Optimization Methods and Software

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/goms20

# On the implementation of a quasi-Newton interior-point method for PDE-constrained optimization using finite element discretizations

Cosmin G. Petra, Miguel Salazar De Troya, Noemi Petra, Youngsoo Choi, Geoffrey M. Oxberry & Daniel Tortorelli

**To cite this article:** Cosmin G. Petra, Miguel Salazar De Troya, Noemi Petra, Youngsoo Choi, Geoffrey M. Oxberry & Daniel Tortorelli (2023) On the implementation of a quasi-Newton interior-point method for PDE-constrained optimization using finite element discretizations, Optimization Methods and Software, 38:1, 59-90, DOI: 10.1080/10556788.2022.2117354

**To link to this article:** https://doi.org/10.1080/10556788.2022.2117354

Published online: 21 Nov 2022.

OPTIMIZATION METHODS & SOFTWARE
2023, VOL. 38, NO. 1, 59–90
https://doi.org/10.1080/10556788.2022.2117354

Taylor & Francis
Taylor & Francis Group



# On the implementation of a quasi-Newton interior-point method for PDE-constrained optimization using finite element discretizations

Cosmin G. Petra [a], Miguel Salazar De Troya [a], Noemi Petra [b], Youngsoo Choi [a], Geoffrey M. Oxberry [a] and Daniel Tortorelli [a]

[a]Lawrence Livermore National Laboratory (LLNL), Livermore, CA, USA; [b]Applied Mathematics Department, University of California, Merced, Merced, CA, USA

**ABSTRACT**

We present a quasi-Newton interior-point method appropriate for optimization problems with pointwise inequality constraints in Hilbert function spaces. Among others, our methodology applies to optimization problems constrained by partial differential equations (PDEs) that are posed in a reduced-space formulation and have bounds or inequality constraints on the optimized parameter function. We first introduce the formalization of an infinite-dimensional quasi-Newton interior-point algorithm using secant BFGS updates and then proceed to derive a discretized interior-point method capable of working with a wide range of finite element discretization schemes. We also discuss and address mathematical and software interface issues that are pervasive when existing off-the-shelf PDE solvers are to be used with off-the-shelf nonlinear programming solvers. Finally, we elaborate on the numerical and parallel computing strengths and limitations of the proposed methodology on several classes of PDE-constrained problems.

**ARTICLE HISTORY**
Received 28 November 2020
Accepted 19 August 2022

**KEYWORDS**
Quasi-Newton interior-point method; constrained optimization in function spaces; mesh independent optimization

**AMS SUBJECT CLASSIFICATIONS**
49M15; 49M37; 65K10; 90C26; 35Q93

## 1. Introduction

Many scientific and engineering applications result in optimization problems described by means of differential equations (DEs). Efficient numerical evaluation of the objective and constraints functions derivatives needed by the optimization method generally requires setting up and solving *adjoint* sensitivity DEs [2,11,23,36], which have a considerable human development cost, sometimes comparable to developing a solver for the underlying DEs. For this reason, many applications only provide first-order derivatives as the computation of second-order derivatives has considerable additional development cost and is not routinely provided.

In these circumstances, *quasi-Newton* algorithms [15] are a pragmatic and convenient choice since they internally build and maintain approximations of the second-order derivatives using only gradient evaluations. One popular class of such approximations consists of the so-called *secant* updates, which are also very appropriate for large-scale problems

**CONTACT** Cosmin G. Petra petra1@llnl.gov

if equipped with a limited-memory (or low-rank updates) mechanism [10]. In general, quasi-Newton algorithms do not have the convergence properties of Newton-like methods, but they can achieve Newton-like superlinear convergence for some classes of problems [14,35,41]; also, they have convergence properties and practical performance superior to gradient-based or derivative-free algorithms [35]. It is also possible that quasi-Newton methods can equal and even outperform Newton-like methods in practice, even in the cases where the second-order derivatives are available and cheap to evaluate. While the iteration count is generally lower for Newton than for quasi-Newton, Newton methods can have a considerably larger cost per iteration than quasi Newton methods, which is enough to offset the benefits of low iteration count. This situation is particularly likely when good preconditioners for the Newton linearization systems are not available.

In [39], the first author showed that a limited memory BFGS interior-point method has great parallelization potential since it scales to thousands of cores of a parallel computer; the present work can be viewed as a further step aimed at ensuring that the BFGS interior-point method performs efficiently and in a mesh independent manner for a few types of PDE-constrained optimization problems. More specifically, the present work addresses a couple of discretization peculiarities of quasi-Newton and interior-point methods in infinite dimensions that stem from the use of incorrect Riesz representers for dual variables and are pervasive when nonuniform discretization meshes are used with a finite element method.

This paper proposes a quasi-Newton optimization method using concepts from interior-point algorithms for optimization problems posed in Hilbert function spaces with complicating infinite-dimensional *inequality* constraints. Recently, interior-point algorithms in function spaces have been proposed and analysed theoretically both from the angle of well-posedness, iteration complexity, and local rate of convergence on one hand, for example in [5,27,48,53–55,62], and from a discretization consistency and order of convergence perspective on the other hand, for example, in [63]. We mention that our methodology only covers PDE-constrained optimization with *control* or *design* constraints; optimization problems with *state* constraints [52,62] have analytical properties (typically more restrictive than control-constrained problems, e.g. Lagrange multipliers are measures) that require different algorithmic approaches than the one we take in this work; we refer the reader to [25,27,48] for some recent algorithmic developments for PDE state-constrained optimization problems. Practical computational methods based on interior-point methods have been also in the attention of the community, for example, see [21,37,38]. The present work is in line with these latter works as it focuses on computational aspects; however, it differs in that it focuses on *quasi-Newton* optimization methods for reasons discussed in the previous paragraphs. Our specific goal in this work is to provide a unified quasi-Newton computational setup and a numerical solver capable of working with various finite element discretization schemes in a manner consistent with the underlying infinite-dimensional space. In our opinion this is a very important first step towards ensuring mesh independence and scalability to large problems of the solution methodology.

A comprehensive list of algorithms and software packages available for PDE-constrained problems is compiled by Funke and Farrell in [18]. This work lists two algorithmic approaches (i.e. [32,45]) as available for the general optimization in infinite dimensions with general inequalities that we consider in this paper, however, these algorithms are only

first-order, gradient-based. PETSc's TAO optimization package is also mentioned, however its module for PDE-constrained optimization currently does not support inequality constraints [13]. In addition, we mention hIPPYlib [56] that implements state-of-the-art scalable first- and second-order adjoint-based algorithms for PDE-based deterministic and Bayesian inverse problems. However this library does not treat general optimization constraints. Finally, we mention the Rapid Optimization Library (ROL) of Trilinos that provides an extensive collection of algorithms for PDE-constrained optimization. For infinite-dimensional inequality constraints, ROL uses a combination of matrix-free trust-region methods, projection methods, and primal-dual active set methods according to the user documentation, without a reference to quasi-Newton IPMs [31]. Therefore, we find that the present work adds to existing capabilities of the community of computational optimization.

A reader with background in nonlinear programming (NLP) in finite-dimensional Euclidean spaces may disagree with our choice of considering a complicating infinite-dimensional setup. Since the infinite-dimensional optimization problems we consider are invariably discretized, one may suggest instead to pose the discretizations as 'finite-dimensional' problems and to solve them using NLP solvers over the Euclidean space. However, using the Euclidean inner product instead of *the inner product of the underlying Hilbert* space causes incorrect representers to be used for certain derivative functionals and introduces discretization inconsistencies (e.g. mesh dependence) in the discretized optimality conditions. These discrepancies cause convergence behaviour dependent on the underlying discretization or meshing of the domain (see [49] and Section 3.6.1 for examples of simple problems and Section 6 for more complex problems where this issue is pervasive). Such mesh dependent behaviour is not necessarily specific to optimization and has been previously identified and addressed in the context of linear systems arising in PDE discretizations (for example, see [24,30]).

In addition, in many cases, NLP solvers over the Euclidean space provide solutions that are mesh dependent, as we show for topology optimization in Section 6.2, or are non-physical, as we illustrate in Figure 1. The lack of spatial symmetry of the right design of Figure 1 is likely caused by numerical round-off errors that are exacerbated by the nonsymmetric mesh and the use of the (unweighted) $l_2$ inner product by the Euclidean NLP solver[1] used to obtain this design. All the aforementioned deficiencies of the NLP solvers are pervasively exacerbated when the mesh is distorted, which is a common occurrence in mesh adaptation and refinement techniques used by state-of-the-art solvers for PDEs. In this regard, the present paper lays out all the necessary mathematical and computational components needed by a *mesh independent* numerical quasi-Newton IPM algorithm.

### 1.1. Notation

Since the paper goes back and forth between optimization, functional analysis, finite-element spaces, and computational engineering, it is difficult to use a notation that transcends all these communities. For this reason, the notation conventions below are a compromise and it is our hope that they are intelligible. Given two normed spaces $U$ and $V$, we denote the metric space of all linear operators from $U$ to $V$ by $\mathcal{L}(U, V)$. The symbol $U$ refers to an Hilbert space of scalar valued functions defined over a bounded, open, and

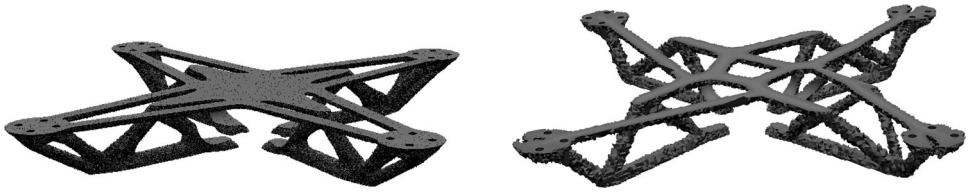

**Figure 1.** Shown are two designs for a commercial quad-motor drone body obtained by solving a structural topology optimization with a slightly distorted mesh. The valid left design was optimized with the proposed infinite-dimensional algorithm while the right design was obtained using an Euclidean NLP solver and is nonphysical since it lacks spatial symmetry.

connected set $\Omega$, of $\mathbb{R}^d$ ($d \in \{1, 2, 3\}$). The dual $\mathcal{L}(U, \mathbb{R})$ of the Hilbert space $U$ is denoted by $U^*$.

Normal math italic font is used for elements of $U$ and for functions, functionals, and operators that act on $U$. In general, we use lower cases for functions and functionals and upper cases for linear operators. For a linear functional $g \in U^*$ we use the shorthand $gu$ to refer to $g(u)$, i.e. the application of $g$ to an $u \in U$. When an operator $E : U \to V$ is linear, we also use $Ev$ to denote $E(v)$. For a linear operator $B : U \to U^*$, $Bv_1v_2$ denotes $[Bv_1](v_2)$. In some circumstances, the operator depends on some $u \in U$, in which case we express this dependence by $E(u)$, and $E(u)v$ denotes the application of the operator to $v \in U$. For $u, v \in U$, $uv$ denotes the pointwise multiplication of $u$ and $v$, that is $w = uv$ is defined by $w(x) = u(x)v(x)$. Similarly, we use $\frac{v}{u}$ for pointwise division and $\frac{\mu}{u}$ with $\mu \in \mathbb{R}$ for the function with values $\frac{\mu}{u(x)}$.

Roman fonts are used to denote discretized quantities. For vectors we use lower cases and for matrices and tensors we use upper cases. The compact, MATLAB-like notation $[v_1; v_2; \ldots; v_n]$ refers to $v \in \mathbb{R}^n$ with components $v_i$. The symbol e denotes the vector with all components equal to 1. The multiplication of vectors u and v from $\mathbb{R}^n$ is understood pointwise, i.e. uv $= [u_1v_1; \ldots; u_nv_n]$.

Our formal derivation below will explicitly use dual spaces of Hilbert spaces to 'type check' various quantities employed by the infinite-dimensional algorithm and to leave no room for guessing which inner products to use and the form of finite-dimensional representations when the discretization is performed. For a functional $f : U \to \mathbb{R}$, $Df(u) \in U^*$ denotes the Fréchet derivative at $u$, while $D^2f(u) \in \mathcal{L}(U, U^*)$ denotes the second derivative. The derivative of $a : U \to \mathbb{R}^m$ at $u \in U$ is denoted by $Da(u)$ and is in $\mathcal{L}(U, \mathbb{R}^m)$; its adjoint $Da(u)^*$ is in $\mathcal{L}(\mathbb{R}^{m*}, U^*)$. We will use linear functionals defined over Euclidean spaces, e.g. $\lambda \in \mathbb{R}^{m*} = \mathcal{L}(\mathbb{R}^m, \mathbb{R})$.

### 1.2. The optimization paradigm

We consider general problems of the form

$$\min_{u \in U} \quad f(u) \tag{1a}$$

$$\text{s.t.} \qquad a(u) = 0, \quad \underline{u} \leq u \leq \overline{u}. \tag{1b}$$

Here $U$ is a *real* Hilbert space of functions $u : \Omega \to \mathbb{R}$ at least square-integrable over a bounded domain $\Omega$ of $\mathbb{R}^d$ ($d = \{1, 2, 3\}$). Also, $f : U \to \mathbb{R}$ and $a : U \to \mathbb{R}^m$ are twice continuously differentiable functions. We assume throughout that the Fréchet derivative $Da$ is surjective and that $m$ is relatively small, i.e. $m = O(1)$. The bound constraints on the optimization variable $u$ are enforced pointwise almost everywhere and $\underline{u}, \overline{u} \in L^\infty(\Omega)$. We consider only bound and equality constraints for the sake of a compact presentation, but we remark that general inequality constraints can be accommodated by expressing them as a combination of equality constraints and slack variables with bound constraints. We do so in our numerical implementation.

The present work is motivated by PDE-constrained optimization problems with *bounds* and *equality* constraints on the controls or designs $u$ in the form of

$$\begin{aligned} \min_{u\in U, y\in Y} \quad & J(u, y) \\ \text{s.t.} \quad & c(y, u) = 0, \\ & a(u) = 0, \quad \underline{u} \le u \le \overline{u}, \end{aligned}$$

where $c : Y \times U \to W$ represents the PDE state equation parametrized by the optimization variable $u$ and $Y$ and $W$ are also real Hilbert spaces. Many such PDE-constrained problems allow solving for $y = y(u)$ for any given $u$, which allows formulating the above problem as a so-called reduced-space problem of the form (1a)–(1b) by taking $f(u) = J(u, y(u))$. We refer the reader to [26,28,52] for a thorough discussion of reduced-space PDE-constrained optimization. A frequent issue in this context is the efficient computation of the derivative $Df$, which almost invariably is done by using the so-called adjoint sensitivity approach. Namely, one can show under mild conditions (for example, see [28, Chapter 5]) that $Df(u) = c_u(y, u)^*\lambda + J_u(u, y)$, where $\lambda \in W^*$ is computed as the solution to the adjoint system $c_y(y, u)^*\lambda + J_y(y, u) = 0$ [2]. The adjoint sensitivity methods used for the inverse and structural topology optimization problems of Section 6 are covered, for example, in [44,51], respectively.

Characterizing solvability of (1a)–(1b) can be involving and needs to be done on a per-problem basis. A general condition of solvability of (1a)–(1b), namely, lower semicontinuity of $f$ with respect to sequential $L^\infty(\Omega)$ − weak$^*$ convergence, can be found in [54]. It is important to mention that the methodology of this paper focuses on pragmatic computational solutions and omits key analyses such as well posedness outside the assumptions of [54], convergence, and discretization error estimates.

### 1.3. Paper contributions

The paper proposes a comprehensive numerical quasi-Newton IPM algorithm for solving the optimization problem (1a)–(1b) in a Hilbert function space $U$ of at least square-integrable functions. First, we provide a mathematically sound formalization of an IPM in an infinite-dimensional setup with emphasis on its quasi-Newton specifics, such as quasi-Newton search direction linear systems, quasi-Newton secant update formulas, and dual variables updating. We also derive the optimization-specific convergence mechanisms, such as the line search and the stopping criteria required by our infinite-dimensional setup. Many constituents of our algorithm are inspired from previously published works: the IPM relies on the theoretical work of [54], the secant quasi-Newton formulas are well known

in the literature (e.g. see [3,15,35,41]), and the convergence mechanisms follow [59,60]). However, the assembly of our quasi-Newton IPM algorithm for the general optimization paradigm (1a)–(1b) posed over generic Hilbert function spaces is, to the best of our knowledge, novel. The closest quasi-Newton algorithm is that from [49, Chapter 1.5]. The limitation of this work is that it only considers a finite number of inequalities as opposed to the infinite dimensional constraints (1b). This is an important difference since the treatment of the (discretization of the) inequalities (1b) without consideration of the underlying Hilbert space and the corresponding finite element discretization space misses a so-called mass weighting matrix in the representation of the dual variables and in the log-barrier term that results in a *mesh dependent* algorithm. This is further illustrated in Section 3.6.1.

Another salient contribution consists of the algorithm's capability to work generically with various finite element discretization schemes for the underlying Hilbert space. We achieve this by judiciously identifying the optimization- and IPM-related discretization issues and by proposing computationally efficient, yet mathematically sound and general numerical resolutions to these issues. These developments are presented in detail in Section 3 for finite element discretization spaces.

We also discuss problem specification considerations and propose a *unified* solver interface that accommodates optimization over generic Hilbert function spaces with minimal user input and without requiring extensive knowledge of the underlying infinite-dimensional and discretization spaces. Central to our approach is the suitability to massively parallel computations. In regard to this aspect, the paper revisits the decomposition technique proposed by the first author in [39] and discusses its parallelization limitations under the infinite-dimensional setup of this paper.

## 2. Formal derivation of a primal-dual interior-point method in a function space

Specific to interior-point methods is the use of the log-barrier problem associated with the bound- and equality-constrained problem (1a)–(1b):

$$\begin{aligned} \min_{u\in U} \quad & \psi_\mu(u) := f(u) - \mu \int_\Omega \ln(u(x) - \underline{u}(x))\,\mathrm{d}x - \mu \int_\Omega \ln(\overline{u}(x) - u(x))\,\mathrm{d}x \\ \text{s.t.} \quad & a(u) = 0, \end{aligned} \tag{2}$$

where the barrier parameter $\mu$ is positive. The integral logarithm terms above correspond to inequalities (1b) and are such that the minimizer of (2) converges to the minimizer of (1a)–(1b) in the limit, as $\mu \to 0$. Under our assumptions for (1a)–(1b), there exist a local solution $u_\mu$ to the above log-barrier problem and a dual multiplier $\lambda_\mu \in \mathbb{R}^{m*}$ that satisfy

$$\begin{aligned} Df(u_\mu)v + [Da(u_\mu)^*\lambda_\mu]v - \mu \int_\Omega \frac{v(x)}{u_\mu(x) - \underline{u}(x)}\,\mathrm{d}x + \mu \int_\Omega \frac{v(x)}{\overline{u}(x) - u_\mu(x)}\,\mathrm{d}x = 0 \\ a(u_\mu) = 0 \end{aligned} \tag{3}$$

for all variations $v$ in a dense subset of $U$ [54, Lemma 1].

In order to derive a primal-dual interior-point method, we introduce the artificial variables $\underline{z}_\mu = \frac{\mu}{u_\mu - \underline{u}} \in U$ and $\overline{z}_\mu = \frac{\mu}{\overline{u}_\mu - u} \in U$. Furthermore, we define the linear operator

$E : U \to U^*$ such that for any given $u \in U$ we have $E(u)v = \int_\Omega u(x)v(x)\,\mathrm{d}x$ for all $v \in U$; we remark that the derivative $DE(u)$ of $E(u)$ satisfies $DE(u)v = E(v)$ for all $u, v \in U$. As a result, the optimality conditions (3) can be written compactly in the following 'primal-dual' form:

$$l(u_\mu, \lambda_\mu, \underline{z}_\mu, \overline{z}_\mu) := Df(u_\mu) + Da(u_\mu)^*\lambda_\mu - E(\underline{z}_\mu) + E(\overline{z}_\mu) = 0, \tag{4a}$$

$$a(u_\mu) = 0, \tag{4b}$$

$$(u_\mu - \underline{u})\underline{z}_\mu = \mu, \quad \text{and} \quad (\overline{u} - u_\mu)\overline{z}_\mu = \mu, \quad \text{a.e. in } \Omega. \tag{4c}$$

Equation (4a) can be safely posed in $U^*$ whenever the operators $E(\underline{z}_\mu)$ and $E(\overline{z}_\mu)$ are bounded (thus, in $U^*$). For $U = L^2(\Omega)$ and $U = H^1(\Omega)$, the boundedness follows from the Cauchy–Schwarz inequality

The solution to the system of Equations (4a)–(4c) is known as the central path and under appropriate assumptions is regular enough to allow the use of Newton's method [54]. The main idea of primal-dual interior-point methods is to approximately compute a solution $(u_\mu, \lambda_\mu, \underline{z}_\mu, \overline{z}_\mu)$ of (4a)–(4c) for a decreasing sequence of $\mu > 0$ that converges to 0. For a given value of the barrier parameter $\mu$, the numerical algorithm computes the Newton search direction $(\Delta u, \Delta\lambda, \Delta\underline{z}, \Delta\overline{z})$ by solving a linearization of (4a)–(4c) in the form of

$$H(u_\mu)\Delta u + Da(u_\mu)^*\Delta\lambda - DE(\underline{z}_\mu)\Delta\underline{z} + DE(\overline{z}_\mu)\Delta\overline{z} = -l(u_\mu, \lambda_\mu, \underline{z}_\mu, \overline{z}_\mu), \tag{5a}$$

$$Da(u_\mu)\,\Delta u = -a(u_\mu), \tag{5b}$$

$$\underline{z}_\mu\Delta u + (u_\mu - \underline{u})\Delta\underline{z} = \mu - (u_\mu - \underline{u})\underline{z}_\mu, \tag{5c}$$

$$-\overline{z}_\mu\Delta u + (\overline{u} - u_\mu)\Delta\overline{z} = \mu - (\overline{u} - u_\mu)\overline{z}_\mu. \tag{5d}$$

Above in (5a), $H(u) \in \mathcal{L}(U, U^*)$ is the partial derivative of $l(u, \lambda, \underline{z}, \overline{z})$ with respect to $u$, namely $H(u) = D^2 f(u) + D(Da(u)^*\lambda)$. We observe that (5a) can be simplified since $DE(\underline{z}_\mu)\Delta\underline{z} = E(\Delta\underline{z})$ and $DE(\overline{z}_\mu)\Delta\overline{z} = E(\Delta\overline{z})$. We also use the compact, operator-like notation $G(u)$ for pointwise product of functions: for a given $u \in U$, $G(u)v = uv$ for any $v \in U$. The linear system (5a)–(5d) becomes

$$H(u_\mu)\Delta u + Da(u_\mu)^*\Delta\lambda - E(\Delta\underline{z}) + E(\Delta\overline{z}) = -l(u_\mu, \lambda_\mu, \underline{z}_\mu, \overline{z}_\mu), \tag{6a}$$

$$Da(u_\mu)\,\Delta u = -a(u_\mu), \tag{6b}$$

$$G(\underline{z}_\mu)\Delta u + G(u_\mu - \underline{u})\Delta\underline{z} = \mu - (u_\mu - \underline{u})\underline{z}_\mu, \tag{6c}$$

$$-G(\overline{z}_\mu)\Delta u + G(\overline{u} - u_\mu)\Delta\overline{z} = \mu - (\overline{u} - u_\mu)\overline{z}_\mu. \tag{6d}$$

Once the search direction $(\Delta u, \Delta\lambda, \Delta\underline{z}, \Delta\overline{z})$ is computed from (6a)–(6d), the primal-dual variables are updated using a linesearch over this direction that ensures certain 'filter' conditions and the barrier parameter $\mu$ is decreased. These are further discussed in the remainder of this section.

### 2.1. Secant quasi-Newton Hessian approximations

The interior-point algorithm we propose uses *secant* approximations $B \in \mathcal{L}(U, U^*)$ for the second-derivative $H(u)$ as a way to circumvent the unavailability of the Hessian $H(u)$.

Namely, assuming that the algorithm finished iteration $k$ and updated the primal-dual iteration variables to $(u_{k+1}, \lambda_{k+1}, \underline{z}_{k+1}, \overline{z}_{k+1})$, the operator $B_{k+1}$ approximates $H$ at iteration $k+1$ by satisfying (i) a secant equation of the form $B(u_{k+1})s_k = g_k$, where $s_k = u_{k+1} - u_k \in U$ and $g_k = l(u_{k+1}, \lambda_{k+1}, \underline{z}_{k+1}, \overline{z}_{k+1}) - l(u_k, \lambda_{k+1}, \underline{z}_{k+1}, \overline{z}_{k+1}) \in U^*$, (ii) a symmetry condition, and (iii) a minimum departure from the previous operator approximation. We refer the reader to [58] for a judicious derivation of various quasi-Newton formulas in Hilbert spaces. Here we use the BFGS operator formula for the Hessian in the following pointwise expression:

$$B_{k+1}v_1v_2 = B_kv_1v_2 - \frac{B_ks_kv_1 \cdot B_ks_kv_2}{B_ks_ks_k} + \frac{g_kv_1 \cdot g_kv_2}{g_ks_k}, \quad \forall\, v_1, v_2 \in U. \tag{7}$$

As pointed out in Section 1.1, $B_kv_1v_2$ and $g_kv_1$ denote (the real number) $B_k(v_1)v_2$ and $g_k(v_1)$, respectively; also '$\cdot$' denotes the multiplication of real numbers. The recursion (7) can be also given in terms of dyadic products (e.g. see [3]), however, we prefer the pointwise expression for compactness.

*Limited-memory* secant BFGS formulas are used in computational practice since full-memory counterparts such as (7) require solving large dense linear systems and thus, may quickly become computationally intractable for large-scale problems. We use a limited-memory BFGS formula that generalizes the compact representation of Byrd et. al. [10] for Hilbert spaces. Namely we store only the last $l$ pairs $(s, g)$, where the approximation 'memory' $l$ is a positive integer $6 \le l \le 24$, and use the compact operator representation

$$B_k = B_0 + F_k^* \mathrm{Q}_k^{-1} F_k, \tag{8}$$

where $B_0 \in \mathcal{L}(U, U^*)$ is an initial approximation for the second-derivative $H$, usually a multiple of the identity, and $F_k \in \mathcal{L}(U, \mathbb{R}^{2l})$ is such that

$$F_k u = \left[B_0 s_{k-l+1}u;\, \ldots;\, B_0 s_k u;\, g_{k-l+1}u;\, \ldots;\, g_k u\right] \in \mathbb{R}^{2l}. \tag{9}$$

Also the matrix $\mathrm{Q}_k = \begin{bmatrix} \mathrm{N}_k & \mathrm{L}_k \\ \mathrm{L}_k^{\mathrm{t}} & -\mathrm{D}_k \end{bmatrix} \in \mathbb{R}^{2l\times 2l}$ has the blocks

$$\mathrm{D}_k = \mathrm{diag}(g_{k-l+1}s_{k-l+1}, \ldots, g_ks_k) \in \mathbb{R}^{l\times l},$$

$$\mathrm{L}_k = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ g_{k-l+1}s_{k-l+2} & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ g_{k-l+1}s_{k-1} & g_{k-l+2}s_{k-1} & g_{k-l+3}s_{k-1} & \ldots & 0 & 0 \\ g_{k-l+1}s_k & g_{k-l+2}s_k & g_{k-l+3}s_k & \ldots & g_{k-1}s_k & 0 \end{bmatrix} \in \mathbb{R}^{l\times l}, \quad \text{and}$$

$$\mathrm{N}_k = \begin{bmatrix} B_0s_{k-l+1}s_{k-l+1} & B_0s_{k-l+1}s_{k-l+2} & \ldots & B_0s_{k-l+1}s_k \\ B_0s_{k-l+2}s_{k-l+1} & B_0s_{k-l+2}s_{k-l+2} & \ldots & B_0s_{k-l+2}s_k \\ \vdots & \vdots & \ddots & \vdots \\ B_0s_ks_{k-l+1} & B_0s_ks_{k-l+2} & \ldots & B_0s_ks_k \end{bmatrix} \in \mathbb{R}^{l\times l}.$$

### 2.2. Conceptual primal-dual linesearch IPM method

To facilitate the presentation and the derivation of the proposed method, we present the skeleton of the infinite-dimensional conceptual primal-dual IPM algorithm in Algorithm 1. The remainder of this section elaborates on the constituents of the algorithm, while the following section provides the discretization details that are needed to derive a 'discrete' numerical algorithm that is suitable to a computer implementation.

**Algorithm 1** Skeleton of a linesearch IPM method that uses BFGS approximations of the second-derivatives

1: **Input:** An initial point $u_0$ strictly feasible with respect to the bound constraints, initial barrier parameter $\mu_0$, initial multiple $\sigma_0$ of the identity operator $I$, and stopping tolerance $\epsilon_{tol}$.
2: Let $\mu = \mu_0$, $B0 = \sigma_0 I$.
3: Initialize the initial 'dual' variables: $\underline{z}_0 = u_0 - ulow$, $\overline{z}_0 = uupp - u_0$, and $\lambda_0$ as the solution to the least-square problem (11).
4: **for** $k = 0, 1, \ldots$ **do**
5: If $e^{rr}(u_k, \lambda_k, \underline{z}, \overline{z}; 0) < \epsilon_{tol}$, then return optimal solution.
6: If $e^{rr}(u_k, \lambda_k, \underline{z}, \overline{z}; \mu) < 10\,\epsilon_{tol}$, reduce $\mu$ to $0.4\mu$ and continue the loop 4.
7: Compute search direction $(\Delta u, \Delta\lambda, \Delta\underline{z}, \Delta\overline{z})$ by solving linear system (6a) at incumbent iteration $(u_k, \lambda_k, \underline{z}_k, \overline{z}_k)$ with $Bk$ given by (8) as an approximation for $H$ at iteration $k$.
8: Perform backtracking filter linesearch to find primal steplengths $\alpha_p$ and $\alpha_d$.
9: Update: $u_{k+1} = u_k + \alpha_p \Delta u$, $\underline{z}_{k+1} = \underline{z}_k + \alpha_d \Delta\underline{z}$, and $\overline{z}_{k+1} = \overline{z}_k + \alpha_d \Delta\overline{z}$.
10: If $e^{rr}_{feas}(u_k) < 10^{-6}$, take $\lambda_{k+1}$ as the solution to the least-squares problem (11), else $\lambda_{k+1} = \lambda_k + \alpha_d \Delta\lambda$.
11: Compute $s_{k+1}$ and $g_{k+1}$ and update the compact representation (8) for $Bk + 1$.
12: **end for**

### 2.3. Stopping criteria

The stopping criteria of Algorithm 1 are based on the norms of the residuals of the optimality system (4a)–(4c). This system has three components: (i) stationary Equations (4a) (also known as dual feasibility equations) posed in $U^*$, (ii) primal feasibility Equations (4b) posed in $\mathbb{R}^m$, and (iii) complementarity Equations from (4c) posed pointwise (almost everywhere). Obviously, the errors/residuals for (i) and (ii) should be measured using $\|\cdot\|_{U^*}$ and one of the finite-dimensional norms (we choose $\|\cdot\|_\infty$), respectively. On the other hand, the pointwise nature of (4c) requires the use of one of the $L^p$ norms ($1 \le p \le \infty$). Using $\|\cdot\|_{L^\infty}$ would be ideal for these pointwise equations; however, we remark that without additional regularity assumptions, the product of two functions from $U \subseteq L^2(\Omega)$ for bounded $\Omega$ generally lies in $L^1(\Omega)$ and not in any of the more regular $L^p(\Omega)$ for $2 \le p \le \infty$. Therefore we use the weaker but mathematically rigorous $\|\cdot\|_{L^1}$ for (4c). Based on the above considerations, we propose the following error measures:

$$e^{rr}_{stat}(u, \lambda, \underline{z}, \overline{z}) = \frac{\|Df(u) + Da(u)^*\lambda - E(\underline{z}) + E(\overline{z})\|_{U^*}}{s_d}, \tag{10a}$$

$$e^{rr}_{feas}(u) = \|a(u)\|_\infty, \quad \text{and} \tag{10b}$$

$$e^{rr}_{compl}(u, \lambda, \underline{z}, \overline{z}; \mu) = \max\{\|\mu - (u - \underline{u})\underline{z}\|_{L^1}, \|\mu - (\overline{u} - u)\overline{z}\|_{L^1}\}/s_c. \tag{10c}$$

The scalars $s_c$ and $s_d$ above are defined by

$$s_c = \max\left\{\frac{1}{2}\|\underline{z}\|_{L^2} + \frac{1}{2}\|\overline{z}\|_{L^2}, s_{\max}\right\}/s_{\max},$$

$$s_d = \max\left\{\frac{1}{3}\|\lambda\|_\infty + \frac{1}{3}\|\underline{z}\|_{L^2} + \frac{1}{3}\|\overline{z}\|_{L^2}, s_{\max}\right\}/s_{\max}.$$

They have the role of 'relaxing' the accuracy of the optimality conditions when the dual variables are large in magnitude [61]. The threshold $s_{\max}$ is taken to be 100 in our implementation. The overall optimality error is defined as

$$e^{rr}(u, \lambda, \underline{z}, \overline{z}; \mu) = \max\left\{e^{rr}_{stat}(u, \lambda, \underline{z}, \overline{z}), e^{rr}_{feas}(u), e^{rr}_{compl}(u, \lambda, \underline{z}, \overline{z}; \mu)\right\}.$$

In particular, $e^{rr}(u, \lambda, \underline{z}, \overline{z}; 0)$ is an appropriate measure of the optimality[3] of $(u, \lambda, \underline{z}, \overline{z})$ with respect to the original problem (1a)–(1b).

### 2.4. Least-squares-based (LSQ) computation of the duals

The dual variables $\lambda_k$ are computed as the solution of a linear least-squares problem in step 1 ($k = 0$) and in step 1 ($k > 0$) of Algorithm 1. More specifically, $\lambda_k$ is chosen to minimize the norm of the residual of (4a) for given variables $(u_k, \underline{z}_k, \overline{z}_k)$, i.e.

$$\lambda_k = \underset{\lambda}{\operatorname{argmin}} \|Df(u_k) + Da(u_k)^*\lambda - E(\underline{z}_k) + E(\overline{z}_k)\|^2_{U^*}. \tag{11}$$

This LSQ updating strategy for $\lambda$ is known to increase the performance of quasi-Newton methods over the Newton-like alternative update from step 1 of Algorithm 1, especially when the feasibility error $e^{rr}_{feas}(u_k)$ is small [61].

### 2.5. Linesearch procedure

To enforce convergence from arbitrary remote points for general problems (1a)–(1b) (e.g. possibly nonconvex and nonlinear), we use the Wächter–Biegler filter linesearch [59,60]. This linesearch algorithm has the combined goal of reducing the log-barrier objective $\psi_\mu(u_k)$ and the infeasibility $\theta(u_k) = \|a(u_k)\|_\infty$ at each iteration $k$ of Algorithm 1. It may happen that only $\psi_\mu(u_k)$ or $\theta(u_k)$ decrease along the search direction at iteration $k$; to prevent cycling between such points a *filter* set is maintained and updated to contain pairs of $(\theta, \psi)$ values that are prohibited for any subsequent updates of the $u$ variables in step 1 of Algorithm 1. We refer the reader to [61] for details of the linesearch algorithms; here we focus on the nontrivial issues that arise in the generalization of this filter linesearch algorithm to the infinite-dimensional problem (1a)–(1b). Such issues occur in formalizing the sufficient-decrease Armijo rule for $\psi_\mu$ because the log-barrier terms of $\psi_\mu$ are not differentiable (Fréchet and even Gâteaux) for the most common functional spaces

(e.g. $U = L^2$ or $U = H^1$) [52]. For this reason, we work with directional variations of the log-barrier objective of (2).

By denoting $\rho(u) = \int_\Omega \ln(u(x))\,\mathrm{d}x$, the variation of $\rho$ at $u$ along the direction $d \in U$ is $\delta\rho(u;d) = \lim_{h\to 0}(\rho(u+hd) - \rho(u))/h$. Similarly to obtaining (3), one can show that $\delta\rho(u;d) = \int_\Omega \frac{\mathrm{d}(x)}{u(x)}\,\mathrm{d}x$. Since $f$ is Fréchet differentiable, the directional variation of $\psi_\mu$ is given by

$$\delta\psi_\mu(u;d) = Df(u)d - \mu\int_\Omega \frac{\mathrm{d}(x)}{u(x)-\underline{u}(x)}\,\mathrm{d}x - \mu\int_\Omega \frac{\mathrm{d}(x)}{\overline{u}(x)-u(x)}\,\mathrm{d}x. \tag{12}$$

Consequently, the Armijo rule in our setup requires that the steplength $\alpha$ satisfies the sufficient decrease condition

$$\psi_\mu(u_k + \alpha\Delta u) \le \psi_\mu(u_k) + \eta_\psi\alpha\cdot\delta\psi_\mu(u_k;\Delta u)$$

in order to be acceptable, where the algorithm parameter $\eta_\psi$ is taken 0.001.

## 3. Discretization using finite element spaces

In this section, we derive a computer implementable variant based on finite element discretizations of the infinite-dimensional quasi-Newton IPM introduced in the previous section. Broadly speaking, we follow an *optimize-then-discretize* approach (see [26, Chapter 3]) and provide discretizations for the algorithm's constituents parts (e.g. variables, vector- and matrix-based formulas for BFGS updates, linearizations, stopping criteria, etc.) in line with previous works in PDE-constrained optimization (e.g. in [8,9,22,42,49]). A notable exception is the discretization of the smoothed complementarity equations from (4c), which poses some difficulties and, to the best of our knowledge, has not been addressed before. In particular, in Section 3.2.1 we derive an 'economy' discretized form for (4c) that is effective computationally and non-intrusive from an implementation perspective. The salient idea of our tedious yet simple derivation is to ensure *consistency* of the discretized algorithm with the inner products of $U$ and $U^*$ (and their dual pairing) as a first necessary step towards mesh independence. Notably, in Section 3.6 we point out that incorrect discretizations of the dual variables are likely to appear when off-the-shelf NLP solvers are used as-is directly on discretized PDE-constrained problems. In this respect, the current work can be regarded as a generalization of the discretization approach in [49] to certain problems with inequality constraints (on the control/design variables) in the context of primal-dual IPMs.

### 3.1. Discretization representations

The finite-dimensional discretization Hilbert space $U_h$ for $U$ can consist of piecewise constant or continuous functions defined over a mesh of $\Omega$, as is often the case of the finite element method. Then a finite-dimensional approximation $u_h \in U_h$ of $u \in U$ is

$$u_h(x) = \sum_{i=1}^{n} \mathrm{u}_i\phi_i(x), \tag{13}$$

where $\phi_i \in U_h$ for all $i \in \{1,\dots,n\}$, the set $\{\phi_i\}_{i=1}^n$ form a function basis of $U_h$, and $\{\mathrm{u}_i\}_{i=1}^n$ are the expansion coefficients that completely and uniquely determine $u_h$ [7].

Common choices for the finite element basis functions are so that $\phi_i$ are defined piecewise over the elements $\Omega_i$ that partition the domain $\Omega$, as it is the case for Lagrange shape polynomials of various degrees [19]. The form of the basis functions depends on $U$. For example, they can be piecewise constant over the elements $\Omega_i$ for $L^2$ and must be piecewise continuous for $H^1$ and other Sobolev spaces. The discretized form of the infinite-dimensional quasi-Newton IPM described in the previous section can be drastically simplified when the basis functions satisfy

$P_1$: a Kronecker delta property, namely $\phi_i(x_j) = \delta_{ij}$[4] whenever $x_j \in \Omega_j$ for piecewise constant basis functions or $x_j$ is a vertex (node) coordinate of $\Omega_j$ for piecewise continuous basis functions, and

$P_2$: $\sum_{i=1}^{n} \phi_i(x) = 1$ for all $x \in \Omega$.

The consequence of $P_1$ is that $\mathrm{u}_i = u(x_i) = u_h(x_i)$ for all $i \in \{1, \ldots, n\}$, which is used by our algorithm to reduce computational cost and simplify software interfacing.

In a numerical algorithm, the vector $\mathrm{u} = [\mathrm{u}_1; \mathrm{u}_2; \ldots; \mathrm{u}_n]$ serves as a computer encoding (or representation) of $u_h \in U_h$ and, by extension, as a discretization representation of $u \in U$. Similarly, finite-dimensional vector encodings or representations are available for functionals from the dual space $U_h^*$; a typical vector representation of $w_h \in U_h^*$ is in the form of the $\mathbb{R}^n$ vector $\mathrm{w} = [\mathrm{w}_1; \mathrm{w}_2; \ldots; \mathrm{w}_n]$, where $\mathrm{w}_i = w_h(\phi_i)$ [30]. We remind the reader that symbols in roman font denote the vector and matrices discretization representations. of the

The finite-element space $U_h$ and its dual $U_h^*$ equipped with the restriction of the inner products of $U$ and $U^*$ are Hilbert spaces [30]. Furthermore, these inner products have 'finite-dimensional' expressions in terms of the discretization representations [30] in the form

$$\langle u_h, v_h \rangle_{U_h} = \mathrm{u}^{\mathrm{t}} \mathrm{H} \mathrm{v} \quad \text{and} \quad \langle w_h, z_h \rangle_{U_h^*} = \mathrm{w}^{\mathrm{t}} \mathrm{H}^{-1} \mathrm{z}, \tag{14}$$

where the symmetric positive definite matrix $\mathrm{H} \in \mathbb{R}^{n \times n}$ is the so-called 'weight' matrix that depends on the spaces $U$ and $U_h$ and the basis functions used in discretization. For example, for our $U = L^2$ case, one can write that

$$\begin{aligned} \langle u_h, v_h \rangle_{U_h} &= \int_\Omega u_h(x) v_h(x) \, \mathrm{d}x \\ &= \int_\Omega \left( \sum_{i=1}^{n} \mathrm{u}_i \phi_i(x) \right) \left( \sum_{j=1}^{n} \mathrm{v}_j \phi_j(x) \right) \mathrm{d}x \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{u}_i \mathrm{v}_j \int_\Omega \phi_i(x) \phi_j(x) \, \mathrm{d}x = \mathrm{u}^{\mathrm{t}} \mathrm{H} \mathrm{v}, \end{aligned}$$

and, therefore, H equals the 'mass' matrix M with entries $\mathrm{M}_{ij} = \int_\Omega \phi_i(x) \phi_j(x) \, \mathrm{d}x$, $i, j \in \{1, \ldots, n\}$. When piecewise constant basis functions are used, we remark that M is diagonal.

Similarly, for $U = H^1$ using piecewise continuous basis functions,

$$\begin{aligned}\langle u_h, v_h\rangle_{U_h} &= \int_\Omega (u_h(x)v_h(x) + \nabla u_h(x)\cdot\nabla v_h(x))\,\mathrm{d}x\\ &= \int_\Omega \left(\sum_{i=1}^n \mathrm{u}_i\phi_i(x)\right)\left(\sum_{j=1}^n \mathrm{v}_j\phi_j(x)\right) + \left(\sum_{i=1}^n \mathrm{u}_i\nabla\phi_i(x)\right)\left(\sum_{j=1}^n \mathrm{v}_j\nabla\phi_j(x)\right)\mathrm{d}x\\ &= \sum_{i=1}^n\sum_{j=1}^n \mathrm{u}_i\mathrm{v}_j\int_\Omega \big(\phi_i(x)\phi_j(x) + \nabla\phi_i(x)\cdot\nabla\phi_j(x)\big)\,\mathrm{d}x = \mathrm{u}^{\mathrm{t}}\mathrm{H}\mathrm{v},\end{aligned}$$

showing that $\mathrm{H} = \mathrm{M} + \mathrm{K}$, where the 'stiffness' matrix K has entries $\mathrm{K}_{ij} = \int_\Omega \nabla\phi_i(x)\cdot\nabla\phi_j(x)\,\mathrm{d}x$, for all $i, j \in \{1,\ldots,n\}$.

In many places we will use vector representations of constant functions. We denote the vector representation of the function identically equal to 1 by $\widetilde{\mathrm{e}}$ and remark that (13) can be used to show that, in general, $\widetilde{\mathrm{e}} = \mathrm{M}^{-1}\left[\int_\Omega \phi_1(x)\,\mathrm{d}x;\ \int_\Omega \phi_2(x)\,\mathrm{d}x;\ \ldots;\ \int_\Omega \phi_n(x)\,\mathrm{d}x\right]$. In particular, when $\mathrm{P}_1$ and $\mathrm{P}_2$ hold, it can be easily proved that $\widetilde{\mathrm{e}} = \mathrm{e}$, where e denotes the vector of all ones.

Hereafter we will work with a generic Hilbert space $U$ and a generic discretization space $U_h$ and (only) make use of the inner product weight matrix H and the mass matrix M in the derivation of the discretized quasi-Newton IPM method. Properties $\mathrm{P}_1$ and $\mathrm{P}_2$ are generally *not* assumed for $U_h$ unless stated otherwise in text.

Similarly to the representations of elements of $U^*$, the Euclidean linear functionals $\lambda \in \mathbb{R}^{m*}$ have vector representations in the form of $\lambda = [\lambda(\mathrm{e}_1), \lambda(\mathrm{e}_2), \ldots, \lambda(\mathrm{e}_m)]$, where $\{\mathrm{e}_j\}_{j=1}^m$ is the standard orthonormal basis of $\mathbb{R}^m$ (i.e. the $i$th entry of vector $\mathrm{e}_j$ is given by $\delta_{ij}$); also, we remark that $\lambda(\mathrm{v}) = \lambda^t\mathrm{v}$ for any $\mathrm{v} \in \mathbb{R}^m$. The discretized counterparts of the objective $f$ and the constraints $a$ are $\mathrm{f} : \mathbb{R}^n \to \mathbb{R}$ and $\mathrm{a} : \mathbb{R}^n \to \mathbb{R}^m$ such that $\mathrm{f}(\mathrm{u}) = f(u_h)$ and $\mathrm{a}(\mathrm{u}) = a(u_h)$. We next state some basic results regarding the discretization of integral forms and of linear functionals from $U^*$.

**Lemma 3.1:** *Given $u_h, v_h \in U_h$, the integral form $\langle u_h, v_h\rangle_{L^2} = \int_\Omega u_h(x)v_h(x)\,\mathrm{d}x$ equals* $\mathrm{u}^{\mathrm{t}}\mathrm{M}\mathrm{v}$. *As an immediate consequence,* $\int_\Omega u_h(x)\,\mathrm{d}x = \widetilde{\mathrm{e}}^{\mathrm{t}}\mathrm{M}\mathrm{u}$.

***Proof:*** We write that $\int_\Omega u_h(x)v_h(x)\,\mathrm{d}x = \sum_{i=1}^n\sum_{j=1}^n \mathrm{u}_i\mathrm{v}_j\int_\Omega \phi_i(x)\phi_j(x)\,\mathrm{d}x = \mathrm{u}^{\mathrm{t}}\mathrm{M}\mathrm{v}$. ■

**Lemma 3.2:** *For any $w_h \in U_h^*$ and $u \in U$, $w_h(u_h) = \mathrm{u}^{\mathrm{t}}\mathrm{w}$.*

***Proof:*** We observe that $w_h(u_h) = w_h(\sum_{i=1}^n \mathrm{u}_i\phi_i) = \sum_{i=1}^n \mathrm{u}_i w_h(\phi_i) = \sum_{i=1}^n \mathrm{u}_i\mathrm{w}_i = \mathrm{u}^{\mathrm{t}}\mathrm{w}$. ■

### 3.2. Vector representations for functionals and derivatives

For a function $f : U \to \mathbb{R}$ with derivative $Df(u) \in U^*$, we will use $D\mathrm{f}(\mathrm{u})$ to denote the vector representation of $Df(u_h)$. As a consequence of Lemma 3.2, we have that

$$Df(u_h)v_h = D\mathrm{f}(\mathrm{u})^{\mathrm{t}}\mathrm{v}, \tag{15}$$

where v is the representation of $v_h$. For example, when $f(u) = \int_\Omega u(x)\,\mathrm{d}x$, it can be easily proved that $Df(u)$ is such that $Df(u)v = \int_\Omega v(x)\,\mathrm{d}x$ for all $v \in U$; therefore, since $Df(\mathrm{u})^{\mathrm{t}}\mathrm{v} = Df(u_h)v_h = \widetilde{\mathrm{e}}^{\mathrm{t}}\mathrm{Mv}$ by Lemma 3.1, we obtain that $Df(\mathrm{u}) = \mathrm{M}\widetilde{\mathrm{e}}$.

Similarly to $Df(\mathrm{u})$, we introduce the (matrix) representation of the derivative $Da(u)$ of $a : U \to \mathbb{R}^m$ as

$$Da(\mathrm{u}) = \begin{bmatrix} Da_1(\mathrm{u})^{\mathrm{t}} \\ \vdots \\ Da_m(\mathrm{u})^{\mathrm{t}} \end{bmatrix},$$

where $Da_j(\mathrm{u})$ is the representation of the derivative of the $j$th function component $a_j$ of $a$ ($j \in \{1, \ldots, m\}$) defined as in (15).

The following lemmas derive discretization representations for various quantities employed by the algorithm that are consistent with the inner products of $U$ and $U^*$.

**Lemma 3.3:** *We have that* $Da(u_h)v_h = Da(\mathrm{u})\mathrm{v}$. *Also, the vector representation of* $Da(u_h)^*\lambda \in U_h^*$ *from the optimality conditions (4a) is given by* $Da(\mathrm{u})^{\mathrm{t}}\lambda$.

***Proof:*** The first relation follows from (15) and the above expression of $Da(\mathrm{u})$.

To prove the second statement we remark that the representation of $Da(u_h)^*\lambda$, which we denote by w, should satisfy $\mathrm{w}^{\mathrm{t}}\mathrm{v} = [Da(u_h)^*\lambda](v_h)$ for all $v_h \in U_h$. On the other hand, for any $v_h \in U_h$ one can write

$$(Da(\mathrm{u})^{\mathrm{t}}\lambda)^{\mathrm{t}}\mathrm{v} = \lambda^{\mathrm{t}}Da(\mathrm{u})\mathrm{v} = \lambda(Da(u_h)v_h) = [Da(u_h)^*\lambda](v_h),$$

where the last equality is given by the definition of an adjoint operator. This shows that $\mathrm{w} = Da(\mathrm{u})^{\mathrm{t}}\lambda$ and concludes the proof. ■

**Lemma 3.4:** *The vector representation of* $E(z_h)$ *from (4a) is* Mz.

***Proof:*** To prove the statement we write $E(z_h)v_h = \int_\Omega z_h(x)v_h(x)\,\mathrm{d}x = \mathrm{z}^{\mathrm{t}}\mathrm{Mv}$, for all $v_h \in U_h$, where the last equality follows from Lemma 3.1. ■

**Lemma 3.5:** *Given* $u_h, v_h \in U_h$, *the vector representation of* $G(u_h)v_h \in U_h$ *appearing in (6c) and (6d) is* $\mathrm{M}^{-1}[\mathrm{u}^{\mathrm{t}}\mathrm{C}_1\mathrm{v}; \ldots; \mathrm{u}^{\mathrm{t}}\mathrm{C}_n\mathrm{v}]$, *where* C *denotes the mass tensor given by* $\mathrm{C}_{ijk} = \int_\Omega \phi_i(x)\phi_j(x)\phi_k(x)\,\mathrm{d}x$ *and the matrix* $\mathrm{C}_k$ *is the kth frontal slice of* C, *namely* $(\mathrm{C}_k)_{ij} = \mathrm{C}_{ijk}$, $k \in \{1, \ldots, n\}$.

***Proof:*** Let $\mathrm{z} \in \mathbb{R}^n$ be the vector representation of $[G(u_h)v_h] = u_h v_h$. We remark that $u_h(x)v_h(x) = \sum_{i=1}^n \mathrm{z}_i\phi_i(x)$ and take the $L^2$ product of the two sides of this identity with

basis functions $\phi_k$ for each $k \in \{1, \dots, n\}$ to obtain

$$\int_\Omega \left(\sum_{i=1}^n \mathrm{u}_i \phi_i(x)\right)\left(\sum_{j=1}^n \mathrm{v}_j \phi_j(x)\right)\phi_k(x)\,\mathrm{d}x = \int_\Omega \sum_{i=1}^n \mathrm{z}_i \phi_i(x)\phi_k(x)\,\mathrm{d}x, \quad \forall\, k \in \{1, \dots, n\}.$$

Furthermore, this can be also written equivalently as

$$\sum_{i=1}^n \sum_{j=1}^n \mathrm{u}_i \mathrm{v}_j \int_\Omega \phi_i(x)\phi_j(x)\phi_k(x)\,\mathrm{d}x = \sum_{i=1}^n \mathrm{z}_i \int_\Omega \phi_i(x)\phi_k(x)\,\mathrm{d}x, \quad \forall\, k \in \{1, \dots, n\},$$

which in matrix form is exactly $\mathrm{u}^{\mathrm{t}}\mathrm{C}_k\mathrm{v} = \mathrm{Mz}$ for all $k \in \{1, \dots, n\}$. This proves the conclusion. ■

### 3.2.1. Discretization of optimality conditions

Lemmas 3.3 and 3.4, and Equation (15) allows to write the infinite-dimensional optimality Equations (4a) and (4b) as the discretized Equations (16a) and (16b) below. Furthermore, since the other two infinite-dimensional optimality equations from (4c) can be written as $G(u - \underline{u})\underline{z} = \mu$ and $G(\overline{u} - u)\overline{z} = \mu$, Lemma 3.5 implies that the discretization of these two equations is given by (16c) below. We have obtained the following matrix-form discretization of optimality conditions:

$$D\mathrm{f}(\mathrm{u}) + D\mathrm{a}(\mathrm{u})^{\mathrm{t}}\lambda - \mathrm{M}\underline{\mathrm{z}} + \mathrm{M}\overline{\mathrm{z}} = 0, \tag{16a}$$

$$\mathrm{a}(\mathrm{u}) = 0, \tag{16b}$$

$$\mathrm{M}^{-1}\begin{bmatrix} (\mathrm{u} - \underline{\mathrm{u}})^{\mathrm{t}}\mathrm{C}_1\underline{\mathrm{z}} \\ \vdots \\ (\mathrm{u} - \underline{\mathrm{u}})^{\mathrm{t}}\mathrm{C}_n\underline{\mathrm{z}} \end{bmatrix} = \mu\widetilde{\mathrm{e}}, \quad \mathrm{M}^{-1}\begin{bmatrix} (\overline{\mathrm{u}} - \mathrm{u})^{\mathrm{t}}\mathrm{C}_1\overline{\mathrm{z}} \\ \vdots \\ (\overline{\mathrm{u}} - \mathrm{u})^{\mathrm{t}}\mathrm{C}_n\overline{\mathrm{z}} \end{bmatrix} = \mu\widetilde{\mathrm{e}}. \tag{16c}$$

A couple of remarks are in order. When piecewise constant elements are used, i.e. our $L^2$ case, one can easily prove that the tensor C has a simple form, namely, the frontal slices $\mathrm{C}_k$ are equal to the (diagonal) mass matrix M. As a result, the Equations (16c) have a simplified form (see (17c) below) that requires only pointwise vector multiplications. This is ideal both computationally and from a software interface perspective since the optimization solver does not need the application of C.

For more general finite element discretizations, the tensor C has a more onerous form [12], which leads to nontrivial complications. For example, C would appear in the Newton linearizations (i.e. the correspondent of Equation (21c)) of (16c) and, as a result, the quasi-Newton linearization system (i.e. the correspondent of (21a)–(21c)) would become much more cumbersome to solve, especially in parallel; in contrast, the linearization (21a)–(21c) that we derive later in the paper involve only diagonal matrices. Another complication consists of the fact that C is not routinely formed for the PDE state and adjoint solves as these only require the mass and stiffness matrices; as a result, requiring C in the optimization phase may impede the use of the optimization solver since the practitioners need to dedicate additional effort for making the tensor C available for optimization.

All the aforementioned complications can be circumvented by using a projection-based discretization of the infinite-dimensional complementarity equations from (4c).

First let use consider the prototype infinite-dimensional equation $v_1(x)v_2(x) = \mu$ a.e. on $\Omega$, where $v_1, v_2 \in U$ and $\mu \in \mathbb{R}$, which is the form present in (4c). Assuming the Kronecker property $\mathrm{P}_1$, we then define the projection $P_{U_h} : U \to U_h$ such that for any $u \in U$, the function $w_h := P_{U_h}(u)$ is the (unique) element of $U_h$ for which $w_h(x_j) = u(x_j)$, for all $j \in \{1, \dots, n\}$. Furthermore, let us consider the projection of prototype equation, namely $P_{U_h}(v_1 v_2) = \mu$. Not only the projected equation is in $U_h$ but also one can equivalently write $v_{1h}(x_j)v_{2h}(x_j) = \mu$ for all $j \in \{1, \dots, n\}$ and, as a result, obtain the discretization $\mathrm{v}_1\mathrm{v}_2 = \mu\mathrm{e}$ for the prototype equation. Consequently, we propose and use throughout the paper the following 'economy' discretization of the infinite-dimensional optimality conditions:

$$Df(u) + Da(u)^{\mathrm{t}}\lambda - M\underline{z} + M\overline{z} = 0, \tag{17a}$$

$$a(u) = 0, \tag{17b}$$

$$(u - \underline{u})\underline{z} = \mu e, \quad (\overline{u} - u)\overline{z} = \mu e. \tag{17c}$$

### 3.3. Stopping criteria discretization

The discretizations of the error measures from (10a) are obtained by replacing the elements in $U$ and $U^*$ with their vector representations and the norms $U$ and $U^*$ with their finite-dimensional algebraic expressions given by (14). In particular, the stationary error is given by

$$\mathrm{e}^{rr}_{stat} = \frac{1}{s_d}\left[\left(Df(u) + Da(u)^{\mathrm{t}}\lambda - M\underline{z} + M\overline{z}\right)^{\mathrm{t}} H^{-1}\left(Df(u) + Da(u)^{\mathrm{t}}\lambda - M\underline{z} + M\overline{z}\right)\right]^{1/2}$$

and we remark that its computation requires the application of $H^{-1}$ and M.

The computation of $e^{rr}_{compl}$ from (10c) requires evaluating terms in the form of $\|v_{1h}v_{2h} - \mu\|_{L^1}$ for $v_{1h}, v_{2h} \in U_h$, which is not necessarily straightforward for general approximation spaces $U_h$. Our general approach is based on the observation that one can write formally that $\|v_{1h}v_{2h} - \mu\|_{L^1} = \langle |v_{1h}v_{2h} - \mu|, 1\rangle_{L^2}$ where the left operand in the last inner product denotes the absolute value function and the right operand denotes the function of all ones. Since $|v_{1h}v_{2h} - \mu|$ is not necessarily in $U_h$, a convenient and yet consistent error estimator can be obtained under $\mathrm{P}_1$ and $\mathrm{P}_2$ properties by projecting this term in $U_h$ based on the considerations said before the economy discretization (17a)–(17c). More specifically, we propose to use $\langle P_{U_h}(|v_{1h}v_{2h} - \mu|), 1\rangle_{L^2} = \mathrm{e}^t M|\mathrm{v}_1\mathrm{v}_2 - \mu\mathrm{e}| := \|\mathrm{v}_1\mathrm{v}_2 - \mu\|_{1\mathrm{M}}$, where $|\mathrm{v}_1\mathrm{v}_2 - \mu\mathrm{e}|$ is the vector of absolute values of the vector $\mathrm{v}_1\mathrm{v}_2 - \mu\mathrm{e}$. This estimator is essentially the finite-dimensional norm $\|\cdot\|_1$ weighted by the mass matrix M. As a consequence, an economy expression for $e^{rr}_{compl}$ is

$$e^{rr}_{compl} = \max\{\|\mu e - (u - \underline{u})\underline{z}\|_{1M}, \|\mu e - (\overline{u} - u)\overline{z}\|_{1M}\}/s_c.$$

Finally, we mention that the weighted two-norm $\|\mathrm{v}\|_{2\mathrm{M}} = \sqrt{\mathrm{v}^{\mathrm{t}}\mathrm{M}\mathrm{v}}$ worked as well as $\|\cdot\|_{1\mathrm{M}}$ in all of our numerical experiments.

### 3.3.1. Matrix representations for BFGS formulas

Since for $f : U \to \mathbb{R}$, $D^2f(u)$ is in $\mathcal{L}(U, U^*)$, we target a matrix representation $D^2\mathrm{f}(\mathrm{u})$ of $D^2f(u_h)$ such that $D^2f(u_h)v_h = D^2\mathrm{f}(\mathrm{u})\mathrm{v}$ for all $v_h \in U_h$. As before, u and v are representations of $u_h$ and $v_h$, respectively. The Hessian application $D^2\mathrm{f}(\mathrm{u})\mathrm{v}$ can be computed efficiently in general by using second-order adjoint sensitivities (see, for example, [43,44,51]). However, the Hessian computation requires nontrivial additional development effort, which can be circumvented by using secant approximations formulas, such as the BFGS formula derived below.

We observe that $D^2f(u_h)v_hw_h = (D^2\mathrm{f}(\mathrm{u})\mathrm{v})^{\mathrm{t}}\mathrm{w} = \mathrm{v}^{\mathrm{t}}D^2\mathrm{f}(\mathrm{u})^{\mathrm{t}}\mathrm{w}$ and similarly, that $D^2f(u_h)w_hv_h = \mathrm{w}^{\mathrm{t}}D^2\mathrm{f}(\mathrm{u})^{\mathrm{t}}\mathrm{v}$. Since $D^2f(u_h)$ is symmetric, we have that $\mathrm{v}^{\mathrm{t}}D^2\mathrm{f}(\mathrm{u})^{\mathrm{t}}\mathrm{w} = \mathrm{w}^{\mathrm{t}}D^2\mathrm{f}(\mathrm{u})^{\mathrm{t}}\mathrm{v}$ for all v and w, which shows that the matrix $D^2\mathrm{f}(\mathrm{u})$ is symmetric. It is illustrative to compute the representation of $D^2f(u)$ for $f(u) = \frac{1}{2}\|u\|^2_{L^2}$. Since one can easily verify that $D^2f(u)vw = \langle v, w\rangle_{L^2} = E(v)w$, Lemma 3.4 implies that $D^2\mathrm{f}(\mathrm{u}) = \mathrm{M}$; also, $D^2\mathrm{f}(\mathrm{u}) = \mathrm{H}$ when $f(u) = \frac{1}{2}\|u\|^2_{H^1}$. Similarly, the representations B of the secant operators $B \in \mathcal{L}(U, U^*)$ from (7) and (8) should satisfy for all $v_h \in U_h$ that

$$\mathrm{Bv} = Bv_h. \tag{18}$$

For a representation $(\mathrm{u}_k, \lambda_k, \underline{\mathrm{z}}_k, \overline{\mathrm{z}}_k)$ of the primal-dual variables, let $\mathrm{s}_k = \mathrm{u}_{k+1} - \mathrm{u}_k$; also, let $\mathrm{g}_k = \mathrm{l}(\mathrm{u}_{k+1}, \lambda_k, \underline{\mathrm{z}}_k, \overline{\mathrm{z}}_k) - \mathrm{l}(\mathrm{u}_k, \lambda_k, \underline{\mathrm{z}}_k, \overline{\mathrm{z}}_k)$ be the representation of $g_k$ from Section 2.1, where l is the discretization of $l$ from (4a), namely, $\mathrm{l}(\mathrm{u}_k, \lambda_k, \underline{\mathrm{z}}_k, \overline{\mathrm{z}}_k) = \mathrm{l}(u_k, \lambda_k, \underline{z}_k, \overline{z}_k)$.

**Lemma 3.6:** *The matrix representation* $\mathrm{B}_k$ *of* $B_k$ *given by the recursion (7) is*

$$\mathrm{B}_{k+1} = \mathrm{B}_k - \frac{\mathrm{B}_k\mathrm{s}_k\mathrm{s}_k^{\mathrm{t}}\mathrm{B}_k^{\mathrm{t}}}{\mathrm{s}_k^{\mathrm{t}}\mathrm{B}_k\mathrm{s}_k} + \frac{\mathrm{g}_k\mathrm{g}_k^{\mathrm{t}}}{\mathrm{g}_k^{\mathrm{t}}\mathrm{s}_k}. \tag{19}$$

*Furthermore, the representation* $\mathrm{B}_k$ *of* $B_k$ *from (8) has the compact representation*

$$\mathrm{B}_k = \mathrm{B}_0 + \mathrm{F}_k^{\mathrm{t}}\mathrm{Q}_k^{-1}\mathrm{F}_k, \tag{20}$$

*where* $\mathrm{F}_k^{\mathrm{t}} = [\ \mathrm{B}_0\mathrm{s}_{k-l+1}\ \ldots\ \mathrm{B}_0\mathrm{s}_k\ \ \mathrm{g}_{k-l+1}\ \ldots\ \mathrm{g}_k\ ] \in \mathbb{R}^{n\times 2l}$ *and* $\mathrm{Q}_k$ *is given by (9), with the blocks* $D_k$ *being the diagonal matrix with entries* $\mathrm{g}_{k-l+1}^{\mathrm{t}}\mathrm{s}_{k-l+1}, \ldots, \mathrm{g}_k^{\mathrm{t}}\mathrm{s}_k$,

$$\mathrm{L}_k = \begin{bmatrix} 0 & 0 & \ldots & 0 & 0 \\ \mathrm{g}_{k-l+1}^{\mathrm{t}}\mathrm{s}_{k-l+2} & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathrm{g}_{k-l+1}^{\mathrm{t}}\mathrm{s}_k & \mathrm{g}_{k-l+2}^{\mathrm{t}}\mathrm{s}_k & \cdots & \mathrm{g}_{k-1}^{\mathrm{t}}\mathrm{s}_k & 0 \end{bmatrix},$$

*and*

$$\mathrm{N}_k = \begin{bmatrix} \mathrm{s}_{k-l+1}^{\mathrm{t}}\mathrm{B}_0\mathrm{s}_{k-l+1} & \mathrm{s}_{k-l+1}^{\mathrm{t}}\mathrm{B}_0\mathrm{s}_{k-l+2} & \cdots & \mathrm{s}_{k-l+1}^{\mathrm{t}}\mathrm{B}_0\mathrm{s}_k \\ \mathrm{s}_{k-l+2}^{\mathrm{t}}\mathrm{B}_0\mathrm{s}_{k-l+1} & \mathrm{s}_{k-l+2}^{\mathrm{t}}\mathrm{B}_0\mathrm{s}_{k-l+2} & \cdots & \mathrm{s}_{k-l+2}^{\mathrm{t}}\mathrm{B}_0\mathrm{s}_k \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{s}_k^{\mathrm{t}}\mathrm{B}_0\mathrm{s}_{k-l+1} & \mathrm{s}_k^{\mathrm{t}}\mathrm{B}_0\mathrm{s}_{k-l+2} & \cdots & \mathrm{s}_k^{\mathrm{t}}\mathrm{B}_0\mathrm{s}_k \end{bmatrix}.$$

***Proof:*** Equation (19) follows from applying the infinite-dimensional counterpart (7) to functions $v_h$ in $U_h$ and by using the definition (18) and that $g_ks_k = \mathrm{g}_k^{\mathrm{t}}\mathrm{s}_k$ and $B_ks_ks_k =$

$s_k^t B_k s_k$. It is also relatively easy to prove that $F_k$ in (20) is the matrix representation of $F_k$ from (8). Furthermore, $N_k$ in (20) is simply the matrix $N_k$ in (8) computed over $U_h$ instead of $U$. Identity (20) follows immediately. ■

We remark that the above discretized BFGS formulas are identical to the formulas derived and used for optimization over finite-dimensional spaces [10,35]. However, as we discuss later, a typical choice for the initial approximation $B_0$ is a multiple of M or H (which are the matrix representations in $L^2$ and $H^1$ of the identity operator), while in the finite-dimensional case $B_0$ is typically a multiple of the identity matrix.

### 3.3.2. The BFGS linear systems for IPM search directions

With the matrix compact representation (20) as the approximation for the second derivative $H$ from (6a), the discretized, matrix-form quasi-Newton search direction equations (compare with (6a)) at iteration $k$ are given by

$$(B_0 - F_k^t Q_k^{-1} F_k)\Delta u + Da(u_k)^t \Delta\lambda - M\Delta \underline{z} + M\Delta \overline{z} = -l(u_k, \lambda_k, \underline{z}_k, \overline{z}_k), \tag{21a}$$

$$Da(u_k)\Delta u = -a(u_k), \tag{21b}$$

$$-\overline{Z}_k \Delta u + \overline{U}_k \Delta \overline{z} = \mu e - (\overline{u} - u_k)\overline{z}_k, \quad \underline{Z}_k \Delta u + \underline{U}_k \Delta \underline{z} = \mu e - (u_k - \underline{u})\underline{z}_k, \tag{21c}$$

The matrices $\underline{U}_k$, $\overline{U}_k$, $\underline{Z}_k$, and $\overline{Z}_k$ above are diagonal with positive diagonal entries given by vectors $u_k - \underline{u}$, $\overline{u} - u_k$, $\underline{z}_k$, and $\overline{z}_k$, respectively. The last two linear equations represent formal linearizations of the economy complementarity Equation (17c), but they can be also derived directly from the infinite-dimensional linear Equation (6a) under the Kronecker property. Finally, we mention that linearizations of the mass tensor-based optimality system from the beginning of Section 3.2.1 could be also derived and would be valid should this be of interest to the reader.

### 3.3.3. The discretization of the log-barrier terms

The discretization of the log-barrier function $\psi_\mu(u)$ and its directional variation $\delta\psi_\mu(u; d)$ poses some nontrivial difficulties because of the use of the logarithm function as a barrier function and the presence of rational functions in the expression (12) of $\delta\rho(u; d)$. We layout one possible discretization that is convenient because it is general, however, we do not exclude the possibility of more accurate discretizations for $\psi_\mu(u)$ and $\delta\psi_\mu(u; d)$, especially when specialized for a particular choice of basis functions.

We discretize log-barrier terms of the form $\rho(u) = \int_\Omega \ln(u(x))\,dx$ using a projection approach. Intuitively, discretizing $\int_\Omega \ln(u(x))\,dx$ is the same as discretizing $\int_\Omega v(x)\,dx$ for $v(x) = \ln(u(x))$ with $v_h(x) = \sum_{i=1}^n v_i \phi_i(x)$, i.e. $v_h$ is the projection of $\ln u_h$ in $U_h$. When properties $P_1$ and $P_2$ hold for $U_h$, we have $v_i = \ln u_i$ and thus write

$$\rho(u) = \int_\Omega v(x)\,dx \approx \int_\Omega v_h(x)\,dx = e^t M v = e^t M \ln u := \rho(u),$$

where $\ln u$ denotes $[\ln u_1; \ldots; \ln u_n]$. Similarly, $\delta\rho(u; d) = \int_\Omega d(x)/u(x)\,dx$ can be viewed as $\int_\Omega d(x)v(x)\,dx$ with $v(x) = 1/u(x)$, and, therefore, one can write

$$\delta\rho(u; d) = \int_\Omega d(x)v(x)\,dx \approx \int_\Omega d_h(x)v_h(x)\,dx = d^t M v = d^t M \frac{1}{u} := \delta\rho(u, d),$$

where $\frac{1}{u}$ denotes the vector with entries $\frac{1}{u_1}, \ldots, \frac{1}{u_n}$ and d and v are the vector representations of $d_h$ and of $v_h$, respectively. We observe that the representation $\delta\rho(u; d)$ is gradient of $\rho(u)$ with respect to u along the direction d, namely $\frac{d\rho(u)}{du} d = \delta\rho(u, d)$. This is important since a discrepancy between the two would indicate inconsistent discretizations and likely cause convergence issues. We have obtained the following discretization form for the log-barrier objective and its directional variation.

**Lemma 3.7:** *Given the vector representations* u *and* $\Delta u$ *of the variable u and search direction* $\Delta u$*, the log-barrier objective from (2) has the discretization*

$$\psi_\mu(u) = f(u) - \mu e^t M \ln(u - \underline{u}) + \mu e^t M \ln(\overline{u} - u)$$

*and its variation (12) along direction* $\Delta u$ *can be discretized as*

$$\delta\psi_\mu(u_k; \Delta u) = Df(u_k)^t \Delta u - \mu \Delta u^t M \frac{1}{u - \underline{u}} + \mu \Delta u^t M \frac{1}{\overline{u} - u}.$$

### 3.4. Discretization of the LSQ-based computation of the duals

The discretization $\lambda$ of the solution $\lambda$ of the least-squares problem (11) is

$$\lambda_k = \underset{\lambda}{\operatorname{argmin}} \|Da(u_k)^t \lambda + Df(u_k) - M\underline{z} + M\overline{z})\|^2_{U_h^*},$$

where the norm $\|v\|_{U_h^*} := \|v_h\|_{U_h^*} = \sqrt{v^t H^{-1} v}$ accordingly to (14). Consequently,

$$\lambda_k = \left[Da(u_k) H^{-1} Da(u_k)^t\right]^{-1} Da(u_k) H^{-1} \left(-Df(u_k) + M\underline{z}_k - M\overline{z}_k\right).$$

We remark that $m + 1$ applications of the inverse of H are needed to compute $\lambda_k$, which is computationally tractable in general since $m$ is relatively small.

### 3.5. The discretized quasi-Newton BFGS linesearch IPM algorithm based on derivative representations (QIpmDDe)

At this point we have all the discretization ingredients needed for the discretization of Algorithm 1 using the vector representations for the functional derivatives introduced in Section 3.2. The numerical QIpmDDe algorithm corresponds to substituting the constituent parts of Algorithm 1 (i.e. primal-dual variables, search directions, linesearch procedure, update of dual variables, error measures, quasi-Newton limited-memory formula) with their discretized counterparts that were introduced in this section. The implementation details, such as various parameters of the algorithm, strategies for constructing starting points, and others, are described in [39,61].

### 3.6. Comparison with Euclidean NLP solvers

Here, we make a couple of observations on the use of NLP solvers equipped with the Euclidean inner product for solving the discretization of (1a)–(1b) in the form

$$\min_{\mathrm{u}\in\mathbb{R}^n} \mathrm{f}(\mathrm{u}) \text{ s.t. } \mathrm{a}(\mathrm{u}) = 0 \in \mathbb{R}^m, \quad \underline{\mathrm{u}} \leq \mathrm{u} \leq \overline{\mathrm{u}}.$$

For illustration purposes, we consider the state-of-the-art solver Ipopt and the similar Euclidean solver from HiOp. The log-barrier problem used by these solvers is

$$\min_{\mathrm{u}} \psi_\mu(\mathrm{u}) := \mathrm{f}(\mathrm{u}) - \mu \sum_{i=1}^{n} \ln(\mathrm{u}_i - \underline{\mathrm{u}}_i) - \mu \sum_{i=1}^{n} \ln(\overline{\mathrm{u}}_i - \mathrm{u}_i) \quad \text{s.t. } \mathrm{a}(\mathrm{u}) = 0,$$

for which the following optimality conditions are used:

$$\begin{aligned} \nabla \mathrm{f}(\mathrm{u}_\mu) + J\mathrm{a}(\mathrm{u}_\mu)^{\mathrm{t}} \lambda_\mu - \underline{\mathrm{z}}_\mu + \overline{\mathrm{z}}_\mu &= 0, \\ \mathrm{a}(\mathrm{u}_\mu) &= 0, \\ (\mathrm{u}_\mu - \underline{\mathrm{u}})\underline{\mathrm{z}}_\mu = \mu, \quad (\overline{\mathrm{u}} - \mathrm{u}_\mu)\overline{\mathrm{z}}_\mu &= \mu. \end{aligned}$$

Here, $\nabla \mathrm{f}(\mathrm{u}) \in \mathbb{R}^n$ denotes the gradient of $\mathrm{f} : \mathbb{R}^n \to \mathbb{R}$ and $J\mathrm{a}(\mathrm{u}) \in \mathbb{R}^{m\times n}$ the Jacobian of $\mathrm{a} : \mathbb{R}^n \to \mathbb{R}^m$ at some $\mathrm{u} \in \mathbb{R}^n$, which coincide with the vector representations $D\mathrm{f}(\mathrm{u})$ and $D\mathrm{a}(\mathrm{u})$ used by our discretized QIpmDDe algorithm.

A quick look at the above optimality conditions and the discretized optimality conditions (17a)–(17c) reveals that the former does not employ the correct dual representer for linear operator $E(z)$ defined in Section 2 after Equation (3). We believe this inconsistency is one of the causes of the *mesh dependent* behaviour of the Euclidean solver(s) that we report in Section 6.

Another severe limitation of the Euclidean NLP solvers comes from the use of the Euclidean norms in the stopping criteria, which results in premature termination at inaccurate solution and in an iteration count that changes drastically as the mesh is refined and/or distorted. Intuitively, the Euclidean, inner-product-oblivious approach to treating NLPs defined in Hilbert spaces divorces the space of decision variables from the finite element mesh. As a result, the decision variables become a tuple of real numbers – without a mesh, operations like refinement and coarsening do not make much sense. Furthermore, the inner-product-oblivious approach treats every component e.g. of the stationarity error vector, as equally important, when one would expect that error components corresponding to larger elements on nonuniform finite element meshes should be weighted more heavily than error components corresponding to smaller elements. The inner-product-aware approach of this paper corrects these deficiencies through the incorporation of the correct inner product.

Finally, Euclidean NLP solvers using secant updates generally hard-code $\mathrm{B}_0$ as a multiple of the identity matrix. This choice results in mesh dependent convergence even for elementary problems such as the unconstrained convex quadratic problem

$$\min_{u\in U} \frac{1}{2}\langle u, u\rangle_U + \langle c, u\rangle_U,$$

posed in a generic Hilbert space $U$ with $c \in U$. The discretized problem would consist of minimizing $\frac{1}{2}\mathrm{u}^{\mathrm{t}}\mathrm{H}\mathrm{u} + \mathrm{c}^{\mathrm{t}}\mathrm{H}\mathrm{u}$. A BFGS quasi-Newton algorithm applied to the discretized

**Table 1.** Discretized input problem and internal discretizations for QIpmDDe and a generic Euclidean NLP solver for (22) is illustrated for $U = L^2(\Omega)$ and $U = H^1(\Omega)$.

| | input problem | | | | internal discretizations | | |
|---|---|---|---|---|---|---|---|
| Algorithm | f(u) | a(u) | 1st order derivatives $Df/\nabla f$ and $Da/Ja$ | | $E(z)$ | $B_0$ | inner products |
| $\mathbf{U = L^2(\Omega)}$ | | | | | | | |
| QIpmDDe | $\frac{1}{2}u^tMu$ | $e^tMu$ | Mu | $e^tM$ | Mz | M | $w_1^tM^{-1}w_2$ $v_1^tMv_2$ |
| Euclidean | —"— | —"— | Mu | $e^tM$ | z | I | $v_1^tv_2$ |
| $\mathbf{U = H^1(\Omega)}$ | | | | | | | |
| QIpmDDe | $\frac{1}{2}u^tHu$ | $e^tMu$ | Hu | $e^tM$ | Mz | H | $w_1^tH^{-1}w_2$ $v_1^tMv_2$ |
| Euclidean | —"— | —"— | Hu | $e^tM$ | z | I | $v_1^tv_2$ |

Note: We highlight in red discretizations used by the Euclidean solver that are inconsistent with the underlying Hilbert space $U$.

problem is identical in exact arithmetic with the preconditioned conjugate gradient method for $Hu = -Hc$ using the preconditioner $B_0^{-1}$ [35, Theorem 6.4]. When $B_0 = I$, as it is the case for Euclidean solvers, the iteration count of the quasi-Newton method is determined by the number of distinct eigenvalues of $B_0^{-1}H = H$, which can be large for distorted meshes, for example; on the other hand, QIpmDDe using $B_0 = H$ converges in one iteration since $B_0^{-1}H = I$.

We remark that QIpmDDe works as an Euclidean solver by taking $M = H = I$.

### 3.6.1. An illustrative example

Let us consider the quadratic problem

$$\min_{u\in U} f(u) := \frac{1}{2}\|u\|_U^2 \quad \text{s.t. } a(u) := \int_\Omega u(x)\,dx = 0, \quad \underline{u} \le u \le \overline{u}. \tag{22}$$

Table 1 shows the discretizations of the objective and constraints for $U = L^2(\Omega)$ and $U = H^1(\Omega)$ as well as the internal representations for $E(z)$ and the initial BFGS operator $B_0$ and the discretized form of the inner product(s) used by QIpmDDe and Euclidean NLP solvers. We assume for simplicity that properties $P_1$ and $P_2$ hold for the underlying discretization space $U_h$. The cells highlighted in red indicate discretization inconsistent with the underlying Hilbert space as per the discussion of the previous subsection.

## 4. Parallelization considerations

A parallelization technique for *Euclidean* quasi-Newton IPMs equipped with limited-memory BFGS updates was proposed in [39] based on a specialized symmetric Guassian elimination to solve the IPM linear systems. Essentially (21a)–(21c) can be manipulated to a system that is more amenable to parallel computations, namely to

$$\begin{aligned} \widetilde{B}_k \Delta u + Da(u_k)^t \Delta\lambda &= r_k, \\ Da(u_k)\Delta u &= -a(u_k), \end{aligned} \tag{23}$$

where $\widetilde{B}_k = B_0 + M\underline{U}_k^{-1}\underline{Z}_k + M\overline{U}_k^{-1}\overline{Z}_k - F_k^tQ^{-1}F_k$, and $r_k = -l(u_k, \lambda_k, \underline{z}_k, \overline{z}_k) + \underline{U}_k^{-1}[\mu e - (u_k - \underline{u})\underline{z}_k] + \overline{U}_k^{-1}[\mu e - (u_k - \underline{u})\overline{z}_k]$. Furthermore, assuming the linear system (23) is solved, the rest of the unknowns of (21a)–(21c) are obtained from $\Delta\underline{z} =$

$-\underline{U}_k^{-1}\underline{Z}_k\Delta u_k + \underline{U}_k^{-1}[\mu e - (u_k - \underline{u})\underline{z}_k]$, and $\Delta\overline{z} = \overline{U}_k^{-1}\overline{Z}_k\Delta u_k + \overline{U}_k^{-1}[\mu e - (\overline{u} - u_k)\overline{z}_k]$, both operations involving only basic vector-vector computations that parallelize well.

The parallelization difficulty occurs in solving the linear system (23). The original approach of [39] exploits that $\widetilde{B}_k$ is a low-rank update of the invertible matrix $B_0 + M\underline{U}_k^{-1}\underline{Z}_k + M\overline{U}_k^{-1}\overline{Z}_k$ and uses the Sherman–Morrison–Woodbury formula to compute $\widetilde{B}_k^{-1}$ explicitly. This requires solving with $B_0 + M\underline{U}_k^{-1}\underline{Z}_k + M\overline{U}_k^{-1}\overline{Z}_k$ for multiple right-hand sides (required by the presence of the low rank term $F_k^t Q^{-1} F_k$) and matrix-matrix multiplications. As a result, this approach parallelizes efficiently as long as the solves with $B_0 + M\underline{U}_k^{-1}\underline{Z}_k + M\overline{U}_k^{-1}\overline{Z}_k$ parallelize well. Once the inverse of $\widetilde{B}_k^{-1}$ is computed, the Schur complement $Da(u_k)\widetilde{B}_k^{-1}Da(u_k)^t \in \mathbb{R}^{m\times m}$ of $\widetilde{B}_k$ in (23) can be computed using parallel matrix-matrix operations and used to solve $[Da(u_k)\widetilde{B}_k^{-1}Da(u_k)^t]\Delta\lambda = a(u_k) + \widetilde{B}_k^{-1}r_k$ for $\lambda$; this solve is done in serial and has negligible computational footprint because its size is small (being the number of contraints, not including the bounds). Finally, one only needs to solve $\widetilde{B}_k\Delta u = r_k - Da(u_k)^t\Delta\lambda$ to compute $\Delta u$ in (23), which requires another multiplication with the inverse of $\widetilde{B}_k$.

This approach extends readily to QIpmDDe since the limited-memory update (20) and the IPM linear system (21a)–(21c) are structurally identically to their Euclidean counterparts. In the Euclidean case ($B_0 = M = I$) this matrix is diagonal and the solves are trivial. Obviously, this approach readily extends to QIpmDDe as long as M and $B_0$ are *diagonal* matrices. For example, M is diagonal for piecewise constant elements and $B_0$ can be taken to be a positive definite diagonal matrix. The diagonality of M can also be ensured by using special quadrature formulas, for example see [27], or by employing mass lumping techniques [27,64]. For non-diagonal matrices $B_0$ and M, the aforementioned technique is efficient as long as solving linear systems involving $B_0 + M\underline{U}_k^{-1}\underline{Z}_k + M\overline{U}_k^{-1}\overline{Z}_k$ can be parallelized efficiently. We expect that this is possible since M is sparse even for higher order finite elements and $B_0$ can be chosen freely. Our current parallel implementation of QIpmDDe from HiOp only supports diagonal matrices M and $B_0$ and we defer the development of a more general implementation to future work.

## 5. Problem specification

It should be apparent from Section 3 that QIpmDDe needs minimal knowledge of the underlying discretized optimization space $U_h$, namely, it only requires the application (to vectors) of the mass matrix M, the inner product weight matrix H, and the inverse $H^{-1}$. This characteristic is of great practical value since it allows to use QIpmDDe for problems posed over Hilbert spaces with generic inner products. For a complete specification of (the discretization of) the infinite-dimensional problem (1a)–(1b), the following are needed:

(I1) the size $n$ of the representation u and the number $m$ of equality constraints;
(I2) routines for evaluating discretized functions f(u) and a(u);
(I3) routines for evaluating derivative representations $Df(u)$ and $Da(u)$;
(I4) representations $\underline{u}$ and $\overline{u}$ for the function bounds $\underline{u}$ and $\overline{u}$, respectively;
(I5) routines for applying M, H, and $H^{-1}$ to vectors from $\mathbb{R}^n$;
(I6) a routine for applying $B_0$ to vectors from $\mathbb{R}^n$, which is evaluated at each iteration;

(I7) a routine returning the distribution of $n$-dimensional vector and matrices across MPI ranks, as per the discussion of data parallelism in Section 4

We remark that I5, I6, and I7 above are optional. If I5 is not provided, QIpmDDe uses $M = H = I$, and thus acts as an Euclidean solver. If I6 is not provided, the solver uses $B_0 = M$. If I7 is not provided, the parallel computations are switched off. HiOp's C++ abstract class for problem specification encapsulates I1–I7 above. We refer the interested reader to HiOp user manual for more software details [40].

## 6. Strengths and weaknesses of QIpmDDe on practical problems

We discuss the numerical performance of QIpmDDe by solving two classes of PDE-constrained problems for which the optimization variable $u$ appears as coefficient in the PDE state equation. For all problems, the optimization derivatives are obtained via an adjoint-based approach and calculus of variations, for example, see [44].

### 6.1. Inverse problem governed by an elliptic PDE

As a first example, we consider the estimation of a coefficient field in a prototype elliptic PDE. Depending on the interpretation of the inputs and the type of measurements, this problem arises, for instance, in inversion for the permeability field in a subsurface flow problem, for the conductivity field in a heat transfer problem, or the stiffness parameter field in a membrane deformation problem. We formulate the inverse problem over $\Omega = [0,1] \times [0,1]$ as follows: given (possibly noisy) observations $\mathrm{d} \in \mathbb{R}^q$ of the state solution $y$ in $\Omega$, a volume force $f \in H^{-1}(\Omega)$, and lower and upper bounds $\underline{u}, \overline{u} \in L^\infty(\Omega)$, we infer the coefficient field $u$ that best reproduces the observations by solving

$$\min_{u \in L^2(\Omega)} \mathcal{J}(u) := \frac{1}{2} \left\langle Oy(u) - \mathrm{d}, Oy(u) - \mathrm{d} \right\rangle_{\mathbb{R}^q} + \frac{\gamma}{2} \langle \nabla u, \nabla u \rangle_{L^2}, \quad \text{s.t. } \underline{u} \le u \le \overline{u}, \tag{24}$$

where $y(u)$ is the solution of the state problem $-\nabla \cdot (u \nabla y) = f$ in $\Omega$ and $y = 0$ on $\partial\Omega$. Above, $O : L^2(\Omega) \to \mathbb{R}^q$ is a linear observation operator that extracts measurements from $y$. The second term in the objective of (24) is the regularization term with parameter $\gamma > 0$ added to render the inverse problem well-posed [16,57].

For the numerical experiments, we used $f = 25$, $\underline{u} = 1.0$, and $\overline{u} = 3.5$. The 'true' parameter field is a Gaussian cut flat at 3.5 in the centre as shown in Figure 2 (bottom left). To synthesize the observations, we add 1% noise to the state solution $y$ corresponding to the 'true' parameter field to lessen the inverse crime [29] (see Figure 2 top right). The observations d are then obtained by extracting the values of the noisy state solution at the points shown in red in Figure 2 (top left). The state and adjoint variables are discretized with quadratic finite element basis functions, while the parameter is discretized with linear elements. We used the COMSOL Multiphysics library for finite element discretizations and MATLAB's sparse backsolve to solve the discretized state and adjoint linear systems [44]. The optimization problem (24) is solved on uniform meshes of various sizes and on a nonuniform mesh, as shown in Figure 2 (top left). For the latter, the dimensions of the state/adjoint and parameter are 5227 and 1328, respectively. The solution of the inverse

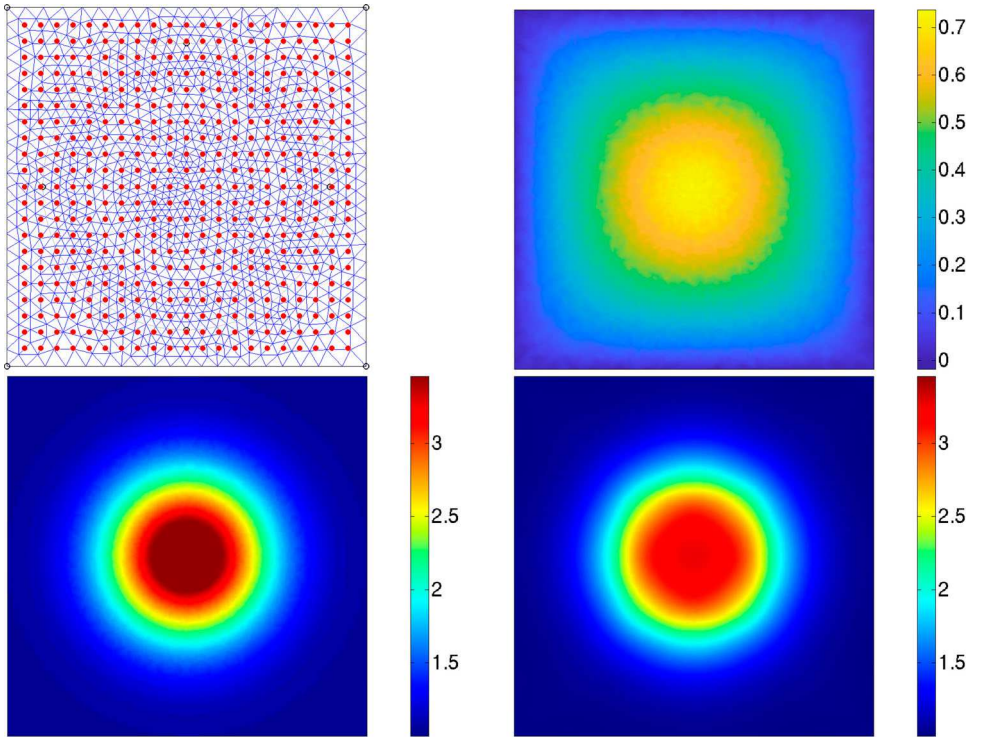


**Figure 2.** Results for the elliptic coefficient field inversion model (24). Top: The nonuniform mesh and the observation locations represented by the red dots (left) and the noisy observable $y$ (right). Bottom: The 'true' coefficient field $u$ (left), and the reconstructed coefficient field (right).

problem 24 with regularization parameter $\gamma = 1.5 \times 10^{-3}$ is shown in Figure 2 (bottom right).

We solve the above inverse problem with a MATLAB implementation of QIpmDDe method and with MATLAB's fmincon BFGS solver with full-memory secant approximation for the Hessian ($l = \infty$). The convergence tolerances were set to $10^{-6}$ for fmincon and to a more stringent $10^{-8}$ for QIpmDDe. QIpmDDe used $\mathrm{B}_0 = \mathrm{M} + \gamma \mathrm{K}$, where K is the stiffness matrix. This choice is motivated by the structure of the objective function: M corresponds to an uninformed second-derivative approximation (by the identity operator) of the first term in the objective, while $\gamma$K is exactly the representer of the second-derivative of the regularization term in the objective. On the other hand, fmincon most used a multiple of the identity matrix for $\mathrm{B}_0$.

Table 2 shows the number of iterations and objective values obtained using the proposed QIpmDDe method and MATLAB BFGS IPM implementation from fmincon. QIpmDDe was used with (full-memory) formula (19), which is denoted by $l = \infty$, and with the limited-memory BFGS secant formula (20) with $l = 6$. On the other hand, we ran fmincon with full-memory secant BFGS approximation. We remark from Table 2 that QIpmDDe shows mesh-independent iteration count with both $l = \infty$ and $l = 6$, whereas the Euclidean BFGS solver of fmincon requires an increasingly large number of iterations as the mesh is refined and/or distorted. Also we remark that the QIpmDDe requires less iterations than the Euclidean solver despite the fact it was ran with a more stringent termination

**Table 2.** QIpmDDe ($l = \infty$ and $l = 6$) and MATLAB BFGS IPM ($l = \infty$) performance for various meshes.

| | QIpmDDe | | | MATLAB BFGS | |
|---|---|---|---|---|---|
| | #iter | | | #iter | |
| **Mesh** | $l = 6$ | $l = \infty$ | obj. | $l = \infty$ | obj. |
| $10 \times 10$ | 44 | 30 | $1.483 \cdot 10^{-2}$ | 232 | $1.483 \cdot 10^{-2}$ |
| $20 \times 20$ | 44 | 34 | $1.455 \cdot 10^{-2}$ | 288 | $1.458 \cdot 10^{-2}$ |
| $30 \times 30$ | 53 | 34 | $1.449 \cdot 10^{-2}$ | 331 | $1.455 \cdot 10^{-2}$ |
| $40 \times 40$ | 46 | 36 | $1.448 \cdot 10^{-2}$ | 350 | $1.457 \cdot 10^{-2}$ |
| $50 \times 50$ | 48 | 35 | $1.447 \cdot 10^{-2}$ | 355 | $1.461 \cdot 10^{-2}$ |
| nonunif. | 45 | 36 | $1.448 \cdot 10^{-2}$ | 418 | $1.453 \cdot 10^{-2}$ |

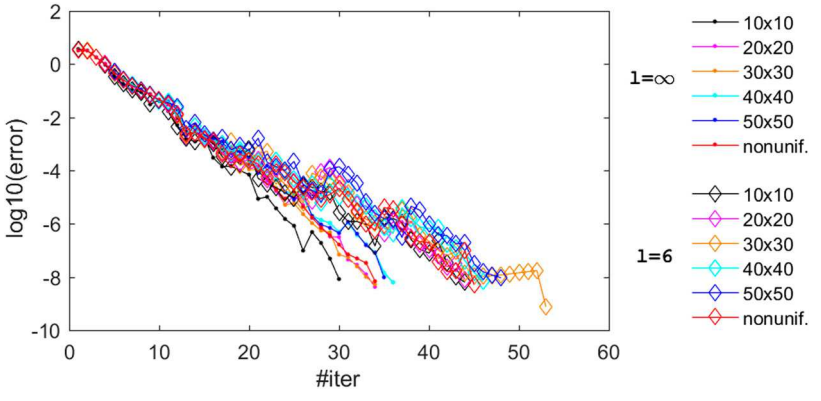


**Figure 3.** Convergence history of QIpmDDe with limited-memory BFGS ($l = 6$) and full-memory ($l = \infty$) when solving the inverse problem (24).

criterion. Our MATLAB implementation takes close to 800 seconds to solve the inverse problem on the finest uniform mesh ($50 \times 50$) and approximatively 320 seconds on the nonuniform mesh for $l = \infty$. The execution times for $l = 6$ are slighter smaller.

In Figure 3 we plot the decimal logarithm of the optimality error $e^{rr}$ at each iteration of QIpmDDe for the problems solved in Table 2. The convergence for both the full-memory and limited-memory QIpmDDe is fast and at a linear rate or better. In particular, QIpmDDe with full-memory ($l = \infty$) seems to exhibit a convergence rate better than linear nearby the optima. This desirable 'superlinear' convergence behaviour of QIpmDDe is in line with previous theoretical work [20,46] related to certain superlinearly convergent secant quasi-Newton methods for some classes of infinite-dimensional problems and, for this reason, can be regarded as a theoretical check of the validity of QIpmDDe (and its implementation).

### 6.2. Structural topology optimization

Topology optimization finds the optimal distribution of material in a given design domain $\Omega$ to minimize a cost function and satisfy constraint functions. We follow the ersatz approach, where the geometry is defined by a continuous field $\nu$ that represents the material

volume fraction. The resulting problem is commonly known as the compliance problem

$$\begin{aligned} \min_{\nu \in U_{ad}} J(\nu) &= \int_{\Gamma_N} y(\nu) \cdot f \, \mathrm{d}s, \\ \text{s.t. } a(\nu) &= \int_{\Omega} \nu \, \mathrm{d}x - V_{\max} \leq 0 \quad \text{and} \quad 0.01 \leq \nu \leq 1, \end{aligned} \tag{25}$$

where d$s$ is a one dimensional measure on the boundary $\Gamma_N$ where the load is applied and $y(\nu)$ is the solution of the linear elasticity problem

$$\begin{aligned} \nabla \cdot \sigma &= 0 \quad \text{in } \Omega, \quad \sigma = r(\tilde{\nu}) C[\nabla y] \quad \text{in } \Omega, \\ \sigma \cdot n &= f \quad \text{on } \Gamma_N, \quad y = 0 \quad \text{on } \Gamma_D. \end{aligned} \tag{26}$$

The filtered volume fraction field $\tilde{\nu}$ is required to obtain a well-posed problem and is obtained using a PDE filter [33], namely by solving

$$-\kappa \nabla^2 \tilde{\nu} + \tilde{\nu} = \nu, \tag{27}$$

where the constant $\kappa > 0$ is the minimum length scale of the design [33]. We use the SIMP penalization [4] to encourage 0–1 designs on the filtered volume fractions $\tilde{\nu}$. Finally, $C$ is the elasticity tensor of an isotropic material with Young's modulus $E = 1$ and Poisson's ratio $\mu = 0.3$ and $f$ is the applied traction on the surface $\Gamma$.

For the computational results of this section we interfaced QIpmDDe's C++ implementation from HiOp solver with Livermore Design Optimization (LiDO). LiDO is a design optimization library that uses the MFEM [34] finite element library and HYPRE [47] linear solvers to scalably discretize and solve the state elasticity and adjoint PDEs governing design physics. For this work, MFEM used piece constant finite elements for the design variables (densities) and Lagrange finite elements for the state. HYPRE used the conjugate gradient method equipped with the BoomerAMG algebraic multigrid preconditioner [17] for both the state and adjoint system. To solve the optimization problems, LiDO includes an internal implementation of the method of moving asymptotes (MMA) and also interfaces with two optimization libraries: HiOp and Ipopt. The combination of HiOp and MFEM provides LiDO competent HPC capabilities. For instance, the left quadmotor drone design from Figure 1 consisted of 880 million design parameters and required 8 hours of simulation on 9 216 cores of the Quartz cluster at LLNL. We will not elaborate here on the parallel capabilities due to space limitations, but we mention that the efficiency of the decomposition technique from [39] translates to QIpmDDe.

We first compare the Euclidean approach with our QIpmDDe algorithm and with MMA [1,6,50] on a standard cantilever beam design problem that is solved using a distorted mesh. This mesh is obtained by halving six times each element of the mesh shown in Figure 4, which resulted in a total of 249 856 elements (and design variables). The distortion factor, by which we mean the ratio of the areas of the largest and smallest discretization elements, is 256. We allowed for generous amount of material ($V_{\max} = 0.3$), which tends to make the problem less challenging.

The Euclidean solver[5] finds a design shown in Figure 5 that lacks the sharp transition from void to solid in the regions where the mesh is fine. Also, the design differs significantly from the 'expected' optimal design, shown in Figure 6 and 7. Numerically, the Euclidean

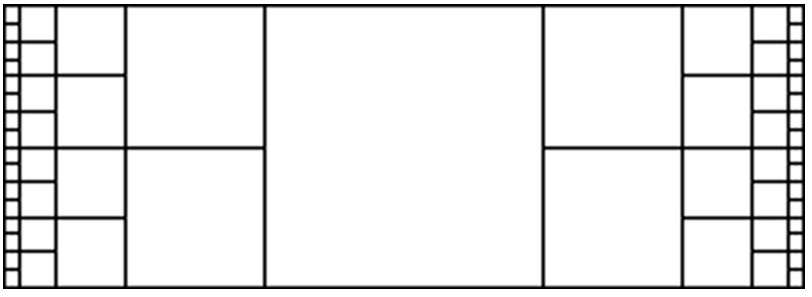

**Figure 4.** A distorted mesh for the cantilever beams used in this section.

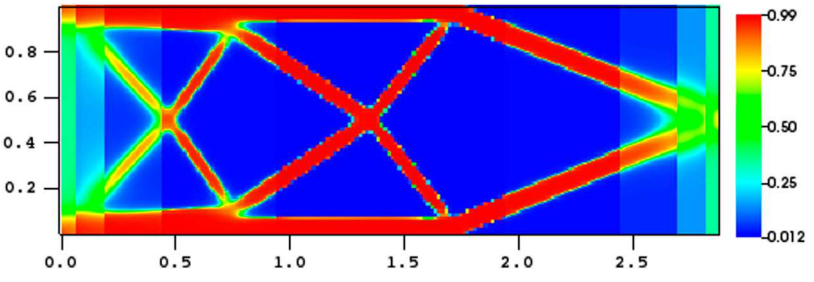


**Figure 5.** Design found by an Euclidean solver with the distorted mesh from Figure 4.

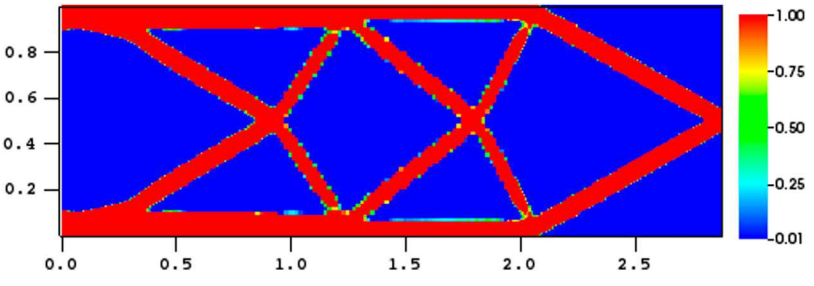


**Figure 6.** Design found by MMA with the distorted mesh from Figure 4 after 50 iterations.

solver converges to the required tolerance ($10^{-6}$) in 213 iterations and reports a minimum compliance of $1.001 \cdot 10^{-2}$. In contrast, both MMA and QIpmDDe find *valid* designs of up to 22.6% smaller compliance *in fewer iterations*. After only 50 iterations for example, MMA reports compliance of $7.787 \cdot 10^{-3}$ (design in Figure 6) and QIpmDDe reports a compliance $7.802 \cdot 10^{-3}$ (design in Figure 7).

Both QIpmDDe and MMA find in a relatively small number of iterations designs with sharp void-solid transitions suitable to manufacturing. They also compare similarly in terms of computational cost/PDEs evaluations. However, under tight stopping criteria, both QIpmDDe and MMA take a large number of iterations to converge and the vast majority of these iterations do not improve the compliance significantly. For instance, for the cantilever beam, at iteration 500 the compliances of MMA and QIpMDDe are $7.761 \cdot 10^{-3}$ and $7.742 \cdot 10^{-3}$, respectively, which is less than half percent improvement over the respective compliances at iteration 50. The execution time for 500 iterations is around 10 minutes

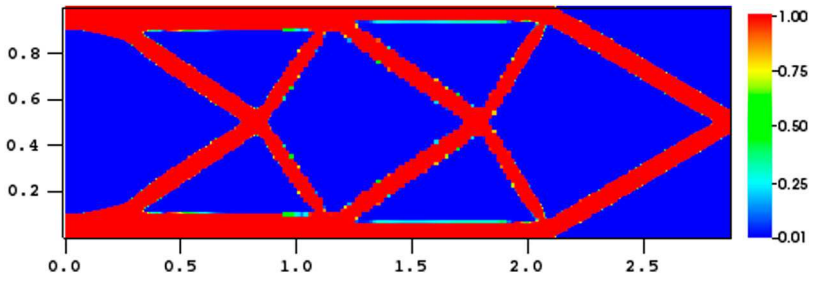


**Figure 7.** QIpmDDe design after 50 iterations.

using six 2.4 GHz cores of a Intel I5 processor. The execution time is overwhelmingly spent in solving the state and adjoint PDE, with the internal times of QIpMDDe, MMA, and Ipopt accounting for only a couple percent of total computational cost.

The slow convergence of QIpmDDe under tight convergence criteria is very likely caused by the choice of $B_0 = M$, which is not a good approximation for the second derivative of the compliance. A potential remedy would be to use an initial $B_0$ that approximates the second derivative of the compliance up to a compact perturbation; such choice can lead to a superlinearly convergent secant updates [20,46]. One potential way to achieve such approximations would be via multigrid technique, namely, to form a computationally affordable 'coarse' second derivative and use it as $B_0$. We defer such nontrivial development to future investigations.

Finally, we report that QIpmDDe consistently yields *mesh-independent* designs when used for topology optimization. For instance, the high-resolution quadmotor drone design at the left of Figure 1 is virtually identical to the designs obtained on coarser meshes. On the other hand, Euclidean solvers generally lack this design robustness feature when the mesh is refined and/or distorted and they provide radically different designs, inaccurate designs such as the one for the simple cantilever beam from Figure 5, or non-physical designs such as the drone design at the right of Figure 1.

## 7. Conclusions and future work

The QIpmDDe algorithm presented in this paper was developed for optimization problems with inequalities to be consistent with the inner product of the Hilbert space used for optimization parameters. In order to achieve this, we derived mathematically sound (and computationally efficient) discretization techniques for use within the filter linesearch mechanism and quasi-Newton secant updates used by QIpmDDe algorithm. The presented numerical evidence shows that the algorithm performs in a mesh-independent way and has good convergence properties even with limited-memory quasi-Newton Hessian approximations for a class of inverse problems.

We stress the limitations of our algorithm and solver. The most prevalent one is that parallel computing capabilities for optimization in Hilbert spaces can be currently achieved with our quasi-Newton strategy only by using a diagonal matrix approximation $B_0$ for the second derivative. As elaborated in Section 4 future work will be dedicated to removing this limitation. Also, we remark that fast (local) convergence may be elusive when

quasi-Newton approximations are used for the Hessian. This situation is especially pervasive when good initial Hessian approximations $B_0$ are not known, as it is the case for the structural topology optimization problems we considered in this work.

## Notes

1. This NLP solver is state-of-the-art and one of the most robust open-source optimization package, thus, we exclude the possibility of malfunction.
2. The subscripts $_u$ and $_y$ denote the partial derivatives with respect to $u$ and $y$, respectively.
3. i.e. $(u, \lambda, \underline{z}, \overline{z})$ would satisfy first-order necessary optimality conditions
4. The Kronecker delta function $\delta$ is such that $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$.
5. We have used the state-of-the-art solver Ipopt.

## Acknowledgments

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. The authors acknowledge valuable guidance from the MFEM team members V. Dobrev, M. Stowell, T. Kolev, and A. Barker on finite element methods, from E. Sachs on infinite-dimensional quasi-Newton algorithms, and from D. White on setting up the drone problem within LiDO.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

The work of the LLNL authors was supported by the LLNL LDRD program [project numbers 16-ERD-025 and 17-SI-005]. The work of C. G. Petra on derivation and formalization of the infinite-dimensional algorithm was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research Program [contract number DE-AC52-07NA27344]. The work of N. Petra was supported by NSF [grant number CAREER-1654311].

## Notes on contributors

***Cosmin G. Petra*** is a computational mathematician in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. Cosmin's work focuses on high-performance computing algorithms and C/C++ solvers for mathematical optimization. Prior to joining the Center for Applied Scientific Computing, Cosmin was with Argonne National Laboratory as a computational mathematician.

***Miguel Salazar De Troya*** is the Head of Simulation at Corintis SA. Previously, he was a postdoctoral researcher at the Lawrence Livermore National Laboratory. His field of expertise is topology optimization.

***Noemi Petra*** is an Associate Professor in the Department of Applied Mathematics at the University of California, Merced. She earned her Ph.D. degree in Applied Mathematics from the University of Maryland, Baltimore County. Prior to joining the University of California, Merced Noemi was the recipient of a Peter O'Donnell Jr. Postdoctoral Fellowship at the Oden Institute for Computational and Engineering Sciences at The University of Texas at Austin. Her research interests include large-scale Bayesian inverse problems governed by differential equations, uncertainty quantification in inference and prediction, and optimal experimental design.

***Youngsoo Choi*** is a computational math scientist in CASC under Computing directorate at LLNL. He has earned his undergraduate degree for Civil and Environmental Engineering from Cornell University and his PhD degree for Computational and Mathematical Engineering from Stanford University. He was a postdoc at Sandia National Laboratories and Stanford University prior to joining LLNL in 2017.

***Daniel Tortorelli***, retired from his 25 career of being the George Grim Professor of Mechanical Sciences and Engineering at the University of Illinois at Urbana-Champaign in 2016 to become the director of Lawrence Livermore National Laboratory's (LLNL) Center for Design Optimization. He has published extensively in the field of design optimization and has assembled a diverse team of subject matter experts and numerical modelers to solve many LLNL design problems.

## ORCID

*Cosmin G. Petra* http://orcid.org/0000-0002-1050-3221
*Miguel Salazar De Troya* http://orcid.org/0000-0002-7168-6172
*Noemi Petra* http://orcid.org/0000-0002-9491-0034
*Youngsoo Choi* http://orcid.org/0000-0001-8797-7970
*Geoffrey M. Oxberry* http://orcid.org/0000-0001-7451-8097
*Daniel Tortorelli* http://orcid.org/0000-0001-5346-5520

## References

[1] N. Aage and B.S. Lazarov, *Parallel framework for topology optimization using the method of moving asymptotes*, Struct. Multidiscip. Optim. 47 (2013), pp. 493–505.
[2] P.I. Barton, R.J. Allgor, W.F. Feehery, and S. Galán, *Dynamic optimization in a discontinuous world*, Ind. Eng. Chem. Res. 37 (1998), pp. 966–981.
[3] B. Benahmed, H. Mokhtar-Kharroubi, B. de Malafosse, and A. Yassine, *Quasi-Newton methods in infinite-dimensional spaces and application to matrix equations*, J. Glob. Optim. 49 (2011), pp. 365–379.
[4] M.P. Bendsøe and O. Sigmund, *Material interpolation schemes in topology optimization*, Arch. Appl. Mech. 69 (1999), pp. 635–654.
[5] M. Bergounioux, M. Haddou, M. Hintermüller, and K. Kunisch, *A comparison of a Moreau–Yosida-based active set strategy and interior point methods for constrained optimal control problems*, SIAM. J. Optim. 11 (2000), pp. 495–521.
[6] T. Borrvall and J. Petersson, *Large-scale topology optimization in 3D using parallel computing*, Comput. Methods Appl. Mech. Eng. 190 (2001), pp. 6201–6229.
[7] S.C. Brenner and R. Scott, *The Mathematical Theory of Finite Element Methods*, 3rd ed., Vol. 15, Springer, NY, USA, 2008.
[8] T. Bui-Thanh, O. Ghattas, J. Martin, and G. Stadler, *A computational framework for infinite-dimensional Bayesian inverse problems part I: The linearized case, with application to global seismic inversion*, SIAM. J. Sci. Comput. 35 (2013), pp. A2494–A2523.
[9] T. Bui-Thanh and Q.P. Nguyen, *FEM-based discretization-invariant MCMC methods for PDE-constrained Bayesian inverse problems*, Inverse Probl. Imaging 10 (2016), pp. 943.
[10] R.H. Byrd, J. Nocedal, and R.B. Schnabel, *Representations of quasi-Newton matrices and their use in limited memory methods*, Math. Program. 63 (1994), pp. 129–156.
[11] Y. Cao, S. Li, L. Petzold, and R. Serban, *Adjoint sensitivity analysis for DAEs: The adjoint system and its numerical solution*, SIAM J. Sci. Comput. 24 (2002), pp. 1076–1089.
[12] B.J. Debusschere, H.N. Najm, P.P. Pebay, O.M. Knio, R.G. Ghanem, and O.P. Le Maitre, *Numerical challenges in the use of polynomial chaos representations for stochastic processes*, SIAM. J. Sci. Comput. 26 (2004), pp. 698–719.
[13] A. Dener, A. Denchfield, T. Munson, J. Sarich, S. Wild, S. Benson, and L.C. McInnes, *TAO 3.10 Users Manual, Tech Report ANL/MCS-TM-322*. Argonne National Laboratory (2018).

[14] J. Dennis and J.J. Moré, *A characterization of superlinear convergence and its application to quasi-Newton methods*, Math. Comput. 28 (1974), pp. 549–560.
[15] J.E. Dennis Jr and J.J. Moré, *Quasi-Newton methods, motivation and theory*, SIAM Rev. 19 (1977), pp. 46–89.
[16] H.W. Engl and C.W. Groetsch, *Inverse and Ill-Posed Problems*, Academic Press, Boston, USA, 1987.
[17] R.D. Falgout and U.M. Yang, *HYPRE: A Library of High Performance Preconditioners*, in *Computational Science — ICCS 2002*, P.M.A. Sloot, A.G. Hoekstra, C.J.K. Tan, and J.J. Dongarra, eds., Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 632–641.
[18] S.W. Funke and P.E. Farrell, *A framework for automated PDE-constrained optimisation*, CoRR abs/1302.3894. Available at http://arxiv.org/abs/1302.3894, 2013.
[19] M.S. Gockenbach, *Understanding and Implementing the Finite Element Method*, SIAM, Philadelphia, PA, USA, 2006.
[20] A. Griewank, *The local convergence of Broyden-like methods on Lipschitzian problems in Hilbert spaces*, SIAM. J. Numer. Anal. 24 (1987), pp. 684–705.
[21] M.J. Grote, J. Huber, D. Kourounis, and O. Schenk, *Inexact interior-point method for pde-constrained nonlinear optimization*, SIAM. J. Sci. Comput. 36 (2014), pp. A1251–A1276.
[22] M.D. Gunzburger, *Perspectives in Flow Control and Optimization*, SIAM, Philadelphia, 2003.
[23] W.W. Hager, *Runge-Kutta methods in optimal control and the transformed adjoint system*, Numer. Math. 87 (2000), pp. 247–282.
[24] R. Herzog and E. Sachs, *Preconditioned conjugate gradient method for optimal control problems with control and state constraints*, SIAM. J. Matrix. Anal. Appl. 31 (2010), pp. 2291–2317.
[25] M. Hintermüller and M. Hinze, *Moreau–Yosida regularization in state constrained elliptic control problems: Error estimates and parameter adjustment*, SIAM. J. Numer. Anal. 47 (2009), pp. 1666–1683.
[26] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints*, Springer, Dordrecht, 2009.
[27] M. Hinze and A. Schiela, *Discretization of interior point methods for state constrained elliptic optimal control problems: Optimal error estimates and parameter adjustment*, Comput. Optim. Appl. 48 (2011), pp. 581–600.
[28] K. Ito and K. Kunisch, *Lagrange Multiplier Approach to Variational Problems and Applications*, SIAM, Philadelphia, PA, USA, 2008.
[29] J. Kaipio and E. Somersalo, *Statistical and Computational Inverse Problems*, Applied Mathematical Sciences, Vol. 160, Springer-Verlag, New York, 2005.
[30] R.C. Kirby, *From functional analysis to iterative methods*, SIAM Rev. 52 (2010), pp. 269–293.
[31] D. Kouri and D. Ridzal, *Rapid Optimization Library – Functional Interface*, Available at https://trilinos.org/docs/r12.12/packages/rol/doc/html/group__func__group.html, 2018.
[32] D. Kraft, *Algorithm 733: TOMP–Fortran modules for optimal control calculations*, ACM Trans. Math. Softw. 20 (1994), pp. 262–281.
[33] B.S. Lazarov and O. Sigmund, *Filters in topology optimization based on Helmholtz differential equations*, Int. J. Numer. Methods Eng. 86 (2011), pp. 765–781.
[34] Modular finite element methods. Available at www.mfem.org.
[35] J. Nocedal and S.J. Wright, *Numerical Optimization*, 2nd ed., Springer, New York, 2006.
[36] D.B. Ozyurt and P.I. Barton, *Cheap second order directional derivatives of stiff ODE embedded functionals*, SIAM. J. Sci. Comput. 26 (2005), pp. 1725–1743.
[37] J.W. Pearson and J. Gondzio, *Fast interior point solution of quadratic programming problems arising from pde-constrained optimization*, Numer. Math. 137 (2017), pp. 959–999.
[38] J.W. Pearson, M. Porcelli, and M. Stoll, *Interior-point methods and preconditioning for pde-constrained optimization problems involving sparsity terms*, Numer. Linear Algebra Appl. 27 (2020), pp. 272–298.
[39] C.G. Petra, *A memory-distributed quasi-Newton solver for nonlinear optimization with a small number of constraints*, J. Parallel Distrib. Comput. 133 (2019), pp. 337–348.
[40] C.G. Petra, *HiOp – User Guide*, Tech. Rep. LLNL-SM-743591, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, 2018.

[41] C.G. Petra, N. Chiang, and M. Anitescu, *A structured quasi-Newton algorithm for separable optimization with incomplete Hessian*, SIAM J. Optim. 28 (2019), pp. 1048–1075.
[42] N. Petra, J. Martin, G. Stadler, and O. Ghattas, *A computational framework for infinite-dimensional Bayesian inverse problems: Part II. stochastic Newton MCMC with application to ice sheet flow inverse problems*, SIAM. J. Sci. Comput. 36 (2014), pp. A1525–A1555.
[43] N. Petra and E.W. Sachs, *Second Order Adjoints in Optimization*, in *Numerical Analysis and Optimization*. Springer, Cham, 2020, pp. 209–230.
[44] N. Petra and G. Stadler, *Model variational inverse problems governed by partial differential equations*, Tech. Rep. 11–05, ICES, The University of Texas at Austin, 2011.
[45] M.J.D. Powell, *A Direct Search Optimization Method that Models the Objective and Constraint Functions by Linear Interpolation*, in *Advances in Optimization and Numerical Analysis*, S. Gomez and J.P. Hennart, eds., Springer, Netherlands, 1994, pp. 51–67.
[46] E.W. Sachs, *Broyden's method in Hilbert space*, Math. Program. 35 (1986), pp. 71–82.
[47] Scalable linear solvers and multigrid methods, Available at www.llnl.gov/casc/hypre.
[48] A. Schiela, *An interior point method in function space for the efficient solution of state constrained optimal control problems*, Math. Program. 138 (2013), pp. 83–114.
[49] T. Schwedes, D.A. Ham, S.W. Funke, and M.D. Piggott, *Mesh Dependence in PDE-Constrained Optimisation an Application in Tidal Turbine Array Layouts*, Springer, Cham, 2017.
[50] K. Svanberg, *A class of globally convergent optimization methods based on conservative convex separable approximations*, SIAM. J. Optim. 12 (2002), pp. 555–573.
[51] D.A. Tortorelli and P. Michaleris, *Design sensitivity analysis: Overview and review*, Inverse Probl. Eng. 1 (1994), pp. 71–105.
[52] F. Tröltzsch, *Optimal Control of Partial Differential Equations*, American Mathematical Society, Providence, RI, USA, 2010.
[53] M. Ulbrich and S. Ulbrich, *Superlinear convergence of affine-scaling interior-point newton methods for infinite-dimensional nonlinear problems with pointwise bounds*, SIAM J. Control Optim. 38 (2000), pp. 1938–1984.
[54] M. Ulbrich and S. Ulbrich, *Primal-dual interior-point methods for PDE-constrained optimization*, Math. Program. 117 (2009), pp. 435–485.
[55] M. Ulbrich, S. Ulbrich, and M. Heinkenschloss, *Global convergence of trust-region interior-point algorithms for infinite-dimensional nonconvex minimization subject to pointwise bounds*, SIAM J. Control Optim. 37 (1999), pp. 731–764.
[56] U. Villa, N. Petra, and O. Ghattas, *hIPPYlib: An extensible software framework for deterministic and Bayesian inverse problems*, J. Open Source Softw. 3 (2018), pp. 1–2.
[57] C.R. Vogel, *Computational Methods for Inverse Problems*, Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 2002.
[58] R.G. Vuchkov, C.G. Petra, and N. Petra, *On the derivation of quasi-Newton formulas for optimization in function spaces*, Numer. Funct. Anal. Optim. 41 (2020), pp. 1564–1587.
[59] A. Wächter and L.T. Biegler, *Line search filter methods for nonlinear programming: Local convergence*, SIAM. J. Optim. 16 (2005), pp. 32–48.
[60] A. Wächter and L.T. Biegler, *Line search filter methods for nonlinear programming: Motivation and global convergence*, SIAM. J. Optim. 16 (2005), pp. 1–31.
[61] A. Wächter and L.T. Biegler, *On the implementation of an interior-point filter line-search algorithm for nonlinear programming*, Math. Program. 106 (2006), pp. 25–57.
[62] M. Weiser, *Interior point methods in function space*, SIAM J. Control Optim. 44 (2005), pp. 1766–1786.
[63] M. Weiser and A. Schiela, *Function space interior point methods for PDE constrained optimization*, in *PAMM: Proceedings in Applied Mathematics and Mechanics*, Vol. 4. Wiley Online Library, Weinheim, 2004, pp. 43–46.
[64] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu *The Finite Element Method: Its Basis and Fundamentals*, Elsevier, Oxford, 2005.