular, we seek to minimize the cost functional:

$$\frac{1}{2}\int_\omega \mu\boldsymbol{\epsilon}(\mathbf{u}_\omega) : \boldsymbol{\epsilon}(\mathbf{u}_\omega)\mathrm{d}\mathbf{x} \tag{6}$$

subject to the constraint[2]

$$|\omega| \leq \overline{v}\,|\Omega| \tag{7}$$

where $\overline{v}$ is the prescribed upper bound on the volume fraction. It can be shown that minimizing this objective function amounts to minimizing the average pressure drop between the inlet and the outlet (Borrvall and Petersson 2003).

Similar to our treatment of optimal shape problems in elasticity in Talischi et al. (2012b), we follow a two-step procedure to reformulate this problem into one of sizing optimization with the design field defined on the entire extended domain Ω. The first step consists of approximating the governing state equation with a boundary value problem defined on Ω in analogous manner to the so-called Ersatz approach in elasticity. For a fixed shape ω and sufficiently small parameter $\varepsilon > 0$, one can show that the solution $\left(\mathbf{u}^\varepsilon_\omega, p^\varepsilon_\omega\right)$ to the generalized Stokes problem given by

$$-2\text{div}\,[\mu\boldsymbol{\epsilon}(\mathbf{u})] + \frac{1}{\varepsilon}(1-\chi_\omega)\,\mathbf{u} + \nabla p = \mathbf{0} \quad \text{in } \Omega \tag{8}$$

$$\text{div}\,\mathbf{u} = 0 \quad \text{in } \Omega \tag{9}$$

$$\mathbf{u} = \mathbf{g} \quad \text{on } \partial\Omega \tag{10}$$

when restricted to ω, is a good approximation to $(\mathbf{u}_\omega, p_\omega)$ (Evgrafov 2005). Here χ_ω denotes the characteristic function associated with ω (so $\chi_\omega(\mathbf{x}) = 1$ when $\mathbf{x} \in \omega$ and $\chi_\omega(\mathbf{x}) = 0$ otherwise). Intuitively, the new dissipative term $(1-\chi_\omega)\,\mathbf{u}/\varepsilon$ in this generalized Stokes system forces $\mathbf{u}^\varepsilon_\omega$ to vanish in $\Omega\backslash\omega$ when the "permeability" coefficient ε is close to zero. In contrast to the Ersatz approximation in elasticity, observe that the viscous term containing the highest order derivatives is left intact in this approximation.

As a surrogate to the minimization of (6), we fix $0 < \varepsilon \ll 1$ and consider the following optimization problem

$$\min_{\omega\subseteq\Omega} \frac{1}{2}\int_\Omega \left[\mu\boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_\omega) : \boldsymbol{\epsilon}(\mathbf{u}^\varepsilon_\omega) + \frac{1}{\varepsilon}(1-\chi_\omega)\,\mathbf{u}^\varepsilon_\omega \cdot \mathbf{u}^\varepsilon_\omega\right]\mathrm{d}\mathbf{x}$$

$$\text{subject to} \quad \frac{1}{|\Omega|}\int_\Omega \chi_\omega \mathrm{d}\mathbf{x} \leq \overline{v} \tag{11}$$

A few remarks are in order regarding the optimization problem (11). Unlike the related compliance minimization problem in elasticity, (11) is well-posed in that it admits minimizers when the space of admissible shapes consists of all measurable subsets of Ω (Borrvall and Petersson 2003). Therefore, there is no need to impose additional regularity on the characteristic functions representing the shapes (e.g. impose a bound on the perimeter). This is, in part, a consequence of the fact that the design field appears in the coefficient of the lower order term in the governing state equation (8). On physical grounds, the favorability of highly oscillatory shapes in the compliance problem is related to the stiffening effect of fine mixtures (Allaire 2001). By contrast, such arrangements in the fluid flow problem lead to larger viscous dissipation and are naturally excluded in the optimal regime. Moreover, we note that the mere fact that (8)–(10) is an approximation to (2)–(5) by itself does not justify the approximation of optimization problem (6) by (11). However, the analysis of Evgrafov (2005) establishes that the sequence of optimal solutions to (11), as $\varepsilon \to 0$, admits a limit point that is a minimizer of (6).

Since only the characteristic function associated with ω appears in (11), we have effectively obtained an optimization problem over the space of measurable binary functions $L^\infty(\Omega; \{0, 1\})$. The second step in deriving a sizing optimization problem is to replace this space by the larger space $L^\infty(\Omega; [0, 1])$ consisting of functions ρ that take any value in the interval $[0, 1]$. Interestingly, this relaxation or enlargement step does not require any additional modification of the problem (e.g., introducing explicit penalization of intermediate fields) since one can show that there exists a characteristic function minimizing J (objective function) in the larger space $L^\infty(\Omega; [0, 1])$ (Borrvall and Petersson 2003; Evgrafov 2005). This result implies that enlarging the space of admissible designs does not change the value of the optimization problem even though it makes the problem amenable to gradient-based optimization algorithms.

The final sizing optimization problem is thus given by

$$\min_{\rho\in\mathcal{A}} \frac{1}{2}\int_\Omega \left[\mu\boldsymbol{\epsilon}(\mathbf{u}_\rho) : \boldsymbol{\epsilon}(\mathbf{u}_\rho) + m_\alpha(\rho)\mathbf{u}_\rho \cdot \mathbf{u}_\rho\right]\mathrm{d}\mathbf{x}$$

$$\text{subject to} \quad \frac{1}{|\Omega|}\int_\Omega m_V(\rho)\mathrm{d}\mathbf{x} \leq \overline{v} \tag{12}$$

[2] The area/volume of the set A is denoted by $|A|$.

where $\mathcal{A} = L^\infty(\Omega; [0, 1])$ and, with a slight abuse of notation, $(\mathbf{u}_\rho, p_\rho)$ is used to denote the solution to

$$-2\,\mathrm{div}\,[\mu\boldsymbol{\epsilon}(\mathbf{u})] + m_\alpha(\rho)\mathbf{u} + \nabla p = \mathbf{0} \quad \text{in } \Omega \tag{13}$$

$$\mathrm{div}\,\mathbf{u} = 0 \quad \text{in } \Omega \tag{14}$$

subject to the boundary conditions (10). In the above expression, we have defined the inverse permeability coefficient

$$m_\alpha(\rho) = \frac{1}{\varepsilon}(1-\rho) \tag{15}$$

and the interpolation function for the volume term is simply $m_V(\rho) = \rho$. We remark that Borrvall and Petersson (2003) arrive at the same sizing problem in two dimensions with

$$m_\alpha(\rho) = \frac{5\mu}{2\rho^2} \tag{16}$$

by considering the three-dimensional Stokes problem under plane flow assumptions wherein ρ represents the distance between two encapsulating plates. The dissipative term $m_\alpha(\rho)\mathbf{u}$ in that setting is due to the out-of-plane shear effects. Naturally, the fluid movement is restricted in the regions where ρ is close to zero. Also, in this context, $\int_\Omega m_V(\rho)\mathrm{d}\mathbf{x}$ is the total volume of the fluid in the domain, and so their derivation gives some physical meaning to the design field ρ. The authors, however, recognize that the sizing formulation can also be used for three-dimensional problems and with an interpolation function for $m_\alpha(\rho)$ that is more suited for numerical calculations. In fact, they advocate the use of

$$m_\alpha(\rho) = \frac{\mu}{\varepsilon}\frac{q\,(1-\rho)}{q+\rho} \tag{17}$$

along with a continuation procedure consisting of gradually increasing the parameter $q > 0$. It is easy to see that this rational function approaches the linear function (15) as $q \to \infty$. Also, referring to (16), ε is set to 4×10^{-5} which corresponds to minimum distance of 0.01 between the encapsulating plates. We note that the authors in Borrvall and Petersson (2003) also use a small positive lower bound on $m_\alpha(\rho)$ at $\rho = 1$ but this is not necessary either from a theoretical or a computational point of view. Observe that (17) coincides with equation (40) of Borrvall and Petersson (2003) if we set $\underline{\alpha} = 0$ and $\overline{\alpha} = \mu/\varepsilon$ in the latter.

3 Discretization

In this section, we derive the discrete optimization problem using a finite element discretization of the state equation and the design space $\mathcal{A}$. First, let us recall that the weak form of (13)–(14) and (10) consists of finding $(\mathbf{u}, p) \in \mathcal{U} \times \mathcal{Q}$ such that

$$2\int_\Omega \mu\boldsymbol{\epsilon}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{v})\mathrm{d}\mathbf{x} + \int_\Omega m_\alpha(\rho)\mathbf{u}\cdot\mathbf{v}\mathrm{d}\mathbf{x} - \int_\Omega p\,\mathrm{div}\,\mathbf{v}\mathrm{d}\mathbf{x} = 0, \quad \forall \mathbf{v} \in \mathcal{V} \tag{18}$$

$$-\int_\Omega q\,\mathrm{div}\,\mathbf{u}\mathrm{d}\mathbf{x} + \lambda\int_\Omega q\mathrm{d}\mathbf{x} = 0, \quad \forall q \in \mathcal{Q} \tag{19}$$

$$\int_\Omega p\mathrm{d}\mathbf{x} = 0 \tag{20}$$

where the velocity trial and test spaces are given by

$$\mathcal{U} = \left\{\mathbf{u} \in \left[H^1(\Omega)\right]^2 : \mathbf{u}|_{\partial\Omega} = \mathbf{g}\right\},$$

$$\mathcal{V} = \left\{\mathbf{v} \in \left[H^1(\Omega)\right]^2 : \mathbf{v}|_{\partial\Omega} = \mathbf{0}\right\} \tag{21}$$

and the pressure space is $\mathcal{Q} = L^2(\Omega)$. Observe that the pressure is determined only up to a constant in the boundary value problem defined by (13), (14), and (10).[3] In order to uniquely define the pressure field, a zero mean condition (20) is enforced in the weak statement above. The constant λ in (19) is the Lagrange multiplier associated with this constraint (we refer the reader to Bochev and Lehoucq 2001 for a more general discussion of enforcing such constraints).

Given a partition $\mathcal{T}_h = \{\Omega_\ell\}_{\ell=1}^N$ of Ω consisting of polygons, the Galerkin approximation of this system is obtained by restricting the variational problem to finite dimensional subspaces $\mathcal{U}_h$, $\mathcal{V}_h$ and $\mathcal{Q}_h$. Following Talischi et al. (2014), within each element, we take a constant pressure field while the velocity field is represented by an isoparametric transformation of the Wachspress basis, similar to what is used for discretization of the displacements in Talischi et al. (2012b). Note that, with this choice, each $p \in \mathcal{Q}_h$ can be written as

$$p(\mathbf{x}) = \sum_{\ell=1}^N P_\ell \chi_{\Omega_\ell}(\mathbf{x}) \tag{22}$$

for some vector $\mathbf{P} = (P_\ell)_{\ell=1}^N$ of element pressures[4] and the integral in constraint (20) can be computed as

$$\int_\Omega p\mathrm{d}\mathbf{x} = \sum_{\ell=1}^N P_\ell \int_\Omega \chi_{\Omega_\ell}\mathrm{d}\mathbf{x} = \sum_{\ell=1}^N P_\ell\,|\Omega_\ell| = \mathbf{a}^T\mathbf{P} \tag{23}$$

where $\mathbf{a} = (|\Omega|_\ell)_{\ell=1}^N$ is the vector of element areas.

Let us assume that $\mathbf{u} \in \mathcal{U}_h$ has an expansion of the form

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^M U_i \mathbf{N}_i(\mathbf{x}) \tag{24}$$

[3] It can also be seen from the weak form that terms $\int_\Omega p\,\mathrm{div}\,\mathbf{v}\mathrm{d}\mathbf{x}$ and $\int_\Omega q\,\mathrm{div}\,\mathbf{u}\mathrm{d}\mathbf{x}$ are not affected by the addition or subtraction of a constant to p or q. Indeed, for $q \equiv c$, a constant, and $\mathbf{v} \in \mathcal{U}$ or $\mathbf{v} \in \mathcal{V}$, we have $\int_\Omega q\,\mathrm{div}\,\mathbf{v}\mathrm{d}\mathbf{x} = c\int_\Omega \mathbf{v}\cdot\mathbf{n}\mathrm{d}s = 0$.

[4] In Talischi et al. (2012b), $\mathbf{P}$ is used to denote the filtering matrix. In this paper, we will not use filtering so no confusion should arise.

for some vector $\mathbf{U} = (U_i)_{i=1}^{M}$. Here $\{\mathbf{N}_i\}_{i=1}^{M}$ denotes the velocity basis functions associated with *all* the nodes in the mesh $\mathcal{T}_h$. To respect the velocity boundary conditions imposed on the flow, the degrees of freedom associated with the nodes on $\partial\Omega$ must be set to the prescribed value $\mathbf{g}$. To facilitate the notation, let $\mathbf{M}_\mathsf{p}$ denote the matrix that extracts these degrees of freedom and $\mathbf{G}$ be the vector of prescribed velocities on $\partial\Omega$. Thus we must have

$$\mathbf{M}_\mathsf{p}\mathbf{U} = \mathbf{G} \tag{25}$$

to enforce $\mathbf{u}|_{\partial\Omega} = \mathbf{g}$. Similarly, if $\mathbf{v} \in \mathcal{V}_h$ has the expansion $\mathbf{v}(\mathbf{x}) = \sum_{i=1}^{M} V_i \mathbf{N}_i(\mathbf{x})$, then $\mathbf{M}_\mathsf{p}\mathbf{V} = \mathbf{0}$ corresponds to $\mathbf{v}|_{\partial\Omega} = \mathbf{0}$. For later use, we define $\mathbf{M}_\mathsf{f}$ to denote the matrix that extracts the remaining *free* degrees of freedom.[5]

As mentioned before, we will also use a piecewise constant approximation of the design field and take the discrete space of admissible designs as

$$\mathcal{A}_h = \left\{\rho : \Omega \to \mathbb{R} : 0 \le \rho \le 1,\ \rho|_{\Omega_\ell} = \text{const},\ \forall \ell = 1, \ldots, N\right\} \tag{26}$$

To obtain a matrix-vector representation of the Galerkin approximation of (18)–(20), we first define local (elemental) matrices

$$\left[\mathbf{A}_\ell^{\mu}\right]_{ij} = \int_{\Omega_\ell} 2\mu\boldsymbol{\epsilon}(\mathbf{N}_i) : \boldsymbol{\epsilon}(\mathbf{N}_j)\mathrm{d}\mathbf{x}, \qquad \left[\mathbf{A}_\ell^{\alpha}\right]_{ij} = \int_{\Omega_\ell} \mathbf{N}_i \cdot \mathbf{N}_j \mathrm{d}\mathbf{x} \tag{27}$$

If $\rho = \sum_{\ell=1}^{N} z_\ell \chi_{\Omega_\ell}$, with elemental design variables $\mathbf{z} = (z_\ell)_{\ell=1}^{N}$, then the global matrices corresponding to the first two bilinear forms in (18) are given by

$$\mathbf{A}^{\mu} = \sum_{\ell=1}^{N} \mathbf{A}_\ell^{\mu}, \qquad \mathbf{A}^{\alpha}(\mathbf{E}) = \sum_{\ell=1}^{N} E_\ell \mathbf{A}_\ell^{\alpha}, \qquad E_\ell = m_\alpha(z_\ell) \tag{28}$$

Observe that $\mathbf{A}^{\mu}$ does not depend on the design variables $\mathbf{z}$. Also, we have defined a new vector $\mathbf{E}$ of elemental inverse permeabilites to facilitate the modular implementation of the analysis routine in the code. The matrix associated with the bilinear form in (19) is given by

$$[\mathbf{B}]_{i\ell} = -\int_{\Omega} \chi_{\Omega_\ell} \operatorname{div} \mathbf{N}_i \mathrm{d}\mathbf{x} = -\int_{\Omega_\ell} \operatorname{div} \mathbf{N}_i \mathrm{d}\mathbf{x} \tag{29}$$

and is also independent of the design.

[5] In Matlab, if `FreeDofs` and `PresDofs` are arrays containing the indices of free and prescribed degrees of freedom, then $\mathbf{M}_\mathsf{f}\mathbf{U}$ and $\mathbf{M}_\mathsf{p}\mathbf{U}$ are simply obtained by `U(FreeDofs,:)` and `U(PresDofs,:)`.

The Galerkin approximation of (18)–(20) can now be written as

$$\begin{aligned} \mathbf{V}^T\left[\mathbf{A}^{\mu} + \mathbf{A}^{\alpha}(m_\alpha(\mathbf{z}))\right]\mathbf{U} + \mathbf{V}^T\mathbf{B}\mathbf{P} = 0, \quad & \forall \mathbf{V} \text{ such that } \mathbf{M}_\mathsf{p}\mathbf{V} = \mathbf{0} \\ \mathbf{U}^T\mathbf{B}\mathbf{Q} + \lambda\mathbf{a}^T\mathbf{Q} = 0, \quad & \forall \mathbf{Q} \\ \mathbf{a}^T\mathbf{P} = 0 & \end{aligned} \tag{30}$$

with $\mathbf{U}$ satisfying $\mathbf{M}_\mathsf{p}\mathbf{U} = \mathbf{G}$. Noting that $\mathbf{V}$ has the form $\mathbf{M}_\mathsf{f}^T\tilde{\mathbf{V}}$ for an arbitrary vector $\tilde{\mathbf{V}}$, this variational problem is equivalent to the following linear systems of equations:

$$\begin{aligned} \mathbf{M}_\mathsf{f}\left[\mathbf{A}^{\mu} + \mathbf{A}^{\alpha}(m_\alpha(\mathbf{z}))\right]\mathbf{U} + \mathbf{M}_\mathsf{f}\mathbf{B}\mathbf{P} &= \mathbf{0} \\ \mathbf{B}^T\mathbf{U} + \lambda\mathbf{a} &= \mathbf{0} \\ \mathbf{a}^T\mathbf{P} &= 0 \\ \mathbf{M}_\mathsf{p}\mathbf{U} &= \mathbf{G} \end{aligned} \tag{31}$$

The discrete form of the sizing optimization problem (14) is thus given by

$$\min_{\mathbf{z}\in[0,1]^N} \frac{1}{2}\mathbf{U}^T\left[\mathbf{A}^{\mu} + \mathbf{A}^{\alpha}(m_\alpha(\mathbf{z}))\right]\mathbf{U} \quad \text{subject to} \quad \frac{\mathbf{a}^T m_V(\mathbf{z})}{\mathbf{a}^T\mathbf{1}} \le \overline{v} \tag{32}$$

where $\mathbf{U}$ is the solution to (31), $\mathbf{a} = (|\Omega|_\ell)_{\ell=1}^{N}$ and $\mathbf{1}$ denotes an array of size N consisting of unit entries.

We conclude this section with a brief study of the accuracy of polygonal finite elements for solving flow problems and compare their performance with popular stable mixed formulations. To this effect, we consider the problem of incompressible flow in an L-shaped domain governed by Stokes equations. The flow behavior, in particular the singularity of the pressure and velocity gradients, are representative of the behavior typically observed at re-entrant corners. The analytical solution for this problem is available in the literature (see, for example, Talischi et al. 2014) and is used to compute the error in the finite element solutions. For the purposes of comparison, we have considered the triangular MINI elements, quadrilateral Q2Q1 (also known as Taylor-Hood) and Q2P1 (sometimes referred to as Crouzeix-Raviart) elements, and finally the Q1Q1M macroelements used in Borrvall and Petersson (2003). A description of these elements and the representative mesh geometries are shown in Figs. 2 and 3, respectively. For the polygonal CVT meshes, the results reflect the average of 10 meshes generated by `PolyMesher`.

In order to compare the performance of these different discretizations, we have plotted three measures of error against the total number of degrees of freedoms (DOFs), which reflects the size of the global linear system and thus correlates with the cost of solving it. We note that the differences in cost of element level calculations (e.g., shape function computation and numerical integration) are

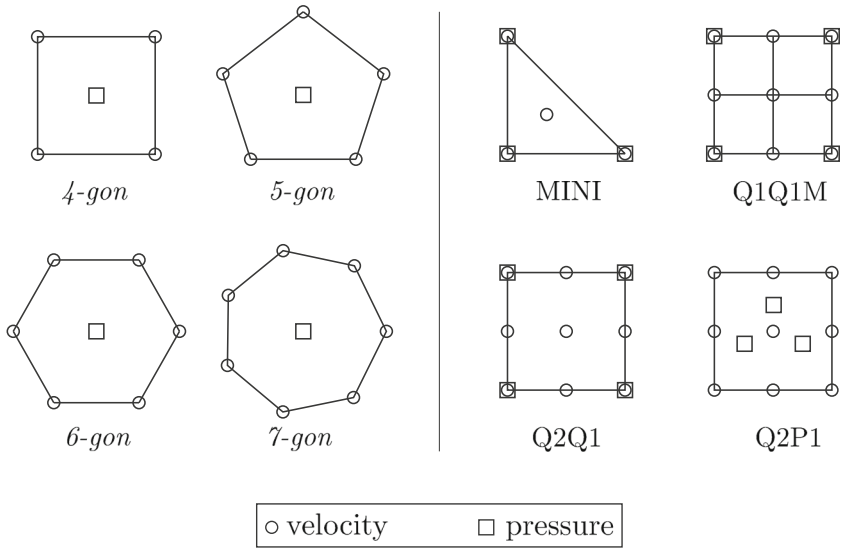


Fig. 2 Description of finite element formulations used for the comparative study: polygonal elements feature constant discontinuous pressure field and continuous velocity field spanned by iso-parametric Wachspress shape functions; the MINI element uses continuous linear pressure field and continuous linear velocity field enriched with an internal cubic bubble function; the macroelement Q1QM formulation uses continuous bilinear velocity and pressure fields on staggered grids; the Q2Q1 element features continuous bilinear pressure and biquadratic velocity fields; the Q2P1 formulation utilizes continuous biquadratic velocities and discontinuous linear pressures

not accounted for in this study, though as discussed in Talischi et al. (2012b), the cost of solving of the linear systems constitutes the major portion of overall computational effort involved in the repeated analyses in topology optimization. The choice of discretization also affects the structure of the global linear system, which can be an important factor if iterative solvers are used.

We can see from the results in Fig. 4 that all formulations exhibit the same convergence rates for this problem.[6] In all cases, the Q2P1 provides the most accurate solutions for a given number of DOFs. Polygonal elements yield similar accuracy in velocity and rate of strain as the Q2Q1 and the macroelement Q1Q1M though they outperform them in pressure accuracy. Finally, polygonal elements can be significantly more accurate than the MINI elements.

4 Implementation in `PolyTop`

We proceed to discuss the implementation of the model problem in `PolyTop`. As mentioned before, owing to the modular structure of the code, the main changes are limited to the analysis routines and parameters, more specifically in the definition of `fem` structure, `ObjectiveFnc` and `FEAnalysis` functions. The structure `opt`, containing the parameters related to optimization, and the `UpdateScheme`, featuring the optimality criteria method, are the same as before, though the design variables `z` and material interpolation function `opt.MatIntFnc` correspond to those in problem (32). Also, since there is no need for filtering, the regularization matrix `opt.P` is set to the identity matrix. Recall that in this case, the intermediate variable `y` coincides with `z`.

Table 1 provides a summary of the fields introduced in the `fem` structure. The implementation of the `FEAnalysis` function follows the same philosophy as the one for elasticity in Talischi et al. (2012b). The input to the function is the array of inverse permeabilities $\mathbf{E} = m_\alpha(\mathbf{z})$ denoted by `E`. During the first call to this function, the `fem` structure is updated and the invariant quantities such as the index arrays and elemental matrices are computed once and for all (see lines 62-94). For example, the vector of prescribed degrees of freedom, `fem.FixedDofs`, as well as the boundary velocities `G` are computed once on lines 90–91.

The same sparse assembly approach is adopted on lines 95-96 to assemble the matrices $\mathbf{A} := \mathbf{A}^\mu + \mathbf{A}^\alpha$ and $\mathbf{B}$ as defined in the previous section. The elemental matrices are computed in function `LocalK` according to (27) and (29) (for more details on the implementation, we refer the reader to the text Donea and Huerta 2003).

To assemble the linear system (31), we first define global matrix and vector

$$\mathbf{K} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{B}^T & \mathbf{0} & \mathbf{a} \\ \mathbf{0} & \mathbf{a}^T & 0 \end{bmatrix}, \qquad \mathbf{S} = \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \\ \lambda \end{bmatrix} \tag{33}$$

Then the usual process of eliminating the prescribed degrees of freedom and computing the remaining unknowns is applied to $\mathbf{K}$ and $\mathbf{S}$ (see lines 101–102). The connection with

[6] The strength of the singularity of the exact solution governs the convergence rates here.

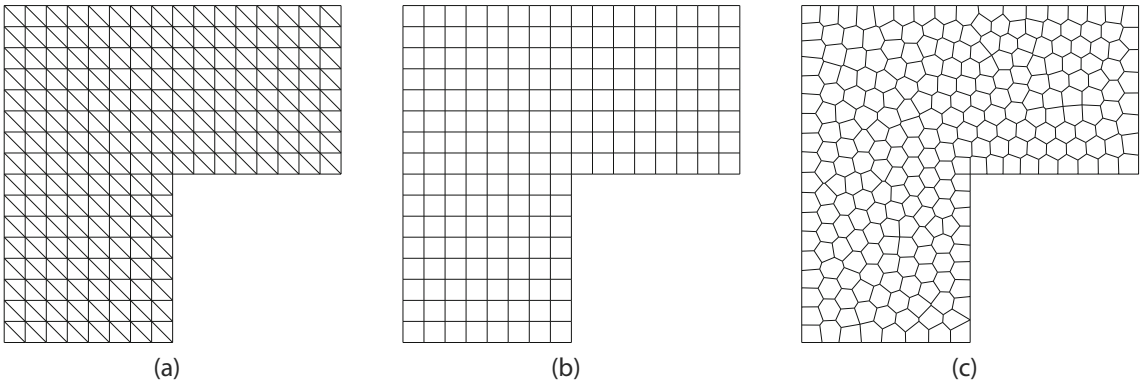


Fig. 3 Representative example of the family of meshes for the L-shaped problem **a** uniform triangular **b** uniform quadrilateral **c** centroidal Voronoi (CVT) generated by PolyMesher

(31) can be seen by noting that the velocity vector can be written as

$$\mathbf{U} = \mathbf{M}_{\mathrm{f}}^T \tilde{\mathbf{U}} + \mathbf{M}_{\mathrm{p}}^T \mathbf{G} \tag{34}$$

where $\tilde{\mathbf{U}}$ denotes the free velocity degrees of freedom. Upon substitution in (31), we obtain the following linear system for $\tilde{\mathbf{U}}$, $\mathbf{P}$ and the Lagrange multiplier λ:

$$\begin{bmatrix} \mathbf{M}_{\mathrm{f}}\mathbf{A}\mathbf{M}_{\mathrm{f}}^T & \mathbf{M}_{\mathrm{f}}\mathbf{B} & \mathbf{0} \\ \mathbf{B}^T\mathbf{M}_{\mathrm{f}}^T & \mathbf{0} & \mathbf{a} \\ \mathbf{0} & \mathbf{a}^T & 0 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{U}} \\ \mathbf{P} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{M}_{\mathrm{f}}\mathbf{A}\mathbf{M}_{\mathrm{p}}^T\mathbf{G} \\ -\mathbf{B}^T\mathbf{M}_{\mathrm{p}}^T\mathbf{G} \\ 0 \end{bmatrix} \tag{35}$$

The FEAnalysis function outputs the computed velocity vector $\mathbf{U}$ and the array $\mathbf{F} = [\mathbf{A}^{\mu} + \mathbf{A}^{\alpha}(\mathbf{E})]\,\mathbf{U}$. The objective function (32) is then computed on line 29 as

$$f = \frac{1}{2}\mathbf{U}^T\left[\mathbf{A}^{\mu} + \mathbf{A}^{\alpha}(\mathbf{E})\right]\mathbf{U} = \frac{1}{2}\mathbf{F}^T\mathbf{U} \tag{36}$$

The sensitivity of f with respect to perturbations in the elemental inverse permeability E_ℓ is given by (cf. Wiker et al. 2007 and (28))

$$\frac{\partial f}{\partial E_\ell} = \frac{1}{2}\mathbf{U}^T\frac{\partial \mathbf{A}^{\alpha}}{\partial E_\ell}\mathbf{U} = \frac{1}{2}\mathbf{U}^T\mathbf{A}_\ell^{\alpha}\mathbf{U} \tag{37}$$

and computed in ObjectiveFnc on lines 27–33. These sensitivities along with the use of the chain rule are used to compute the sensitivity of f with respect to the design variable $\mathbf{z}$ in the main function on line 17.

As in Talischi et al. (2012b), the code is run via the Matlab script PolyScript, wherein a call is first made to PolyMesher to generate the mesh of the problem domain, the structures fem and opt are initialized, and PolyTop is executed inside a continuation loop on the parameter q associated with the material interpolation function m_α. The representative implementation of this script is included in the supplementary material (also see Appendix B).

5 Numerical examples

We proceed to present several numerical examples for benchmark problems in drag minimization. Unless otherwise stated, the viscosity of the fluid μ is set to unity and the small parameter ε in (17) is fixed at 4×10^{-5}. For each value of parameter q, the tolerance for the change in design variables is taken to be 1 % and a maximum of 150 iterations

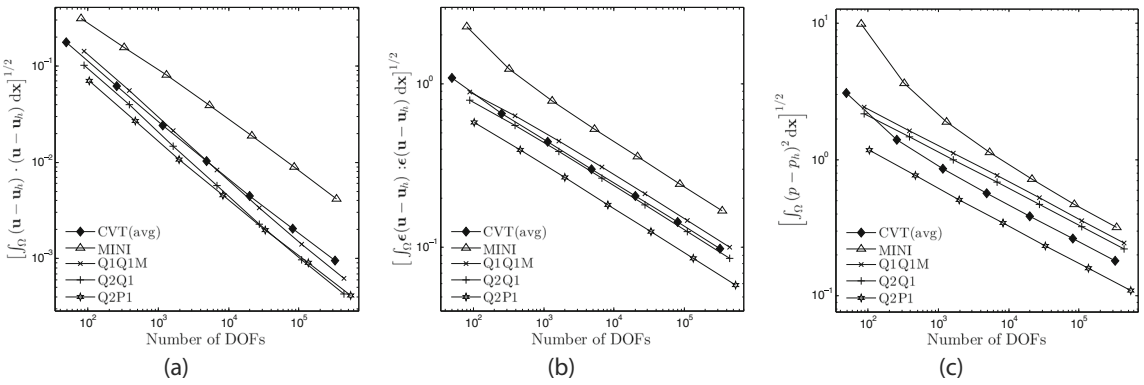


Fig. 4 Plots of error vs the number of DOFs for the L-shaped problem. Here $(\mathbf{u}, p)$ and $(\mathbf{u}_h, p_h)$ denote the exact and computed solutions, respectively **a** the error in the velocity field **b** the error in the rate of strain **c** the error in pressure

Table 1 List of fields in the `fem` structures

`fem` field	Definition
`fem.NNode`	Number of nodes
`fem.NElem`	Number of elements
`fem.Node`	[`NNode` × 2] array of nodes
`fem.Element`	[`NElem` × Var] cell array of elements
`fem.NodeBC`	[`NNodeBC` × 3] array of velocity boundary conditions
`fem.mu0`	Fluid viscosity
`fem.Reg`	Tag for regular meshes
`fem.ElemArea`†	Array of element areas
`fem.ElemNDofA`†	Array showing number of velocity DOFs of elements
`fem.ShapeFnc`†	Cell array with tabulated shape functions & weights
`fem.kAmu`†	Array of local stiffness matrix entries associated with $\mathbf{A}^{\mu}$
`fem.iA`†	Index array for sparse assembly of `fem.kAmu`
`fem.jA`†	Index array for sparse assembly of `fem.kAmu`
`fem.e`†	Array of element IDs corresponding to `fem.kAmu`
`fem.kAalpha`†	Array of local stiffness matrix entries associated with $\mathbf{A}^{\alpha}$
`fem.iB`†	Index array for sparse assembly of `fem.kB`
`fem.jB`†	Index array for sparse assembly of `fem.kB`
`fem.kB`†	Array of local stiffness matrix entries associated with $\mathbf{B}$
`fem.NDof`†	Number of degrees of freedom
`fem.FixedDofs`†	Array of prescribed degrees of freedom
`fem.FreeDofs`†	Array of free degrees of freedom
`fem.G`†	Array of boundary velocities

The fields marked with the superscript †, if empty, are populated inside `PolyTop`

is allowed. We will consider two continuation schemes for q. The "default" scheme uses $q = 0.01$ as the initial value, followed by an increase to $q = 0.1$ and $q = 1$. In the "conservative" scheme, smaller initial values of q are used which leads to a better-behaved problem in the early iterations. More specifically, we set to $q = 10^k$ for $k = -6, -5, \ldots, 0$. The initial guess for all the examples is $\rho \equiv \overline{v}$.

5.1 Diffuser

The first example is the diffuser problem with two different choices of extended design domain shown in Fig. 5. The inlet and outlet velocities have the form $\mathbf{g} = g\mathbf{n}$ where $\mathbf{n}$ is the outward unit normal to the boundary and g is a parabolic function defined such that $\int_{\Gamma_{\text{in}}} g\mathrm{d}s = \int_{\Gamma_{\text{out}}} g\mathrm{d}s$. Zero velocities are imposed on the remaining portion of the boundary $\partial\Omega\setminus(\Gamma_{\text{in}} \cup \Gamma_{\text{out}})$. The results for meshes consisting of 5,000 polygonal elements are shown in this figure. The solution for the curved domain is computed with volume fraction $\overline{v} = 0.46085$ to match the solution for the square domain based on $\overline{v} = 0.5$, as prescribed in Borrvall and Petersson (2003). Note that the design field ρ is plotted in greyscale where $\rho = 0$ is shown in black (corresponding to the solid region where flow is restricted) and $\rho = 1$ is indicated in white (corresponding to the fluid region). For this problem, the parameter q was adapted according to the default continuation procedure. The evolution of the design on the curved domain is shown in Fig. 6 by plotting the design field after convergence is obtained for each value of q. As predicted by the theory, the solution is almost completely binary though for $q = 1$ the outline of corresponding shape is similar to that of $q = 0.1$.

The pressure and velocity fields for the final solutions are also shown in Fig. 5. We can see that the flow velocity is small in the region where $\rho = 1$ and thus the zero flow condition (5) along the boundary of the solution is well represented. Moreover, the pressure field is smooth and free of any spurious oscillations. Note, however, the pressure field in the non-fluid region is not the physical pressure that solid material experiences but merely the approximate solution to the generalized Stokes system (13)–(14). Finally, we remark that the optimal solution on the curved domain is similar to the one obtained on the rectangular domain despite the difference in the choice of the extended design domain Ω. This is reasonable because the optimal channel is contained in both domains

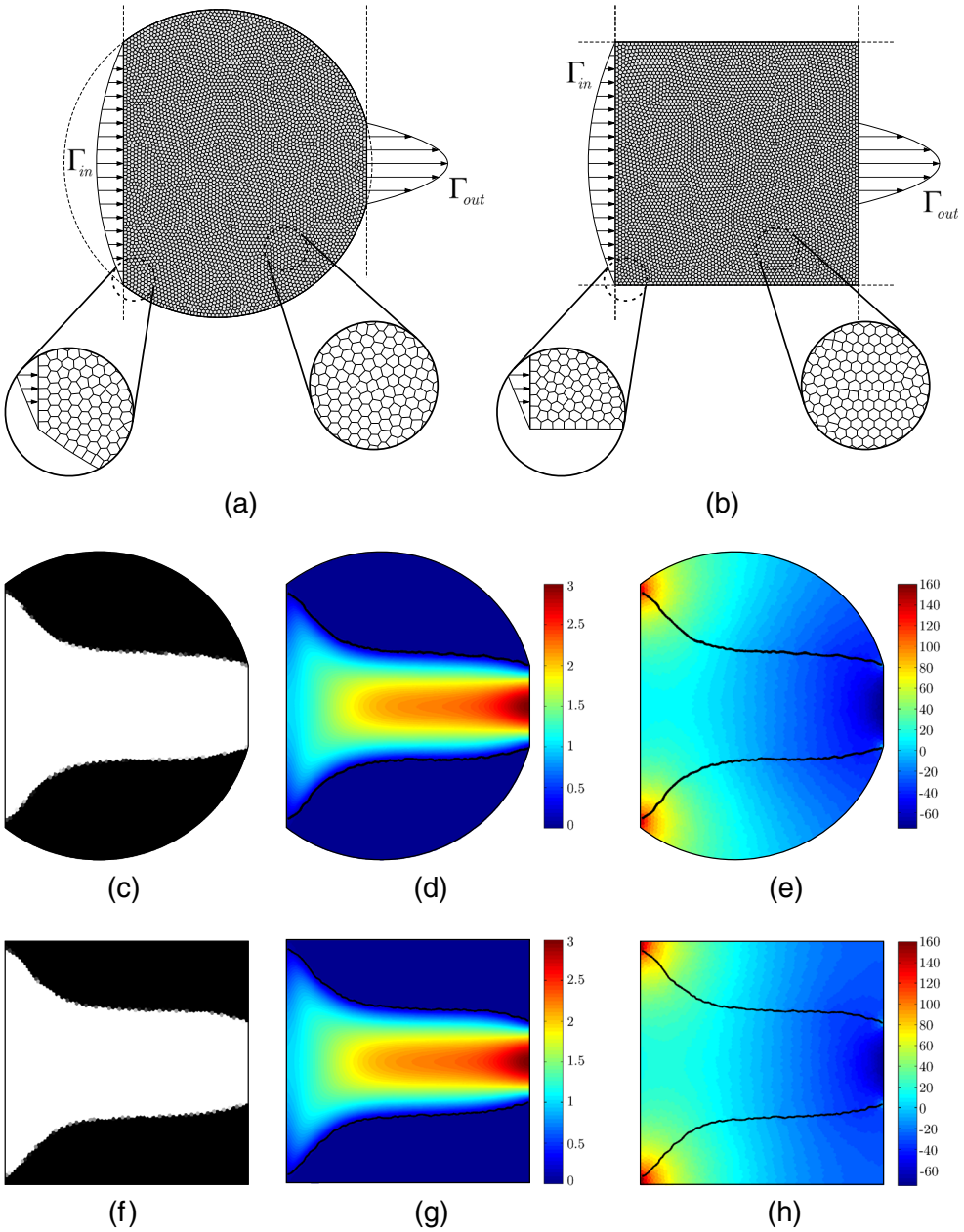


Fig. 5 Solution to the diffuser problem for two different choices of the extended design domain **a**, **b** geometry and boundary conditions for each domain **c**, **f** corresponding optimal solutions **d**, **g** velocity fields and **e**, **h** pressure fields. The meshes used are composed of 5,000 elements

and the flow is essentially subject to zero boundary conditions along the boundary of the channel. Table 2 also contains a comparison with the solutions presented by Borrvall and Petersson (2003) for this problem.

5.2 Double pipe

The next example is the double pipe problem which consists of two velocity inlets and outlets in a serpentine domain as shown in Fig. 7. The flow velocity profile is again parabolic with unit amplitude and orthogonal to the domain boundary. In order to specify the exact location of the inlets and outlets, `PolyMesher` is slightly modified to allow for prescribing vertex locations in the mesh. This generalization of `PolyMesher` is discussed in Appendix A. For all the cases investigated, the underlying mesh has 10,000 n-gons.

Depending on the aspect ratio of Ω, determined by the angle θ of the centerline, the optimal shape consists of two separate pipes or one connected pipe transporting the fluid (cf. Fig. 8). We can see from Fig. 8b that for larger angles, a single but wider pipe is more advantageous even though it requires the fluid to travel a longer path. By contrast, in the smaller angles and shorter domains, the optimal design consists of two straight pipes.

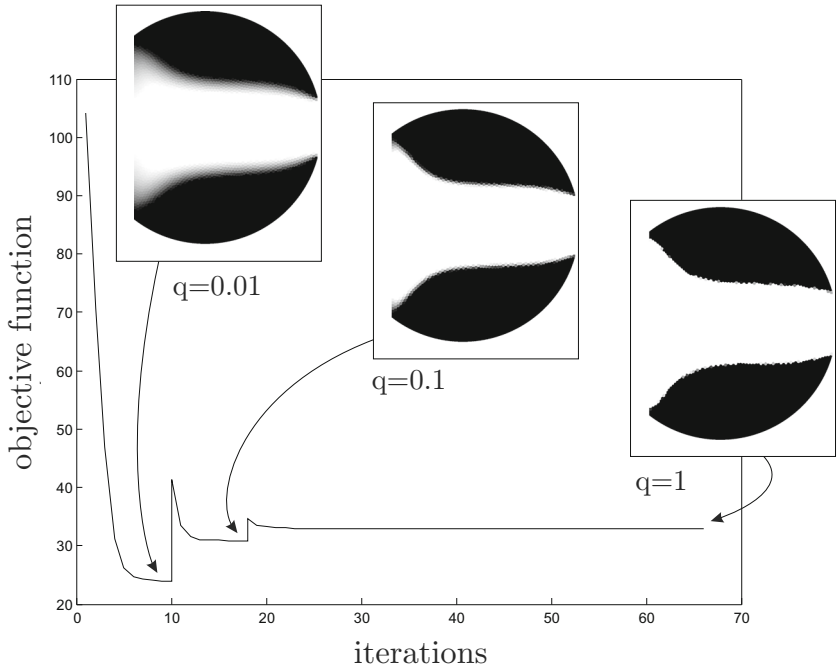


Fig. 6 Diffuser problem convergence

The results shown in Fig. 8a and b were obtained using the conservative continuation procedure. We next use the double pipe problem to explore the sensitivity of the optimal solution to the choice of parameters in the formulation. For the domain defined by $\theta = 23°$, both continuation schemes produce solutions consisting of two separate pipes, as seen from Fig. 8a and c. Increasing the aspect ratio of the domain using $\theta = 24°$, we obtain two different solutions on the same mesh. The conservative continuation scheme leads to the single pipe solution, which, upon inspection of the value of the cost function, happens to be the better solution for this domain (compare Fig. 8b and d).

In the case of the longer domain (Fig. 9) , with $\theta = 45°$, the default continuation scheme leads to the suboptimal curved solution shown in Fig. 9a. The value of the cost function for this solution is 44.440, compared to 32.663 associated with the solution in Fig. 9c on the same mesh. When we take the extended domain Ω to be the bounding box containing the inlets and the outlet, the default continuation procedure again yields the "correct" topology of a single straight pipe but shape is suboptimal (Fig. 9b). The conservative scheme, on the other hand, produces nearly the same solution for both domains (see Fig. 9c and d). Table 3 summarizes the results of the double pipe problem.

6 Extensions

We conclude the paper with a brief discussion of possible extensions of the present implementation to accommodate alternative formulations and cost functions. First we will discuss the implementation of the formulation of the flow topology optimization wherein the viscosity is also a function of the design. Such a formulation has been previously used to solve the drag minimization problem in references Guest and Prévost (2006) and Wiker et al. (2007), though in different contexts and with different intended applications. Next we present an example of flow control where the velocity at a certain location in the domain is maximized in a prescribed direction. The term "fluid mechanism" is used in Gersborg-Hansen et al. (2005) to described this problem due to its similarities with the compliant mechanism design

Table 2 Diffuser problem considering continuation with $q = 0.01$ as the initial value, followed by an increase to $q = 0.1$ comparable to the scheme used in Borrvall and Petersson (2003)

Diffuser problem	2,500 elements		10,000 elements	
	# iterations	Objective	# iterations	Objective
Present work (curved domain)	18	31.31	19	30.64
Present work (square domain)	19	31.34	19	30.70
Borrvall and Petersson (2003)	29	30.59	33	30.46

Table 3 Double pipe problem data

Domain	θ	Continuation	# elements	# iterations	Objective
Serpentine	23°	Conservative	10,000	64	38.74
Serpentine	24°	Conservative	10,000	281	30.92
Serpentine	23°	Default	10,000	99	38.40
Serpentine	24°	Default	10,000	87	40.09
Serpentine	45°	Default	10,000	331	44.44
Box	45°	Default	16,910	254	50.09
Serpentine	45°	Conservative	10,000	181	32.66
Box	45°	Conservative	16,910	188	32.63

in solids. While we will not provide the Matlab code, we discuss the major changes needed in each case. For example, both problems involve objective functions with sensitivities that assume both positive and negative values and as a results the optimality criteria method cannot be used for the optimization. The `UpdateScheme` can be replaced by calls to Method of Moving Asymptotes (Svanberg 1987) or by a heuristic modification of the OC method (cf. section 2.6 of Bendsøe and Sigmund 2003). The latter was used for the numerical results reported here.

6.1 Design-dependent viscosity

In Talischi et al. (2012b), we discussed how different formulations for the compliance minimization can be incorporated into `PolyTop` by changing the material interpolation functions and the regularization maps. The same can be done in the present context by replacing the functions $m_\alpha(\rho)$ and $m_V(\rho)$ with suitable alternatives. To accommodate the formulation in Guest and Prévost (2006) and Wiker et al. (2007), however, an additional interpolation function needs to be introduced. Returning to the governing equations (13), we consider the formulation where the viscosity is a function of the design field ρ as follows:

$$-2\,\mathrm{div}\left[m_\mu(\rho)\boldsymbol{\epsilon}(\mathbf{u})\right] + m_\alpha(\rho)\mathbf{u} + \nabla p = \mathbf{0} \tag{38}$$

If $m_\mu(\rho)$ is defined in such a way that $m_\mu(1) \approx 0$, (38) reduces to Darcy law and thus the flow in the "solid" region can be viewed as flow in a porous medium with permeability proportional to $1/m_\alpha(\rho)$. If $m_\mu(1) = \mu$, we recover Stokes flow in the fluid domain since $m_\alpha(1) = 0$. While this difference in the physics of the flow motivates the study by Wiker et al. (2007), the goal in Guest and Prévost (2006) is to solve the same shape optimization problem as (6) but use the Darcy region to further penalize flow velocity in $\Omega\backslash\omega$. In both cases, the same objective function is considered[7]

$$\min_{\rho\in\mathcal{A}} \frac{1}{2}\int_\Omega \left[m_\mu(\rho)\boldsymbol{\epsilon}(\mathbf{u}_\rho) : \boldsymbol{\epsilon}(\mathbf{u}_\rho) + m_\alpha(\rho)\mathbf{u}_\rho\cdot\mathbf{u}_\rho\right]\mathrm{d}\mathbf{x}$$
$$\text{subject to} \quad \frac{1}{|\Omega|}\int_\Omega m_V(\rho)\mathrm{d}\mathbf{x} \le \overline{v} \tag{39}$$

where $\mathbf{u}_\rho$ solves (38) supplemented with incompressibility constraint and appropriate boundary conditions. Here we consider the material interpolation function for viscosity defined by

$$m_\mu(\rho) = \mu\left[\tilde{\varepsilon}\,(1-\rho) + \rho\right] \tag{40}$$

where $\tilde{\varepsilon}$ is a small positive lower bound introduced to avoid zero viscosity and elimination of the higher order term in (38). Here we have used $\tilde{\varepsilon} = \varepsilon = 4\times 10^{-5}$.

Aside from the obvious change to `MatIntFnc`, the `FEAnalysis` must be supplied with an array of elemental viscosities $\tilde{\mathbf{E}} = m_\mu(\mathbf{z})$. The matrix $\mathbf{A}^\mu$ is accordingly modified as:

$$\mathbf{A}^\mu(\tilde{\mathbf{E}}) = \sum_{\ell=1}^{N} \tilde{E}_\ell \mathbf{A}^\mu_\ell, \qquad \tilde{E}_\ell = m_\mu(z_\ell) \tag{41}$$

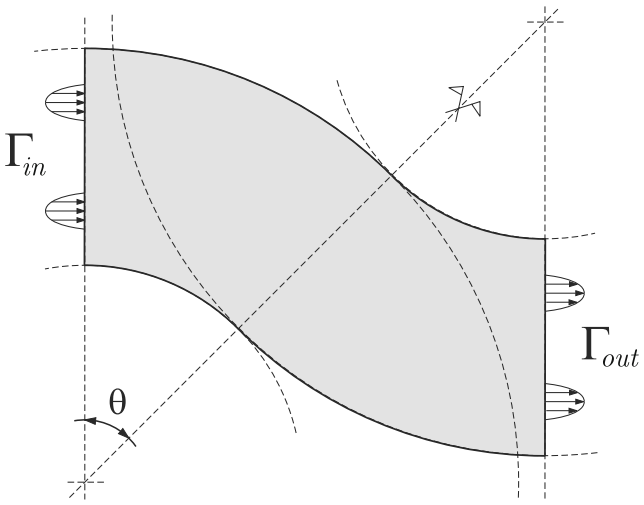
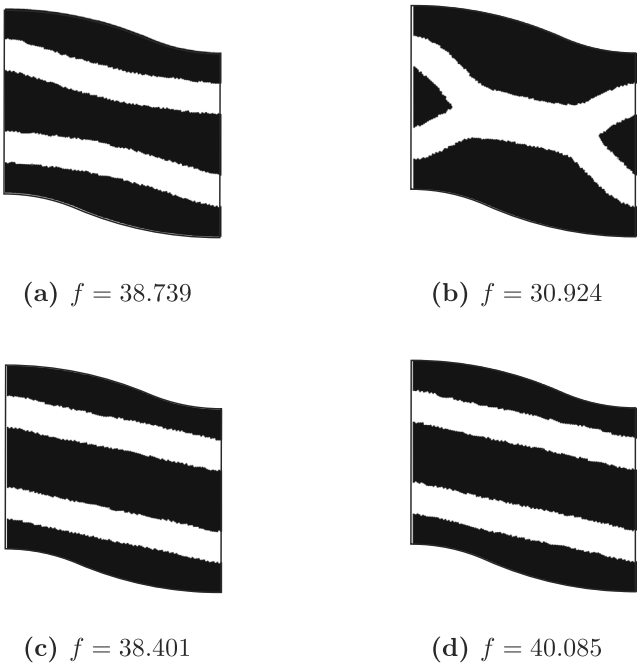


Fig. 7 Double pipe problem extended domain and boundary conditions

[7] We must note that, unlike optimization problem (12), this problem is ill-posed unless additional regularity is imposed on ρ, for instance, via filtering (Wiker et al. 2007). In the numerical result presented here, we do not consider filtering as we are mainly interested in comparing the results to the solutions of the optimization problem (12). However, we note that filtering can be easily enabled in `PolyTop` through the filtering matrix `opt.P`.



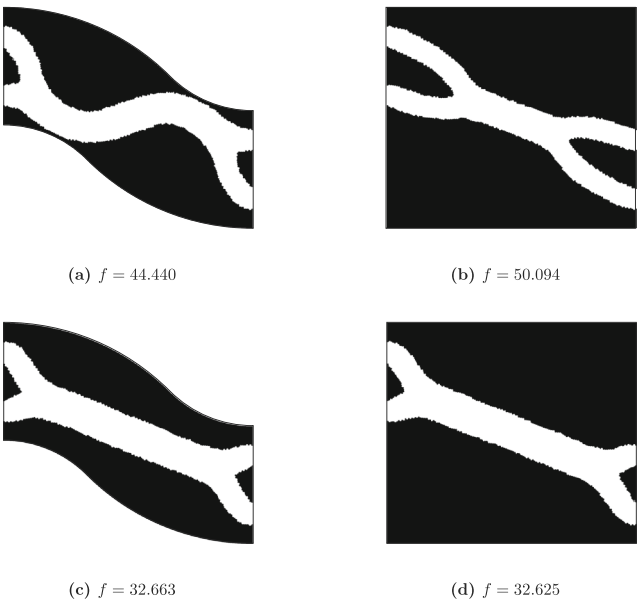
(a) $f = 38.739$ (b) $f = 30.924$

(c) $f = 38.401$ (d) $f = 40.085$

Fig. 8 Double pipe problem with the conservative scheme **a** $\theta = 23°$; **b** $\theta = 24°$ and default continuation scheme **c** $\theta = 23°$; **d** $\theta = 24°$

The discrete version of the objective function is given by $f = \frac{1}{2}\mathbf{U}^T[\mathbf{A}^\mu(\tilde{\mathbf{E}}) + \mathbf{A}^\alpha(\mathbf{E})]\mathbf{U}$ and its sensitivities with respect to elemental viscosity, permeability and volume are as follows:

$$\frac{\partial f}{\partial \tilde{E}_\ell} = \frac{1}{2}\mathbf{U}^T\mathbf{A}_\ell^\mu\mathbf{U}, \qquad \frac{\partial f}{\partial E_\ell} = \frac{1}{2}\mathbf{U}^T\mathbf{A}_\ell^\alpha\mathbf{U}, \qquad \frac{\partial f}{\partial V_\ell} = 0 \tag{42}$$



(a) $f = 44.440$ (b) $f = 50.094$

(c) $f = 32.663$ (d) $f = 32.625$

Fig. 9 Solutions to the double pipe problem **a** $\theta = 45°$, default continuation; **b** box domain, default continuation; **c** $\theta = 45°$, conservative continuation; **d** box domain, conservative continuation

Note that the sensitivity with respect to the design variable now requires an additional term in the chain rule (cf. line 17 of `PolyTop`):

$$\frac{\partial f}{\partial z_k} = \sum_{\ell=1}^{N}\left(\frac{\partial \tilde{E}_\ell}{\partial z_k}\frac{\partial f}{\partial \tilde{E}_\ell} + \frac{\partial E_\ell}{\partial z_k}\frac{\partial f}{\partial E_\ell} + \frac{\partial V_\ell}{\partial z_k}\frac{\partial f}{\partial V_\ell}\right) \tag{43}$$

We solve the double pipe problem on the serpentine domain using this formulation and the results are shown in Fig. 10. We can see that the default continuation scheme leads to a suboptimal topology of two pipes (Fig. 10 b). Interestingly, small initial values of q (less than 10^{-4}) also makes the algorithm very sensitive and in some case divergent. This is perhaps due to competing effects of changes in viscosity and inverse permeability. As seen from Fig. 10a, a continuation scheme with increment $q = 10^k$, $k = -4, \ldots, 0$, does yield the optimal single pipe topology.

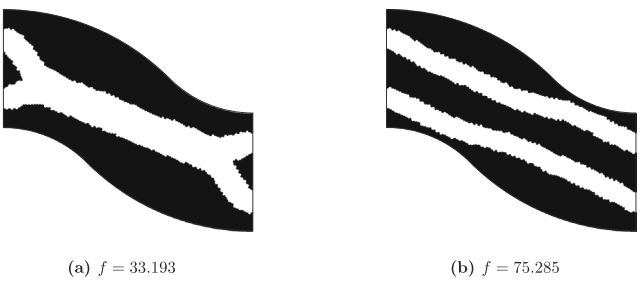
6.2 Fluid mechanism

The objective of the mechanism problem is maximize velocity of the fluid in a region $D \subseteq \Omega$ and in a prescribed direction $\mathbf{d}$. We consider an optimization problem that consists of minimizing the following cost functional

$$-\int_D \mathbf{d}\cdot\mathbf{u}_\rho \mathrm{d}\mathbf{x} + \frac{\gamma}{2}\int_\Omega \left[\mu\boldsymbol{\epsilon}(\mathbf{u}_\rho) : \boldsymbol{\epsilon}(\mathbf{u}_\rho) + m_\alpha(\rho)\mathbf{u}_\rho\cdot\mathbf{u}_\rho\right]\mathrm{d}\mathbf{x} \tag{44}$$

subject to the usual volume constraint. The dissipation term is included here only to prevent the optimization algorithm from getting trapped at a local optimum (the scaling coefficient γ is set to 10^{-4}). Without this term, which is essentially a penalty term representing a constraint on the total pressure drop, the solid elements tend to appear in the inlet and outlet regions during the early stages of the optimization, effectively reducing the velocity at D to zero.

Since this objective is not self-adjoint, the sensitivity analysis requires the solution to an adjoint boundary value problem. This adjoint problem must be introduced in the `FEAnalysis` function and the function evaluation and



(a) $f = 33.193$ (b) $f = 75.285$

Fig. 10 Solutions to the double pipe problem considering design-dependent viscosity and $\theta = 45°$ **a** continuation with increments $q = 10^k$ for $k = -4, -3, \ldots, 0$ **b** default continuation on q

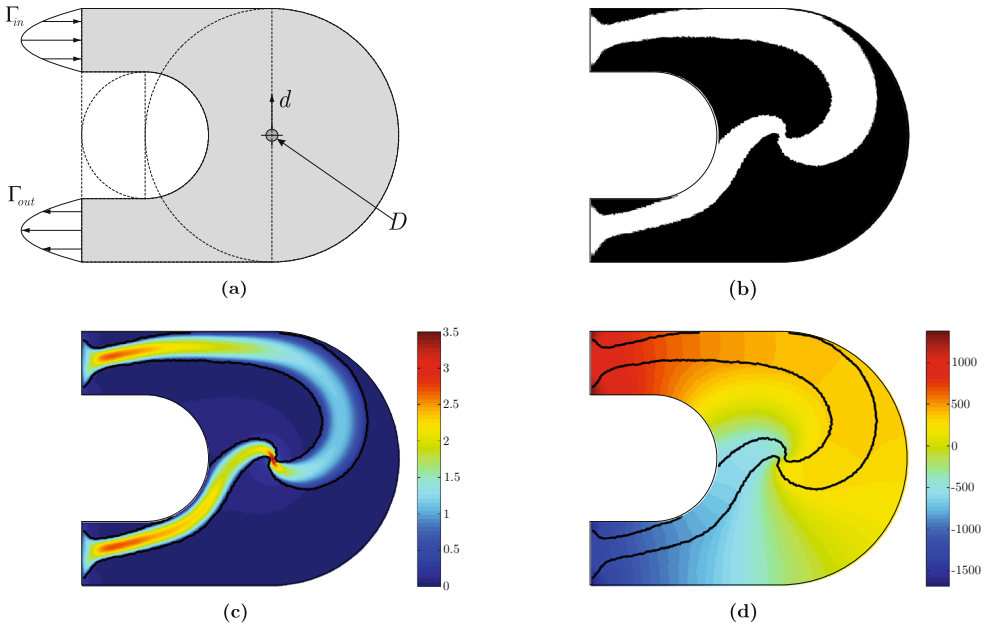


Fig. 11 Fluid mechanism problem **a** domain geometry and boundary conditions; The mesh is composed of 20,000 elements (18 4-gons, 2,207 5-gons, 16,309 6-gons and 1,466 7-gons) and 39,959 nodes; **b** Final solution; **c** Velocity field and **d** Pressure field

sensitivity calculations in `ObjectiveFnc` should be modified accordingly. We refer the reader to Pereira et al. (2011) which contains a discussion of modification of `PolyTop` for elasticity to the compliant mechanism problem. The results for a representative mechanism problem are shown in Fig. 11. Here the region D is considered to collapse to a single point.

7 Concluding remarks

Following the `PolyTop` philosophy (Talischi et al. 2012b), we have presented a Matlab implementation of topology optimization for fluid flow problems using unstructured polygonal FE meshes in arbitrary domains. Due to the modular structure of the Matlab code, the analysis routine and optimization algorithm are separated from the specific choice of topology optimization formulation. For the model problem of minimizing dissipated power in Stokes flow, the changes to the `PolyTop` function (originally written for compliance minimization in elasticity) involve few lines of code. As polygonal finite elements are employed for the analysis, the basis function construction and element integration routines remain intact. Moreover, the FE and sensitivity analysis routines contain no information related to the formulation and thus can be extended, maintained, developed, and/or modified independently. In a companion paper, we have provided a general purpose mesh

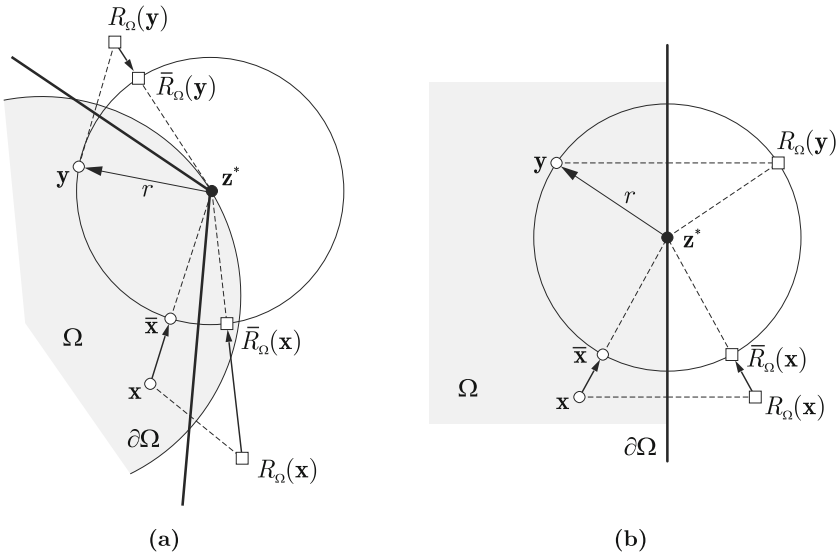


Fig. 12 Fixed vertices on the boundary are specified by adjusting the location of the nearby seeds and their reflections. Two scenarios are shown in the figure

generator for polygonal elements written in Matlab, called PolyMesher (Talischi et al. 2012a), which was used to generate all the meshes used in the present study. In this paper, the discretization of the governing incompressible Stokes equations uses an efficient low-order mixed FE formulation, featuring piecewise constant approximation of the pressure field and a linear velocity field on the boundary of the polygonal elements. The stability of this formulation is addressed in a companion paper (Talischi et al. 2014) in which the Babuska-Brezzi conditions are shown to hold for a large class of polygonal meshes including those obtained from Voronoi tessellations generated by PolyMesher. Here, the same mesh represents the velocity and pressure fields that characterize the response, as well as the design field.

To illustrate the versatility of the present approach, we have explored alternative formulations and cost functions. For instance, we have extended the implementation of the formulation for flow topology optimization where the viscosity is also a function of the design. We have also investigated flow control where the velocity at a certain location in the domain is maximized in a prescribed direction. Finally, we hope that the modularity and flexibility offered by PolyTop/PolyMesher will be a motivating factor for the community to explore such framework in other problems beyond those covered in our publications.

Acknowledgments Glaucio H. Paulino acknowledges support from the US National Science Foundation through grants #1559594 (formerly #1335160) and #1624232 (formerly #1437535). Ivan F. M. Menezes and Anderson Pereira acknowledge the financial support provided by Tecgraf/PUC-Rio (Group of Technology in Computer Graphics), Rio de Janeiro, Brazil. The information presented in this publication is the sole opinion of the authors and does not necessarily reflect the views of the sponsors or sponsoring agencies.

Appendix A: Additions to PolyMesher

The polygonal finite element meshes used in this work are generated by PolyMesher (Talischi et al. 2012a) but with one main modification that allows to specify the exact location of the inlets and outlets. To this effect, a new function is added to the kernel in order to generate meshes that included user-defined fixed points as vertices in the mesh. The function PolyMshr_FixedPoints uses the property of the Voronoi diagram that an empty circle through three or more points defines a vertex. As shown in Fig. 12, consider seeds $\mathbf{x}$ and $\mathbf{y}$ and their reflections $R_\Omega(\mathbf{x})$ and $R_\Omega(\mathbf{y})$ are the two closest points in the mesh to a given fixed point $\mathbf{z}^*$. If $\mathbf{y}$ has a smaller distance, then the function moves the seed $\mathbf{x}$, $R_\Omega(\mathbf{x})$ and $R_\Omega(\mathbf{y})$ on a circle of radius $r = |\mathbf{y} - \mathbf{z}^*|$ and center $\mathbf{z}^*$. The new points $\bar{\mathbf{x}}$, $\bar{R}_\Omega(\mathbf{x})$ and $\bar{R}_\Omega(\mathbf{y})$ are now positioned such that the desired point $\mathbf{z}^*$ is a vertex of the resulting Voronoi diagram.

The user must define a list of fixed points within the Domain function, more specifically, in a function FixedPoints, which returns a two-column array containing a list with the coordinates of the fixed points. The PolyMesher kernel function is also modified on line 10 as follows:

```
BdBox = Domain('BdBox');
  PFix = Domain('PFix');
```

During each iteration of the Lloyd's algorithm, a call is made to the function PolyMshr_FixedPoints. In particular, the line P=Pc (line 15) is replaced by the following line of code:

```
P = PolyMshr_FixedPoints(Pc,PFix);
```

The code PolyMshr_FixedPoints function is short and is provided below:

```
function [P,R_P] = PolyMshr_FixedPoints(P,R_P,PFix)
PP = [P;R_P];
for i = 1:size(PFix,1)
  [B,I] = sort(sqrt((PP(:,1)-PFix(i,1)).^2+(PP(:,2)-PFix(i,2)).^2));
  for j = 2:4
    n = PP(I(j),:) - PFix(i,:); n = n/norm(n);
    PP(I(j),:) = PP(I(j),:)-n*(B(j)-B(1));
  end
end
P = PP(1:size(P,1),:); R_P = PP(1+size(P,1):end,:);
```

In order to enforce velocity boundary conditions, the domain function must identify the nodes that lie on the boundary $\partial\Omega$ and assign prescribed velocity values to the associated degrees of freedom. This information is provided by a call to the auxiliary function PolyBoundary. This function uses the Matlab built-in function freeBoundary that queries the edges of a triangulation and outputs those incident on exactly one triangle. In PolyBoundary, each polygonal element of the mesh is first triangulated with respect to its centroid. The resulting triangulation is used as the input to freeBoundary. The list of boundary nodes is the list of nodes corresponding to the boundary edges output by freeBoundary.

Appendix B: PolyScript

```
%--------------------------- PolyScript ---------------------------------%
% Ref: A Pereira, C Talischi, GH Paulino, IFM Menezes, MS Carvalho       %
%      "Fluid Flow Topology Optimization in PolyTop: Stability and       %
%      Computational Implementation", Struct Multidisc Optim,            %
%      DOI 10.1007/s00158-014-1182-z                                     %
%-----------------------------------------------------------------------%

%% ------------------------------------------------------ CREATE 'fem' STRUCT
[Node,Element,NodeBC] = PolyMesher(@DiffuserDomain,5000,30);
fem = struct(...
  'NNode',size(Node,1),...      % Number of nodes
  'NElem',size(Element,1),...   % Number of elements
  'Node',Node,...               % [NNode x 2] array of nodes
  'Element',{Element},...       % [NElement x Var] cell array of elements
  'NodeBC',NodeBC,...           % Array of velocity boundary conditions
  'mu0',1,...                   % Dynamic viscosity
  'Reg',0 ...                   % Tag for regular meshes
   );
%% ------------------------------------------------------ CREATE 'opt' STRUCT
R = -1; % P is set to identity when R<0
VolFrac = 0.460847363744793;
m = @(y)MatIntFnc(y,[fem.mu0 0.01]);
P = PolyFilter(fem,R);
zIni = VolFrac*ones(size(P,2),1);
opt = struct(...
  'zMin',0.0,...                % Lower bound for design variables
  'zMax',1.0,...                % Upper bound for design variables
  'zIni',zIni,...               % Initial design variables
  'MatIntFnc',m,...             % Handle to material interpolation fnc.
  'P',P,...                     % Matrix that maps design to element vars.
  'VolFrac',VolFrac,...         % Specified volume fraction cosntraint
  'Tol',0.01,...                % Convergence tolerance on design vars.
  'MaxIter',150,...             % Max. number of optimization iterations
  'OCMove',0.2,...              % Allowable move step in OC update scheme
  'OCEta',0.5 ...               % Exponent used in OC update scheme
   );
%% ------------------------------------------------------------ RUN 'PolyTop'
figure;
for q = [0.01,0.1,1]  % Continuation on the penalty parameter
   disp(['current q: ', num2str(q)]);
   opt.MatIntFnc = @(y)MatIntFnc(y,[fem.mu0 q]);
   [opt.zIni,V,fem] = PolyTop(fem,opt);
end
%% -----------------------------------------------------------------------
```

Appendix C: PolyTop

```
%------------------------------ PolyTop ----------------------------------%
% Ref: A Pereira, C Talischi, GH Paulino, IFM Menezes, MS Carvalho        %
%      "Fluid Flow Topology Optimization in PolyTop: Stability and        %
%      Computational Implementation", Struct Multidisc Optim,             %
%      DOI 10.1007/s00158-014-1182-z                                      %
%-------------------------------------------------------------------------%
function [z,V,fem] = PolyTop(fem,opt)
Iter=0; Tol=opt.Tol*(opt.zMax-opt.zMin); Change=2*Tol; z=opt.zIni; P=opt.P;
[E,dEdy,V,dVdy] = opt.MatIntFnc(P*z);
[FigHandle,FigData] = InitialPlot(fem,V);
while (Iter<opt.MaxIter) && (Change>Tol)
  Iter = Iter + 1;
  %Compute cost functionals and analysis sensitivities
  [g,dgdE,dgdV,fem] = ConstraintFnc(fem,E,V,opt.VolFrac);
  [f,dfdE,dfdV,fem] = ObjectiveFnc(fem,E,V);
  %Compute design sensitivities
  dfdz = P'*(dEdy.*dfdE + dVdy.*dfdV);
  dgdz = P'*(dEdy.*dgdE + dVdy.*dgdV);
  %Update design variable and analysis parameters
  [z,Change] = UpdateScheme(dfdz,g,dgdz,z,opt);
  [E,dEdy,V,dVdy] = opt.MatIntFnc(P*z);
  %Output results
  fprintf('It: %i \t Objective: %1.3f\tChange: %1.3f\n',Iter,f,Change);
  set(FigHandle,'FaceColor','flat','CData',V(FigData)); drawnow
end
%---------------------------------------------------- OBJECTIVE FUNCTION
function [f,dfdE,dfdV,fem] = ObjectiveFnc(fem,E,V)
[U,F,fem] = FEAnalysis(fem,E);
f = 1/2*dot(F,U);
temp = cumsum(U(fem.iA).*fem.kAalpha.*U(fem.jA));
temp = temp(cumsum(fem.ElemNDofA.^2));
dfdE = 1/2*[temp(1);temp(2:end)-temp(1:end-1)];
dfdV = zeros(size(V));
%---------------------------------------------------- CONSTRAINT FUNCTION
function [g,dgdE,dgdV,fem] = ConstraintFnc(fem,E,V,VolFrac)
if ~isfield(fem,'ElemArea')
  fem.ElemArea = zeros(fem.NElem,1);
  for el=1:fem.NElem
    vx = fem.Node(fem.Element{el},1); vy = fem.Node(fem.Element{el},2);
    fem.ElemArea(el) = 0.5*sum(vx.*vy([2:end 1])-vy.*vx([2:end 1]));
  end
end
g = sum(fem.ElemArea.*V)/sum(fem.ElemArea)-VolFrac;
dgdE = zeros(size(E));
dgdV = fem.ElemArea/sum(fem.ElemArea);
%------------------------------------------- OPTIMALITY CRITERIA UPDATE
function [zNew,Change] = UpdateScheme(dfdz,g,dgdz,z0,opt)
zMin=opt.zMin; zMax=opt.zMax;
move=opt.OCMove*(zMax-zMin); eta=opt.OCEta;
l1=0; l2=1e6;
while l2-l1 > 1e-4
  lmid = 0.5*(l1+l2);
  B = -(dfdz./dgdz)/lmid;
  zCnd = zMin+(z0-zMin).*B.^eta;
  zNew = max(max(min(min(zCnd,z0+move),zMax),z0-move),zMin);
  if (g+dgdz'*(zNew-z0)>0),  l1 = lmid;
  else                       l2 = lmid;  end
```

```
end
Change = max(abs(zNew-z0))/(zMax-zMin);
%---------------------------------------------------------- FE-ANALYSIS
function [U,F,fem] = FEAnalysis(fem,E)
if ~isfield(fem,'kAmu')
  fem.ElemNDofA = 2*cellfun(@length,fem.Element);
  fem.iA = zeros(sum(fem.ElemNDofA.^2),1);
  fem.jA=fem.iA; fem.kAmu=fem.iA; fem.kAalpha=fem.iA; fem.e=fem.iA;
  fem.iB = zeros(sum(fem.ElemNDofA),1); fem.jB=fem.iB; fem.kB=fem.iB;
  fem.NDof = 2*fem.NNode+fem.NElem+1;
  indexA = 0; indexB = 0;
  if ~isfield(fem,'ShapeFnc'), fem=TabShapeFnc(fem); end
  for el = 1:fem.NElem
    eNode = fem.Element{el};
    eNDof = fem.ElemNDofA(el);
    if (el==1 || ~fem.Reg),  [Amu_e,Aalpha_e,Be]=LocalK(fem,eNode); end
    eDofA = reshape([2*eNode-1;2*eNode],eNDof,1);
    I=repmat(eDofA ,1,eNDof); J=I';
    fem.iA(indexA+1:indexA+eNDof^2) = I(:);
    fem.jA(indexA+1:indexA+eNDof^2) = J(:);
    fem.kAmu(indexA+1:indexA+eNDof^2) = Amu_e(:);
    fem.kAalpha(indexA+1:indexA+eNDof^2) = Aalpha_e(:);
    fem.e(indexA+1:indexA+eNDof^2) = el;
    fem.iB(indexB+1:indexB+eNDof) = eDofA(:);
    fem.jB(indexB+1:indexB+eNDof) = el;
    fem.kB(indexB+1:indexB+eNDof) = Be(:);
    indexA = indexA + eNDof^2;
    indexB = indexB + eNDof;
  end
  fem.FixedDofs = zeros(1,size(fem.NodeBC,1));
  fem.G = zeros(size(fem.NodeBC,1),1);
  for i = 1:size(fem.NodeBC,1)
    fem.FixedDofs(i) = 2*(fem.NodeBC(i,1)-1) + fem.NodeBC(i,2);
    fem.G(i) = fem.NodeBC(i,3);
  end
  fem.FreeDofs = setdiff(1:fem.NDof,fem.FixedDofs);
end
A = sparse(fem.iA,fem.jA,fem.kAmu+E(fem.e).*fem.kAalpha);
B = sparse(fem.iB,fem.jB,fem.kB);
Z=zeros(2*fem.NNode,1); O=sparse(fem.NElem,fem.NElem);
K = [A B Z; B' O fem.ElemArea; Z' fem.ElemArea' 0];
K = (K+K')/2;
S = zeros(fem.NDof,1); S(fem.FixedDofs,:)=fem.G;
S(fem.FreeDofs,:) = K(fem.FreeDofs,fem.FreeDofs)\...
                      (-K(fem.FreeDofs,fem.FixedDofs)*S(fem.FixedDofs,:));
U = S(1:2*fem.NNode);
F = A*U;
%------------------------------------------------ ELEMENT STIFFNESS MATRICES
function [Amu_e,Aalpha_e,Be] = LocalK(fem,eNode)
Cmu = fem.mu0*[2 0 0; 0 2 0; 0 0 1];
nn=length(eNode); Amu_e=zeros(2*nn,2*nn); Aalpha_e=Amu_e; Be=zeros(2*nn,1);
W = fem.ShapeFnc{nn}.W;
for q = 1:length(W)  %quadrature loop
  dNdxi = fem.ShapeFnc{nn}.dNdxi(:,:,q);
  J0 = fem.Node(eNode,:)'*dNdxi;
  dNdx = dNdxi/J0;
  B = zeros(3,2*nn);
  B(1,1:2:2*nn) = dNdx(:,1)';
```

```
  B(2,2:2:2*nn) = dNdx(:,2)';
  B(3,1:2:2*nn) = dNdx(:,2)';
  B(3,2:2:2*nn) = dNdx(:,1)';
  Amu_e = Amu_e + B'*Cmu*B*W(q)*det(J0);
  N = fem.ShapeFnc{nn}.N(:,:,q);
  Nu = zeros(2,2*nn);
  Nu(1,1:2:2*nn) = N(:);
  Nu(2,2:2:2*nn) = N(:);
  Aalpha_e = Aalpha_e + Nu'*Nu*W(q)*det(J0);
  dNudx = reshape(dNdx',1,2*nn);
  Be = Be - dNudx'*W(q)*det(J0);
end
%------------------------------------------------- TABULATE SHAPE FUNCTIONS
function fem = TabShapeFnc(fem)
ElemNNode = cellfun(@length,fem.Element); % number of nodes per element
fem.ShapeFnc = cell(max(ElemNNode),1);
for nn = min(ElemNNode):max(ElemNNode)
  [W,Q] = PolyQuad(nn);
  fem.ShapeFnc{nn}.W = W;
  fem.ShapeFnc{nn}.N = zeros(nn,1,size(W,1));
  fem.ShapeFnc{nn}.dNdxi = zeros(nn,2,size(W,1));
  for q = 1:size(W,1)
    [N,dNdxi] = PolyShapeFnc(nn,Q(q,:));
    fem.ShapeFnc{nn}.N(:,:,q) = N;
    fem.ShapeFnc{nn}.dNdxi(:,:,q) = dNdxi;
  end
end
%------------------------------------------------- POLYGONAL SHAPE FUNCTIONS
function [N,dNdxi] = PolyShapeFnc(nn,xi)
N=zeros(nn,1); alpha=zeros(nn,1); dNdxi=zeros(nn,2); dalpha=zeros(nn,2);
sum_alpha=0.0; sum_dalpha=zeros(1,2); A=zeros(nn,1); dA=zeros(nn,2);
[p,Tri] = PolyTrnglt(nn,xi);
for i=1:nn
  sctr = Tri(i,:); pT = p(sctr,:);
  A(i) = 1/2*det([pT,ones(3,1)]);
  dA(i,1) = 1/2*(pT(3,2)-pT(2,2));
  dA(i,2) = 1/2*(pT(2,1)-pT(3,1));
end
A=[A(nn,:);A]; dA=[dA(nn,:);dA];
for i=1:nn
  alpha(i) = 1/(A(i)*A(i+1));
  dalpha(i,1) = -alpha(i)*(dA(i,1)/A(i)+dA(i+1,1)/A(i+1));
  dalpha(i,2) = -alpha(i)*(dA(i,2)/A(i)+dA(i+1,2)/A(i+1));
  sum_alpha = sum_alpha + alpha(i);
  sum_dalpha(1:2) = sum_dalpha(1:2)+dalpha(i,1:2);
end
for i=1:nn
  N(i) = alpha(i)/sum_alpha;
  dNdxi(i,1:2) = (dalpha(i,1:2)-N(i)*sum_dalpha(1:2))/sum_alpha;
end
%------------------------------------------------- POLYGON TRIANGULATION
function [p,Tri] = PolyTrnglt(nn,xi)
p = [cos(2*pi*((1:nn))/nn); sin(2*pi*((1:nn))/nn)]';
p = [p; xi];
Tri = zeros(nn,3); Tri(1:nn,1)=nn+1;
Tri(1:nn,2)=1:nn; Tri(1:nn,3)=2:nn+1; Tri(nn,3)=1;
%------------------------------------------------- POLYGONAL QUADRATURE
function [weight,point] = PolyQuad(nn)
```

```
[W,Q]= TriQuad;                    %integration pnts & wgts for ref. triangle
[p,Tri] = PolyTrnglt(nn,[0 0]);    %triangulate from origin
point=zeros(nn*length(W),2); weight=zeros(nn*length(W),1);
for k=1:nn
  sctr = Tri(k,:);
  for q=1:length(W)
    [N,dNds] = TriShapeFnc(Q(q,:));  %compute shape functions
    J0 = p(sctr,:)'*dNds;
    l = (k-1)*length(W) + q;
    point(l,:) = N'*p(sctr,:);
    weight(l) = det(J0)*W(q);
  end
end
%------------------------------------------------------ TRIANGULAR QUADRATURE
function [weight,point] = TriQuad
point=[1/6,1/6;2/3,1/6;1/6,2/3]; weight=[1/6,1/6,1/6];
%------------------------------------------------------ TRIANGULAR SHAPE FUNCTIONS
function [N,dNds] = TriShapeFnc(s)
N=[1-s(1)-s(2);s(1);s(2)]; dNds=[-1,-1;1,0;0,1];
%------------------------------------------------------------------ INITIAL PLOT
function [handle,map] = InitialPlot(fem,z0)
Tri = zeros(length([fem.Element{:}])-2*fem.NElem,3);
map = zeros(size(Tri,1),1); index=0;
for el = 1:fem.NElem
  for enode = 1:length(fem.Element{el})-2
    map(index+1) = el;
    Tri(index+1,:) = fem.Element{el}([1,enode+1,enode+2]);
    index = index + 1;
  end
end
handle = patch('Faces',Tri,'Vertices',fem.Node,'FaceVertexCData',...
                z0(map),'FaceColor','flat','EdgeColor','none');
axis equal; axis off; axis tight; colormap(gray); caxis([0 1]);
%-------------------------------------------------------------------------%
```

References

Allaire G (2001) Shape optimization by the homogenization method. Springer, New York

Beirão Da Veiga L, Lipnikov K (2010) A mimetic discretization of the Stokes problem with selected edge bubbles. SIAM J Sci Comput 32(2):875–893. doi:10.1137/090767029

Bendsøe MP, Sigmund O (2003) Topology optimization: theory, methods and applications. Springer-Verlag, Berlin

Bochev P, Lehoucq RB (2001) On finite element solution of the pure Neumann problem. SIAM Rev 47:50–66. doi:10.1137/S0036144503426074

Borrvall T, Petersson J (2003) Topology optimization of fluids in Stokes flow. Int J Numer Methods Fluids 41(1):77–107. doi:10.1002/fld.426

Challis VJ, Guest JK (2009) Level set topology optimization of fluids in Stokes flow. Int J Numer Methods Eng 79(10):1284–1308. doi:10.1002/nme.2616

Deng Y, Liu Z, Zhang P, Liu Y, Wu Y (2011) Topology optimization of unsteady incompressible Navier-Stokes flows. J Comput Phys 230(17):6688–6708. doi:10.1016/j.jcp.2011.05.004

Donea J, Huerta A (2003) Finite element methods for flow problems. Wiley, West Sussex

Evgrafov A (2005) The limits of porous materials in the topology optimization of Stokes flows. Appl Math Optim 52:263–277. doi:10.1007/s00245-005-0828-z

Gersborg-Hansen A, Sigmund O, Haber R (2005) Topology optimization of channel flow problems. Struct Multidiscip Optim 30(3):181–192. doi:10.1007/s00158-004-0508-7

Guest JK, Prévost JH (2006) Topology optimization of creeping fluid flows using a Darcy-Stokes finite element. Int J Numer Methods Eng 66(3):461–484. doi:10.1002/nme.1560

Kreissl S, Maute K (2012) Levelset based fluid topology optimization using the extended finite element method. Struct Multidiscip Optim 46(3):311–326. doi:10.1007/s00158-012-0782-8

Kreissl S, Pingen G, Maute K (2011) An explicit level set approach for generalized shape optimization of fluids with the Lattice Boltzmann method. Int J Numer Methods Fluids 65(5):496–519. doi:10.1002/fld.2193

Kreissl S, Pingen G, Maute K (2011) Topology optimization for unsteady flow. Int J Numer Methods Eng 87(13):1229–1253. doi:10.1002/nme.3151

Pereira A, Menezes IFM, Talischi C, Paulino GH (2011) An efficient and compact Matlab implementation of topology optimization: application to compliant mechanisms. In: XXXII Iberian Latin-American Congress on Computational Methods in Engineering

Pingen G, Evgrafov A, Maute K (2007) Topology optimization of flow domains using the lattice Boltzmann method. Struct Multidisc Optim 34(6):507–524. doi:10.1007/s00158-007-0105-7

Svanberg K (1987) The method of moving asymptotes—a new method for structural optimization. Int J Numer Methods Eng 24(2):359–373. doi:10.1002/nme.1620240207

Talischi C, Paulino GH, Pereira A, Menezes IFM (2012) PolyMesher: a general-purpose mesh generator for polygonal elements written in Matlab. Struct Multidiscip Optim 45(3):309–328. doi:10.1007/s00158-011-0706-z

Talischi C, Paulino GH, Pereira A, Menezes IFM (2012) PolyTop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes. Struct Multidiscip Optim 45(3):329–357. doi:10.1007/s00158-011-0696-x

Talischi C, Pereira A, Paulino GH, de Menezes IFM, da Silveira Carvalho M (2014) Polygonal finite elements for incompressible fluid flow. Int J Numer Methods Fluids 74(2):134–151. doi:10.1002/fld.3843

Wiker N, Klarbring A, Borrvall T (2007) Topology optimization of regions of Darcy and Stokes flow. Int J Numer Methods Eng 69(7):1374–1404. doi:10.1002/nme.1811