An ADMM-Newton-CNN Numerical Approach to a TV Model for Identifying Discontinuous Diffusion Coefficients in Elliptic Equations: Convex Case with Gradient Observations

Wenyi Tian[*] Xiaoming Yuan[†] Hangrui Yue[‡]

July 22, 2021

Identifying the discontinuous diffusion coefficient in an elliptic equation with observation data of the gradient of the solution is an important nonlinear and ill-posed inverse problem. Models with total variational (TV) regularization have been widely studied for this problem, while the theoretically required nonsmoothness property of the TV regularization and the hidden convexity of the models are usually sacrificed when numerical schemes are considered in the literature. In this paper, we show that the favorable nonsmoothness and convexity properties can be entirely kept if the well-known alternating direction method of multipliers (ADMM) is applied to the TV-regularized models, hence it is meaningful to consider designing numerical schemes based on the ADMM. Moreover, we show that one of the ADMM subproblems can be well solved by the active-set Newton method along with the Schur complement reduction method, and the other one can be efficiently solved by the deep convolutional neural network (CNN). The resulting ADMM-Newton-CNN approach is demonstrated to be easily implementable and very efficient even for higher-dimensional spaces with fine mesh discretization.

Keywords: Diffusion coefficient identification; elliptic equation; total variation; alternating direction method of multipliers; active-set Newton method; Schur complement reduction; convolutional neural network.

1 Introduction

Consider the canonical elliptic equation

$$\begin{cases} -\nabla\cdot(q(x)\nabla u(x)) = f(x), & x\in\Omega,\\ u(x)=0, & x\in\Gamma, \end{cases} \tag{1.1}$$

where Ω is a bounded polyhedral domain in $\mathbb{R}^d$ ($d=1,2,3$) with a piecewise smooth boundary $\Gamma := \partial\Omega$; $u : \Omega\cup\Gamma \to \mathbb{R}$ with $u\in H_0^1(\Omega)$; $q:\Omega\to\mathbb{R}$ with $q\in L^\infty(\Omega)$; and $f:\Omega\to\mathbb{R}$ with $f\in H^{-1}(\Omega)$ is given. The elliptic equation (1.1) describes various physical phenomena such as the flow of a fluid through some medium with the permeability $q(x)$ and the concentration $u(x)$, and the heat transfer in a material with the conductivity $q(x)$ and the temperature $u(x)$. For the diffusion coefficient $q(x)$, it is often impractical to measure it directly (e.g., when it is the conductivity of a medium), but it is easier to observe the solution u of (1.1) or its gradient [11, 17]. Hence, it is interesting to consider the inverse problem of identifying the diffusion coefficient $q(x)$ with observation data of the solution u of the elliptic equation (1.1) or its gradient. This inverse problem finds applications in various industrial areas such as reservoir simulations, underground water investigations, geophysics

[*]Center for Applied Mathematics, Tianjin University, Tianjin 300072, China. Email: twymath@gmail.com

[†]Department of Mathematics, The University of Hong Kong, Hong Kong, China. Email: xmyuan@hku.hk

[‡]Corresponding author. Department of Mathematics, The University of Hong Kong, Hong Kong, China. Email: yuehangrui@gmail.com

and electrical impedance tomography. We refer to the monographs [3, 14] for more introductions. Note that the elliptic equation (1.1) is linear if $q(x)$ is known, but the inverse problem of identifying $q(x)$ is nonlinear. Also, as mentioned in [14, 37], $q(x)$ cannot be uniquely determined by $u(x)$ since $q(x)$ can be arbitrary when u is constant on some open subset of Ω. Thus, identifying the diffusion coefficient $q(x)$ of (1.1) is an ill-posed inverse problem.

1.1 TV model

Let us consider the case where observation data of the gradient of the solution u of (1.1) are available subject to some noise with the noisy level $\delta > 0$; it is denoted by $\nabla u_\delta \in (L^2(\Omega))^d$. Note that the gradient type data ∇u_δ can be constructed via a mollification procedure by the Clément interpolation [12] if only the observation data of u is available, see, e.g. [20, 31, 43]. In some literatures such as [11, 19, 51], it has been proposed to recover the discontinuous coefficient $q(x)$ in (1.1) with $\nabla u_\delta \in (L^2(\Omega))^d$ via the model

$$\begin{aligned} &\min_{q,u} \left\{ \frac{1}{2}\int_\Omega q|\nabla u - \nabla u_\delta|^2 \mathrm{d}x + \alpha \int_\Omega |\nabla q| \right\}, \\ &\text{s.t.} \quad -\nabla\cdot(q\nabla u) = f, \quad (q,u) \in K \times H_0^1(\Omega), \end{aligned} \tag{1.2}$$

in which $\frac{1}{2}\int_\Omega q|\nabla u - \nabla u_\delta|^2 \mathrm{d}x$ is a data-fidelity term and $\int_\Omega |\nabla q|$ is the total variation (TV) regularization term defined in [2]. That is, we have

$$\int_\Omega |\nabla q| := \sup \left\{ \int_\Omega q \,\mathrm{div}\varphi \,\mathrm{d}x : \varphi \in C_c^1(\Omega;\mathbb{R}^d), \|\varphi\|_\infty \le 1 \right\} \tag{1.3}$$

with $\|\varphi\|_\infty = \sup_{x\in\Omega}(\sum_{i=1}^d |\varphi_i(x)|^2)^{1/2}$, "div" denotes the divergence operator, and $C_c^1(\Omega;\mathbb{R}^d)$ is the set of once continuously differentiable $\mathbb{R}^d$-valued functions with compact support in Ω, see, e.g., [2, 50] for more details. Moreover, the admissible set K is

$$K := \{q \in L^\infty \cap BV(\Omega) : 0 < a_0 \le q(x) \le a_1, \text{ a.e. in } \Omega\}, \tag{1.4}$$

and the $BV(\Omega)$ space endowed with the norm $\|q\|_{BV} := \|q\|_{L^1(\Omega)} + \int_\Omega |\nabla q|$ is a Banach space; see, e.g., [2, 50] for more details. Also, in (1.2), $\alpha > 0$ is a parameter determining the relative weights of the data-fidelity and TV regularization terms in the objective functional.

Note that the elliptic equation (1.1) is the Euler-Lagrange equation of the energy functional

$$\frac{1}{2}\int_\Omega (q|\nabla u|^2 - 2fu)\mathrm{d}x,$$

and the following identity holds (see, e.g., [34]):

$$\frac{1}{2}\int_\Omega q|\nabla u - \nabla u_\delta|^2 \mathrm{d}x = \frac{1}{2}\int_\Omega (q|\nabla u_\delta|^2 - 2fu_\delta)\mathrm{d}x - \frac{1}{2}\int_\Omega (q|\nabla u|^2 - 2fu)\mathrm{d}x,$$

where u_δ denotes the approximation to the solution u subject to the noise level $\delta > 0$. Hence, the data-fidelity term in (1.2) measures the difference of the energy functional of the elliptic equation (1.1) at u_δ and u, and it has been widely used in the literature, see, e.g., [11, 18, 19, 25, 34, 35, 36, 37]. For the TV regularization term, it is capable of reserving the piecewise-constant property and it has found various applications such as image denoising or reconstruction. The TV regularization has also been considered for identifying the diffusion coefficient $q(x)$ of (1.1) because it is generally discontinuous and also owns the piecewise-constant property for many applications such as reservoir simulations and electrical impedance tomography. We refer to, e.g., [8, 9, 11, 19], for more discussions. One interesting fact is that, as proved in [18], although the data-fidelity functional in (1.2) is nonconvex with respect to q and u jointly, it is convex with respect to q if u is represented as a function of q.

1.2 ALM for the smoothing TV_ϵ model

The TV term $\int_\Omega |\nabla q|$ in (1.2) is not differentiable and it could be difficult to tackle the nonsmoothness property for algorithmic design. In earlier literatures, it is popular to consider smoothing the TV term and then use the smoothing surrogate to replace the original TV term. For instance, in [1, 8, 9, 11, 32, 46], it is suggested to replace the TV term $\int_\Omega |\nabla q|$ with the surrogate

$$TV_\epsilon(q) := \int_\Omega \sqrt{|\nabla q|^2 + \epsilon}\, \mathrm{d}x,$$

where $\epsilon > 0$ is a smoothing parameter such that $TV_\epsilon(q) \to \int_\Omega |\nabla q|$ as $\epsilon \to 0$. In other words, instead of considering the TV model (1.2), the following approximated model with a smoothing regularization term is popularly considered:

$$\begin{aligned} &\min_{q,u} \ \Big\{\frac{1}{2}\int_\Omega q|\nabla u - \nabla u_\delta|^2 \mathrm{d}x + \alpha TV_\epsilon(q)\Big\}, \\ &\text{s.t.} \quad -\nabla\cdot(q\nabla u) = f, \ \ (q,u) \in K \times H_0^1(\Omega). \end{aligned} \tag{1.5}$$

To solve (1.5) numerically, we choose the following piecewise linear finite element space V_h to discretize the functions q and u:

$$V_h = \big\{v_h \in C(\Omega) : v_h|_\tau \in \mathcal{P}_1 \text{ for each } \tau \in \mathcal{T}_h\big\}, \tag{1.6}$$

where $\mathcal{P}_1$ is the space consisting of polynomials of degrees less than or equal to one, $\mathcal{T}_h$ denotes a regular partition of Ω into d-simplexes, and $h = \max\limits_{\tau\in\mathcal{T}_h} \operatorname{diam}(\tau)$ is the maximal diameter. Then, we obtain a discretized version of (1.5) with finite element approximation as

$$\begin{aligned} &\min_{q_h,u_h} \ \Big\{\frac{1}{2}\int_\Omega q_h|\nabla u_h - \nabla u_\delta|^2 \mathrm{d}x + \alpha TV_\epsilon(q_h)\Big\}, \\ &\text{s.t.} \ \ \big(q_h\nabla u_h, \nabla\phi_h\big) = (f,\phi_h), \ \forall\, \phi_h \in \bar{V}_h; \ \ (q_h,u_h) \in K_h \times \bar{V}_h, \end{aligned} \tag{1.7}$$

where $K_h := V_h \cap K$, $\bar{V}_h := V_h \cap H_0^1(\Omega)$ and $(\cdot,\cdot)$ is the regular inner-product in $L^2(\Omega)$. For solving the model (1.7), a particularly useful approach is the augmented Lagrangian technique developed in [27] and then widely used in other literatures such as [8, 9, 11, 26, 28, 32, 46]. More precisely, the augmented Lagrangian functional of (1.7) is

$$L_{\gamma_k}(q_h,u_h;\mu_h) : = \frac{1}{2}\int_\Omega q_h|\nabla u_h - \nabla u_\delta|^2 \mathrm{d}x + \alpha TV_\epsilon(q_h) + (\nabla e(q_h,u_h), \nabla \mu_h) + \frac{\gamma_k}{2}\|\nabla e(q_h,u_h)\|^2_{L^2(\Omega)}, \tag{1.8}$$

with $\gamma_k > 0$ the penalty parameter and $\mu_h \in V_h$ the Lagrange multiplier. The constraint in (1.7) is augmented by

$$(\nabla e(q_h,u_h), \nabla\phi_h) := (q_h\nabla u_h, \nabla\phi_h) - (f,\phi_h), \ \forall\,(q_h,u_h) \in K_h \times \bar{V}_h, \ \forall\, \phi_h \in \bar{V}_h, \tag{1.9}$$

where $e(q,u) = (-\Delta)^{-1}\big(\nabla\cdot(q\nabla u) + f\big)$ and $e(\cdot,\cdot)$ can be viewed as an operator from $K \times H_0^1(\Omega)$ to $H_0^1(\Omega)$. In [11], it is proved that the discretized augmented Lagrangian functional (1.8) exists at least one saddle-point, and the finite element solution (u_h, p_h) converges to the solution of (1.2). In [11, 27], it is suggested to apply the classic augmented Lagrangian method (ALM) originally proposed in [23, 42] to (1.7), and the iterative scheme reads as

$$\begin{cases} (q_h^{k+1}, u_h^{k+1}) = \mathop{\arg\min}\limits_{(q_h,u_h)\in K_h\times\bar{V}_h} L_{\gamma_k}(q_h,u_h;\mu_h^k), \\ \mu_h^{k+1} = \mu_h^k + \gamma_k e(q_h^{k+1}, u_h^{k+1}). \end{cases} \tag{1.10}$$

Note that the (q_h,u_h)-subproblem in (1.10) is a smooth optimization problem with a box constraint K_h on the variable q_h. In [11], global convergence of (1.10) is proved under the condition that the exact solution of the (q_h,u_h)-subproblem of (1.10) can be obtained at each iteration.

1.3 Motivations and goals

Smoothing the TV term loses the originally favorable nonsmoothness property, but enables the eligibility of applying the well known ALM (1.10). Meanwhile, the ALM (1.10) is mainly of conceptual sense because it is very challenging to implement it numerically. As remarked in [33], the smoothing surrogate $TV_\epsilon(q)$ leads to a nearly singular and indefinite nonlinear minimization system and solving this system is "a big difficulty to the numerical resolution process". Indeed, augmenting the constraint $e(q_h, u_h) = 0$ makes the augmented Lagrangian functional $L_\gamma(q_h, u_h; \mu_h^k)$ nonconvex, and hence the hidden convexity with respect to q in (1.2) is also lost in the ALM (1.10). The nonconvex (q_h, u_h)-subproblem in the ALM (1.10) is numerically difficult also because of the high dimensionality of its variables, the coupling of different variables, as well as its nonlinear structure and ill-conditionedness. In literatures such as [11, 33], it is suggested to solve the (q_h, u_h)-subproblem inexactly by splitting the variables q_h and u_h, and then solving them alternatively. As analyzed in [33], the resulting u_h-subproblem is a linear yet ill-conditioned saddle-point system and the q_h-subproblem is a nearly singular nonlinear minimization problem — both are still very difficult. It is suggested in [11] to apply some first-order algorithm with an Armijo line search to solve the decomposed q_h-subproblem, each iteration of which also requires solving an ill-conditioned linear system. All these strategies are targeted for approximating the solution of the (q_h, u_h)-subproblem in (1.10) heuristically, without any guarantee to the theoretically rigorous convergence. All these difficulties become much severer if a higher-dimensional space with $d \geq 2$ is considered and fine mesh discretization is used. Indeed, the dimension and condition numbers of the involved linear systems are both of order $O(h^{-d})$. Hence, it is easy to understand the lack of numerical study in the literatures for higher-dimensional spaces of $d \geq 2$ in (1.2) with fine mesh discretization. To the best of our knowledge, only some limited numerical studies for the case where $d = 1$ in (1.5) and coarse mesh discretization (e.g., $h = 1/80$) are available in [11, 51]. To summarize, it is extremely challenging to find the exact solution, or even an approximate solution with good accuracy, of the nonconvex, nonlinear, ill-conditioned and large-scaled (q_h, u_h)-subproblem in (1.10). This challenge posts substantial difficulties to validate the condition in [11] to guarantee the convergence of the ALM (1.10).

Because of the mentioned difficulties in the smoothing surrogate $TV_\epsilon(q)$ and the ALM (1.10), we are motivated to turn to consider solving the original model (1.2) directly. Our goals are: (1) to tackle the original TV model (1.2) so that the nonsmoothness properties of the diffusion coefficient of (1.1) can be inherited throughout; (2) to keep the convexity of the data-fidelity functional in (1.2) with respect to q throughout; (3) to design an implementable algorithm without any difficult subproblem such as the (q_h, u_h)-subproblem in (1.10) while it is efficient even for higher-dimensional space of (1.2) with $d = 2$ and fine mesh discretization. We will show that the first two goals can be fully achieved by applying the well-known alternating direction method of multipliers (ADMM) which was proposed originally in [16]. For the third goal, we should meticulously investigate the resulting subproblems, and propose some structure-exploiting strategies to tackle these subproblems more effectively. It is mentionable that the curse of dimensionality really matters from the numerical point of view. For example, for the case where the uniform mesh size $h = 1/1024$, the order of dimensionality of the resulting linear systems increases from 10^3 to 10^6 if the domain is changed from the unit interval $\Omega \subset \mathbb{R}$ to the unit square $\Omega \subset \mathbb{R}^2$, while if the domain is fixed as the unit square $\Omega \subset \mathbb{R}^2$, then the order of dimensionality of the resulting linear systems increases from 10^4 to 10^6 if the mesh size is refined from $1/128$ to $1/1024$.

1.4 Conceptual application of ADMM to the original TV model

As mentioned in [19], the elliptic equation (1.1) has a unique weak solution u in $H_0^1(\Omega)$ for each $q \in K$ and u is nonlinearly dependent on q. Then, the nonlinear coefficient-to-solution mapping $U : K \to H_0^1(\Omega)$, which maps each $q \in K$ to the unique solution $u = U(q) \in H_0^1(\Omega)$ of (1.1), is well defined. Instead of augmenting the elliptic equation (1.1) as a constraint by introducing $e(q, u)$ in (1.9), we temporarily take the liberty to represent u as a function of q via the equation (1.1), denote by

$$J(q) =: \frac{1}{2}\int_\Omega q|\nabla U(q) - \nabla u_\delta|^2 \mathrm{d}x,$$

and then reformulate the model (1.2) as a minimization problem only depending on q. Then the finite element discretized version of the nonsmooth problem (1.2) can be written as

$$\min_{q_h \in K_h} \Big\{ J(q_h) + \alpha \|\nabla q_h\|_{L^1(\Omega)} \Big\}. \tag{1.11}$$

To implement the ADMM to solve (1.11), there are multiple ways. For instance, it is easy to consider introducing an auxiliary variable $p_h := \nabla q_h$ so as to replace ∇q_h with p_h in the objective functional of (1.11). That is, the model (1.11) can be reformulated as

$$\begin{aligned} &\min_{q_h, p_h} \Big\{ J(q_h) + \alpha \|p_h\|_{L^1(\Omega)} \Big\}, \\ &\text{s.t.} \quad \nabla q_h - p_h = 0, \quad (q_h, p_h) \in K_h \times W_h, \end{aligned}$$

where $W_h := \{p_h \in L^1(\Omega; \mathbb{R}^d) : p_h|_\tau \text{ is constant for each } \tau \in \mathcal{T}_h\}$. Inspired by [5], we can employ the weighted L^2-inner product $(\cdot,\cdot)_h = h^d(\cdot,\cdot)$ and the corresponding norm $\|\cdot\|_h = h^{d/2}\|\cdot\|_{L^2(\Omega)}$ with d the space dimension, to penalize the constraint. Then, the augmented Lagrangian functional is

$$\hat{L}_\beta(q_h, p_h; \lambda_h) := \; J(q_h) + \alpha\|p_h\|_{L^1(\Omega)} + (\nabla q_h - p_h, \lambda_h)_h + \frac{\beta}{2}\|\nabla q_h - p_h\|_h^2, \tag{1.12}$$

where $\beta > 0$ is a penalty parameter. The corresponding ADMM iterative scheme reads as

$$\begin{cases} q_h^{k+1} = \underset{q_h \in K_h}{\arg\min}\, \hat{L}_\beta(q_h, p_h^k; \lambda_h^k), \\ p_h^{k+1} = \underset{p_h \in W_h}{\arg\min}\, \hat{L}_\beta(q_h^{k+1}, p_h; \lambda_h^k), \\ \lambda_h^{k+1} = \lambda_h^k + \beta(\nabla q_h^{k+1} - p_h^{k+1}). \end{cases} \tag{1.13}$$

In (1.12), we do not use the regular L^2-inner product $(\cdot,\cdot)$ and its induced L^2-norm penalty term $\frac{\beta}{2}\|\nabla q_h - p_h\|^2_{L^2(\Omega)}$. Indeed, as analyzed in [5], the L^2-inner product may lead to numerical instability because $\|\nabla q_h\|_{L^2(\Omega)}$ is unbounded. It is further noticed in [5] that an inverse estimate shows that ∇q_h is bounded with respect to $\|\cdot\|_h$ and the corresponding scheme tends to be more numerically stable.

Note that the gradient operator ∇ is involved in the penalty term in (1.12), and as analyzed in [40], the condition number of the corresponding stiffness matrix (whose entries are $(\nabla\phi_h^j, \nabla\phi_h^i)$ with ϕ_h^i the finite element basis functions of V_h) is of order h^{-d}. Hence, the condition number of the stiffness matrix may be extremely high for fine mesh. Accordingly, numerical performance of (1.13) may be more severely affected by the penalty parameter β if fine mesh is used for discretization, which can be easily verified by numerical experiments. Because of this concern, we prefer to penalize some term irrelevant to the gradient operator ∇. Note that the condition number of the mass matrix is bounded and independent of the mesh size h; see, e.g., [40]. Therefore, we introduce the auxiliary variable p_h and replace q_h in the TV term. That is, we reformulate the TV model (1.11) as

$$\begin{aligned} &\min_{q_h, p_h} \Big\{ J(q_h) + \alpha \|\nabla p_h\|_{L^1(\Omega)} \Big\}, \\ &\text{s.t.} \quad q_h - p_h = 0, \quad (q_h, p_h) \in K_h \times V_h. \end{aligned} \tag{1.14}$$

The corresponding augmented Lagrangian functional of (1.14) is

$$\mathcal{L}_\beta(q_h, p_h; \lambda_h) := \; J(q_h) + \alpha\|\nabla p_h\|_{L^1(\Omega)} + (q_h - p_h, \lambda_h) + \frac{\beta}{2}\|q_h - p_h\|^2_{L^2(\Omega)}, \tag{1.15}$$

and the corresponding ADMM scheme reads as:

$$\begin{cases} q_h^{k+1} = \arg\min\limits_{q_h \in K_h} \mathcal{L}_\beta(q_h, p_h^k; \lambda_h^k), & \text{(1.16a)} \\ p_h^{k+1} = \arg\min\limits_{p_h \in V_h} \mathcal{L}_\beta(q_h^{k+1}, p_h; \lambda_h^k), & \text{(1.16b)} \\ \lambda_h^{k+1} = \lambda_h^k + \beta(q_h^{k+1} - p_h^{k+1}). & \text{(1.16c)} \end{cases}$$

It is arguably trivial to derive the ADMM (1.16) conceptually. But it is clear that both the subproblems in (1.16) are convex and the scheme is for solving the discretized version of the original TV model (1.2). Hence, the ADMM (1.16) essentially differs from the ALM (1.10) in the sense that the original TV term as well as the convexity with respect to the variable q_h are both kept. On the other hand, as we shall show in Section 3, despite its convexity, it is highly nontrivial to solve the resulting subproblems, especially the q_h-subproblem (1.16a). Hence, the ADMM (1.16) is numerically meaningful only if both the q_h- and p_h-subproblems can be solved efficiently, especially for the case where $d \geq 2$ and h is small.

1.5 Organization

The rest of this paper is organized as follows. In Section 2, some preliminaries are summarized for further analysis. Then, we focus on the subproblems (1.16a) and (1.16b) in Sections 3 and 4, respectively. The flowchart of implementation of the proposed numerical approach is presented in Section 5. Some preliminary numerical results are reported in Section 6 to verify the efficiency of the proposed numerical approach. Finally, some conclusions are drawn in Section 7.

2 Preliminaries

In this section, we summarize some preliminaries which will be used for further analysis. We say that $u \in H_0^1(\Omega)$ is a weak solution of the elliptic equation (1.1) if it satisfies the following variational form:

$$a(u, \phi) := \int_\Omega q \nabla u \cdot \nabla \phi \mathrm{d}x = \int_\Omega f \phi \mathrm{d}x, \ \ \forall\, \phi \in H_0^1(\Omega).$$

The bilinear form $a(\cdot,\cdot)$ satisfies the coercivity condition $a(u,u) \geq C_\Omega \|u\|^2_{H_0^1(\Omega)}$ for any $u \in H_0^1(\Omega)$ and $q \in K$, where C_Ω is a positive constant depending on Ω and the low bound a_0 of q. By the Lax-Milgram theorem [15], there exists a unique weak solution u of (1.1) in $H_0^1(\Omega)$ for each $q \in K$, and u is nonlinearly dependent on q. Then, we can define the nonlinear coefficient-to-solution mapping $U : K \to H_0^1(\Omega)$, which maps each $q \in K$ to the unique solution $u = U(q) \in H_0^1(\Omega)$ of (1.1); see [19] for more details.

For discretization, because of the low regularity of the functions in the space $BV(\Omega)$, only some low order polynomials will be chosen for the finite element space. As mentioned in [4], the piecewise affine globally continuous finite element spaces are dense in $BV(\Omega)$ with respect to weak* convergence in $BV(\Omega)$, while in general the piecewise constant finite element approximation for u cannot be expected to converge to an exact solution. Thus, we discretize the model (1.2) in the finite element space V_h (see (1.6)) and obtain the following discrete problem:

$$\min_{q_h \in K_h} \left\{ J(q_h) + \alpha \|\nabla q_h\|_{L^1(\Omega)} \right\}, \tag{2.1}$$

where $K_h = V_h \cap K$, and V_h and K are given by (1.6) and (1.4), respectively. The energy functional $J(q_h)$ of (2.1) is

$$J(q_h) := \frac{1}{2} \int_\Omega q_h |\nabla U(q_h) - \nabla u_\delta|^2 \mathrm{d}x, \tag{2.2}$$

where $U(q_h)$ is the solution of the following variational form:

$$a_h(q_h, U(q_h); \phi_h) = (f, \phi_h),\ \forall\, \phi_h \in \bar{V}_h,$$

with $\bar{V}_h := V_h \cap H_0^1(\Omega)$ and

$$a_h(q_h, U(q_h); \phi_h) := \int_\Omega q_h \nabla U(q_h) \cdot \nabla \phi_h \mathrm{d}x. \tag{2.3}$$

According to [18, Lemma 2.3], the functional $J(\cdot)$ in (2.2) is convex on the convex set K. For any $q \in K, \xi, \eta \in L^\infty(\Omega)$, the first derivative of $J(\cdot)$ is given by

$$J'(q)\xi = -\frac{1}{2}\int_\Omega \xi |\nabla U(q)|^2 \mathrm{d}x + \frac{1}{2}\int_\Omega \xi |\nabla u_\delta|^2 \mathrm{d}x, \tag{2.4}$$

and the second derivative of $J(\cdot)$ is given by

$$J''(q)(\xi, \eta) = -\int_\Omega \xi \nabla U(q) \cdot \nabla U'(q)\eta \mathrm{d}x, \tag{2.5}$$

where $U'(q)\eta$ satisfies

$$\int_\Omega q \nabla U'(q)\eta \cdot \nabla v \mathrm{d}x = -\int_\Omega \eta \nabla U(q) \cdot \nabla v \mathrm{d}x,\ \forall\, v \in H_0^1(\Omega). \tag{2.6}$$

In the next two sections, we will elaborate on how to solve the resulting subproblems for the ADMM (1.16). For notational convenience, we denote by $\boldsymbol{q} := (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N)^\top \in \mathbb{R}^N$ the coefficients of q_h. That is, $q_h = \sum_{i=1}^N \boldsymbol{q}_i \phi_h^i$, where $\{\phi_h^i\}_{i=1}^N$ are the finite element basis functions in V_h. The same setting is also applied to $p_h,\ \lambda_h,\ u_h\ , r_h$, with the coefficients $\boldsymbol{p},\ \boldsymbol{\lambda},\ \boldsymbol{u},\ \boldsymbol{r} \in \mathbb{R}^N$, respectively. Then, we define the function $J(\boldsymbol{q}) := J(q_h)$.

3 Active-set Newton method for the q_h-subproblem

In this section, we focus on the q_h-subproblem (1.16a). How to solve this subproblem is crucial to ensure the performance of the ADMM (1.16), and this is the most technical part of the paper.

3.1 Optimality conditions

For the q_h-subproblem (1.16a), it can be written as the following smooth and nonlinear optimization problem:

$$q_h^{k+1} = \underset{q_h \in K_h}{\arg\min} \left\{ J(q_h) + \frac{\beta}{2} \| q_h - p_h^k + \frac{\lambda_h^k}{\beta} \|_{L^2(\Omega)}^2 \right\}. \tag{3.1}$$

We further reformulate the problem (3.1) in Euclidean space and derive its first-order optimality conditions. Recall the definitions of $\boldsymbol{q},\ \boldsymbol{p},\ \boldsymbol{\lambda}$ and $J(\boldsymbol{q})$ in Section 2. The optimization problem (3.1) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{q} \in \mathbb{R}^N} &\left\{ J(\boldsymbol{q}) + \frac{\beta}{2} \| \boldsymbol{q} - \boldsymbol{p}^k + \frac{\boldsymbol{\lambda}^k}{\beta} \|_M^2 \right\}, \\ \text{s.t.} \quad & \boldsymbol{a}_0 \le \boldsymbol{q} \le \boldsymbol{a}_1, \end{aligned} \tag{3.2}$$

where M denotes the mass matrix as $M_{i,j} = (\phi_h^j, \phi_h^i)$, $\boldsymbol{a}_0 = a_0 \mathbf{1}$, and $\boldsymbol{a}_1 = a_1 \mathbf{1}$. Here, $\mathbf{1} \in \mathbb{R}^N$ denotes the vector with constant entries 1. The Lagrangian function of (3.2) is

$$l(\boldsymbol{q}, \boldsymbol{\eta}_0, \boldsymbol{\eta}_1) = J(\boldsymbol{q}) + \frac{\beta}{2} \| \boldsymbol{q} - \boldsymbol{p}^k + \frac{\boldsymbol{\lambda}^k}{\beta} \|_M^2 + \boldsymbol{\eta}_0^\top (\boldsymbol{q} - \boldsymbol{a}_0) + \boldsymbol{\eta}_1^\top (\boldsymbol{q} - \boldsymbol{a}_1),$$

with $\boldsymbol{\eta}_0, \boldsymbol{\eta}_1 \in \mathbb{R}^N$ the Lagrange multipliers. Then the corresponding KKT conditions are

$$\begin{cases} J'(\boldsymbol{q}) + \beta M(\boldsymbol{q} - \boldsymbol{p}^k + \dfrac{\boldsymbol{\lambda}^k}{\beta}) + (\boldsymbol{\eta}_0 + \boldsymbol{\eta}_1) = 0, \\ \boldsymbol{\eta}_1 \geq 0,\ \boldsymbol{q} \leq \boldsymbol{a}_1,\ \boldsymbol{\eta}_1^\top(\boldsymbol{q} - \boldsymbol{a}_1) = 0, \\ \boldsymbol{\eta}_0 \leq 0,\ \boldsymbol{q} \geq \boldsymbol{a}_0,\ \boldsymbol{\eta}_0^\top(\boldsymbol{q} - \boldsymbol{a}_0) = 0. \end{cases}$$

Furthermore, denoting $\boldsymbol{\eta} := \boldsymbol{\eta}_0 + \boldsymbol{\eta}_1$ and

$$C(\boldsymbol{q}, \boldsymbol{\eta}) := \boldsymbol{\eta} - \max\{0, \boldsymbol{\eta} + c(\boldsymbol{q} - \boldsymbol{a}_1)\} - \min\{0, \boldsymbol{\eta} + c(\boldsymbol{q} - \boldsymbol{a}_0)\},\ c > 0,$$

we can represent the KKT conditions as the equation

$$F(\boldsymbol{q}, \boldsymbol{\eta}) := \begin{pmatrix} J'(\boldsymbol{q}) + \beta M(\boldsymbol{q} - \boldsymbol{p}^k + \frac{\boldsymbol{\lambda}^k}{\beta}) + \boldsymbol{\eta} \\ C(\boldsymbol{q}, \boldsymbol{\eta}) \end{pmatrix} = 0. \tag{3.3}$$

3.2 Computation of the first-order derivative

To solve the q_h-subproblem (3.1), it is natural to consider the first-order derivative of $J(q_h)$ and probe its computational complexity. It follows from (2.4) that the first-order derivative of $J(q_h)$ satisfies

$$(\xi_h, J'(q_h)) = (\xi_h, -\frac{1}{2}|\nabla U(q_h)|^2 + \frac{1}{2}|\nabla u_\delta|^2),\ \forall\, \xi_h \in V_h,$$

where $U(q_h)$ is the solution to

$$a_h(q_h, U(q_h); \phi_h) = (f, \phi_h),\ \forall\, \phi_h \in \bar{V}_h = V_h \cap H_0^1(\Omega). \tag{3.4}$$

Let k_m with $m \geq 0$ be iteration counter for the inner loop for solving the q_h-subproblem at the k-th iteration; k_0 be the initial iterate for the inner loop. Then, how to compute $J'(\boldsymbol{q}^{k_m})$ can be summarized in the following Subroutine 1.

Subroutine 1 Computation of $J'(\boldsymbol{q}^{k_m})$.

```
1: function GRADIENT($\boldsymbol{q}^{k_m}$)
2:     Obtain $\bar{\boldsymbol{u}}^{k_m}$ via solving
           $$A_{k_m}\bar{\boldsymbol{u}}^{k_m} = \boldsymbol{f}, \tag{3.5}$$
       where $(A_{k_m})_{i,j} = (q_h^{k_m}\nabla\phi_h^j, \nabla\phi_h^i)$ and $\boldsymbol{f}$ defined as $\boldsymbol{f}_i = (f, \phi_h^i)$.
3:     Substitute $U(q_h^{k_m}) = \sum_{j=1}^N \bar{\boldsymbol{u}}_j^{k_m}\phi_h^j$ into the following equation to compute $J'(\boldsymbol{q}^{k_m})$
           $$(J'(\boldsymbol{q}^{k_m}))_i = (\phi_h^i, J'(q_h^{k_m})) = (\phi_h^i, -\frac{1}{2}|\nabla U(q_h^{k_m})|^2 + \frac{1}{2}|\nabla u_\delta|^2),\ i = 1, \ldots, N.$$
4:     return $J'(\boldsymbol{q}^{k_m})$.
5: end function
```

It is easy to see that the computation of $J'(q_h)$ requires values of $U(q_h)$, which should be obtained by computing A_{k_m} and solving the linear system (3.5) iteratively. Note that the linear system (3.5) is a discretized formulation of the elliptic equation (1.1). As analyzed in [40], its dimension and the condition number of the coefficient matrix A_{k_m} are both of order $O(h^{-d})$. Thus the linear system (3.5) is large-scaled and ill-conditioned for discretization with fine mesh, and computing $J'(q_h)$ may be expensive. Note that computing the objective function value in (3.1) requires values of $U(q_h)$ as well. Hence, these difficulties essentially imply that it is computationally demanding even if some first-order algorithm is applied to seek a medium- or low-accuracy numerical solution of the problem (3.1). Indeed, implementing a first-order algorithm usually requires certain line-search techniques with multiple computations of the objective function values, to discern appropriate step sizes. Our numerical experiments actually validate the failure of a number of popular first-order algorithms (such as the gradient projection method and the conjugate gradient projection method with backtracking line-search) firmly for solving the subproblem (3.1).

3.3 Active-set Newton method for the problem (3.1)

As analyzed, though the ADMM (1.16) per se can be easily derived, it is keen to solve the q_h-subproblem (1.16a), i.e., the problem (3.1). Because demanding computation is required yet only a medium- or low-accuracy solution can be targeted, it is not attractive to consider first-order algorithms for this subproblem. It is thus interesting to investigate how much more complicated if a second-order algorithm is applied to the problem (3.1). In this and the next subsections, we will show that, counter-intuitively, the Newtonian system of (3.3) can be appropriately reformulated and relaxed so that its computation reduces to solving a simple positive definite linear system, and then the benchmark active-set Newton method in, e.g., [24, 39], can be applied very efficiently. Computation of the Newton step is comparable with, and usually less than, that of a single iteration of the gradient projection method with some backtracking line-search strategy, while the accuracy is much higher. This is a convincing example of deriving model-tailored efficient algorithms by taking full advantage of the structure of the model under discussion.

To elaborate on the active-set Newton method for (3.3), let us define

$$\mathcal{A}_{k_m} = \mathcal{A}^+_{k_m} \cup \mathcal{A}^-_{k_m} \quad \text{and} \quad \mathcal{I}_{k_m} = \{1, \dots, N\} \setminus \mathcal{A}_{k_m}$$

as the sets of the active and inactive indices at $(\boldsymbol{q}^{k_m}, \boldsymbol{\eta}^{k_m})$, respectively, where $\mathcal{A}^+_{k_m}$ and $\mathcal{A}^-_{k_m}$ are the sets given respectively by

$$\mathcal{A}^+_{k_m} = \{i \,|\boldsymbol{\eta}^{k_m}_i + c(\boldsymbol{q}^{k_m} - \boldsymbol{a}_1)_i > 0\} \quad \text{and} \quad \mathcal{A}^-_{k_m} = \{i \,|\boldsymbol{\eta}^{k_m}_i + c(\boldsymbol{q}^{k_m} - \boldsymbol{a}_0)_i < 0\}.$$

For the mapping $F(\boldsymbol{q}^{k_m}, \boldsymbol{\eta}^{k_m})$ defined in (3.3), let $\partial F(\boldsymbol{q}^{k_m}, \boldsymbol{\eta}^{k_m})$ be the generalized Jacobian of (3.3) in sense of Clarke (see [13]). Then, as analyzed in [39], we have

$$F'(\boldsymbol{q}^{k_m}, \boldsymbol{\eta}^{k_m}) := \begin{pmatrix} J''(\boldsymbol{q}^{k_m}) + \beta M & I \\ -c\Pi_{\mathcal{A}_{k_m}} & \Pi_{\mathcal{I}_{k_m}} \end{pmatrix} \in \partial F(\boldsymbol{q}^{k_m}, \boldsymbol{\eta}^{k_m}), \tag{3.6}$$

where $\Pi_{\mathcal{A}_{k_m}}$ and $\Pi_{\mathcal{I}_{k_m}}$ denote the diagonal binary matrices with nonzero entries in $\mathcal{A}_{k_m}$ and $\mathcal{I}_{k_m}$, respectively. With (3.6), it is easy to see that the k_m-th iteration of the active-set Newton method for (3.3) is solving

$$\begin{pmatrix} J''(\boldsymbol{q}^{k_m}) + \beta M & I \\ -c\Pi_{\mathcal{A}_{k_m}} & \Pi_{\mathcal{I}_{k_m}} \end{pmatrix} \begin{pmatrix} \boldsymbol{q}^{k_m+1} - \boldsymbol{q}^{k_m} \\ \boldsymbol{\eta}^{k_m+1} - \boldsymbol{\eta}^{k_m} \end{pmatrix} = - \begin{pmatrix} J'(\boldsymbol{q}^{k_m}) + \beta M(\boldsymbol{q}^k - \boldsymbol{p}^k + \frac{\boldsymbol{\lambda}^k}{\beta}) \\ \boldsymbol{\eta}^{k_m} - \Pi_{\mathcal{A}^+_{k_m}}(\boldsymbol{\eta}^{k_m} + c(\boldsymbol{q}^{k_m} - \boldsymbol{a}_1)) - \Pi_{\mathcal{A}^-_{k_m}}(\boldsymbol{\eta}^{k_m} + c(\boldsymbol{q}^{k_m} - \boldsymbol{a}_0)) \end{pmatrix}. \tag{3.7}$$

Since the second equation of (3.7) implies that

$$(\boldsymbol{\eta}^{k_m+1})_{\mathcal{I}_{k_m}} = (\boldsymbol{\eta}^{k_m+1} - \boldsymbol{\eta}^{k_m})_{\mathcal{I}_{k_m}} + (\boldsymbol{\eta}^{k_m})_{\mathcal{I}_{k_m}} = 0,$$

we can remove those rows that belong to the indices in $\mathcal{I}_{k_m}$ from the second equation of (3.7), and simplify (3.7) as

$$\begin{pmatrix} \beta J''(\boldsymbol{q}^{k_m}) + \beta M & \mathcal{P}^\top_{\mathcal{A}_{k_m}} \\ \mathcal{P}_{\mathcal{A}_{k_m}} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{q}^{k_m+1} - \boldsymbol{q}^{k_m} \\ (\boldsymbol{\eta}^{k_m+1})_{\mathcal{A}_{k_m}} \end{pmatrix} = \begin{pmatrix} -J'(\boldsymbol{q}^{k_m}) - \beta M(\boldsymbol{q}^{k_m} - \boldsymbol{p}^k + \frac{\boldsymbol{\lambda}^k}{\beta}) \\ \mathcal{P}_{\mathcal{A}^+_{k_m}}\boldsymbol{a}_1 + \mathcal{P}_{\mathcal{A}^-_{k_m}}\boldsymbol{a}_0 - \mathcal{P}_{\mathcal{A}_{k_m}}\boldsymbol{q}^{k_m} \end{pmatrix}. \tag{3.8}$$

In (3.8), $\mathcal{P}_\mathcal{C}$ denotes the matrix consisting of those rows of $\Pi_\mathcal{C}$ that belong to the indices in a given set $\mathcal{C}$. Obviously, it holds that $\Pi_\mathcal{C} = \mathcal{P}^\top_\mathcal{C}\mathcal{P}_\mathcal{C}$.

Though it is trivial to analytically derive the system (3.8) for the k_m-th iteration of the active-set Newton method for (3.3), how to solve (3.8) numerically deserves meticulous analysis mainly because computing the second-order derivative $J''(\boldsymbol{q}^{k_m})$ is very expensive. Indeed, it follows from (2.5) that computing the second-order derivative $J''(\boldsymbol{q}^{k_m})$ directly at each iteration requires computing $\{U'(q_h^{k_m})\phi_h^i\}_{i=1}^N$ from (2.6) for each finite element basis function ϕ_h^i. This means a sequence of discretized elliptic equations in form of (3.5) are needed to be solved, and recall that each of them is large-scaled and ill-conditioned for fine mesh discretization.

To avoid computing $J''(\boldsymbol{q}^{k_m})$, we take an alternative approach to compute $\boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m}$. The key idea is substituting the discrete equations of (2.5) and (2.6) into (3.8) to eliminate $J''(\boldsymbol{q}^{k_m})$. To see the details, it follows from (2.5) that

$$J''(q_h^{k_m})(\xi_h, q_h^{k_{m+1}} - q_h^{k_m}) = -(\xi_h \nabla U(q_h^{k_m}), \nabla U'(q_h^{k_m})(q_h^{k_{m+1}} - q_h^{k_m})).$$

Then, we have

$$J''(\boldsymbol{q}^{k_m})(\boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m}) = -N_{k_m}\boldsymbol{r}^{k_{m+1}},$$

where

$$(N_{k_m})_{i,j} = (\phi_h^j \nabla U(q_h^{k_m}), \nabla \phi_h^i), \quad r_h^{k_{m+1}} = U'(q_h^{k_m})(q_h^{k_{m+1}} - q_h^{k_m}) = \sum_{i=1}^N \boldsymbol{r}_i^{k_{m+1}}\phi_h^i$$

and

$$\boldsymbol{r}^{k_{m+1}} = (\boldsymbol{r}_1^{k_{m+1}}, \ldots, \boldsymbol{r}_N^{k_{m+1}})^\top.$$

It also follows from (2.6) that

$$a_h(q_h^{k_m}, r_h^{k_{m+1}}; v_h) = -((q_h^{k_{m+1}} - q_h^{k_m})\nabla U(q_h^{k_m}), \nabla v_h), \ \forall\, v_h \in \bar{V}_h,$$

which implies that

$$A_{k_m}\boldsymbol{r}^{k_{m+1}} = -N_{k_m}^\top(\boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m}).$$

Thus we have

$$J''(\boldsymbol{q}^{k_m})(\boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m}) = -N_{k_m}\boldsymbol{r}^{k_{m+1}} \text{ with } A_{k_m}\boldsymbol{r}^{k_{m+1}} = -N_{k_m}^\top(\boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m}).$$

Next, substituting

$$J''(\boldsymbol{q}^{k_m})(\boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m}) = -N_{k_m}\boldsymbol{r}^{k_{m+1}}$$

into the Newtonian system (3.8), we obtain the under-determined linear system

$$\begin{pmatrix} -\beta N_{k_m}\boldsymbol{r}^{k_{m+1}} \\ 0 \end{pmatrix} + \begin{pmatrix} \beta M & \mathcal{P}_{\mathcal{A}_{k_m}}^\top \\ \mathcal{P}_{\mathcal{A}_{k_m}} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m} \\ (\boldsymbol{\eta}^{k_{m+1}})_{\mathcal{A}_{k_m}} \end{pmatrix} = \begin{pmatrix} -J'(\boldsymbol{q}^{k_m}) - \beta M(\boldsymbol{q}^{k_m} - \boldsymbol{p}^k + \frac{\boldsymbol{\lambda}^k}{\beta}) \\ \mathcal{P}_{\mathcal{A}_{k_m}^+}\boldsymbol{a}_1 + \mathcal{P}_{\mathcal{A}_{k_m}^-}\boldsymbol{a}_0 - \mathcal{P}_{\mathcal{A}_{k_m}}\boldsymbol{q}^{k_m} \end{pmatrix} \tag{3.9}$$

with respect to $(\boldsymbol{r}^{k_{m+1}}, \boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m}, (\boldsymbol{\eta}^{k_{m+1}})_{\mathcal{A}_{k_m}})$. Then, combining (3.9) with

$$A_{k_m}\boldsymbol{r}^{k_{m+1}} = -N_{k_m}^\top(\boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m}),$$

we obtain the following expanded linear system:

$$\begin{pmatrix} A_{k_m} & N_{k_m}^\top & 0 \\ -N_{k_m} & \beta M & \mathcal{P}_{\mathcal{A}_{k_m}}^\top \\ 0 & \mathcal{P}_{\mathcal{A}_{k_m}} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{r}^{k_{m+1}} \\ \boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m} \\ (\boldsymbol{\eta}^{k_{m+1}})_{\mathcal{A}_{k_m}} \end{pmatrix} = \begin{pmatrix} 0 \\ -J'(\boldsymbol{q}^{k_m}) - \beta M(\boldsymbol{q}^{k_m} - \boldsymbol{p}^k + \frac{\boldsymbol{\lambda}^k}{\beta}) \\ \mathcal{P}_{\mathcal{A}_{k_m}^+}\boldsymbol{a}_1 + \mathcal{P}_{\mathcal{A}_{k_m}^-}\boldsymbol{a}_0 - \mathcal{P}_{\mathcal{A}_{k_m}}\boldsymbol{q}^{k_m} \end{pmatrix}, \tag{3.10}$$

which is equivalent to the Newtonian system (3.8). Note that there is no need to compute the usually expensive $J''(\boldsymbol{q}^{k_m})$ in (3.10), and all the matrices $A_{k_m}, N_{k_m}, M, \mathcal{P}_{\mathcal{A}_{k_m}^+}$ and $\mathcal{P}_{\mathcal{A}_{k_m}^-}$ are easy to compute. For convenience, we denote

$$F^{k_m} := \begin{pmatrix} A_{k_m} & N_{k_m}^\top & 0 \\ -N_{k_m} & \beta M & \mathcal{P}_{\mathcal{A}_{k_m}}^\top \\ 0 & \mathcal{P}_{\mathcal{A}_{k_m}} & 0 \end{pmatrix} \text{ and } \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \\ \boldsymbol{d}_3 \end{pmatrix} := \begin{pmatrix} 0 \\ -J'(\boldsymbol{q}^{k_m}) - \beta M(\boldsymbol{q}^{k_m} - \boldsymbol{p}^k + \frac{\boldsymbol{\lambda}^k}{\beta}) \\ \mathcal{P}_{\mathcal{A}_{k_m}^+}\boldsymbol{a}_1 + \mathcal{P}_{\mathcal{A}_{k_m}^-}\boldsymbol{a}_0 - \mathcal{P}_{\mathcal{A}_{k_m}}\boldsymbol{q}^{k_m} \end{pmatrix}. \tag{3.11}$$

3.4 Schur complement reduction

Recall that the Newtonian system (3.10) is an expanded system of the linear saddle-point problem (3.8), and it is clear that (3.10) is indefinite. Moreover, because of the stiffness matrix A_{k_m} in its coefficient matrix F^{k_m}, the system (3.10) is also ill-conditioned. Hence, it is not easy to solve the Newtonian system (3.10). As analyzed in [6], there are two types of algorithms that can be used to solve (3.10): the *segregated* and *coupled* (also known as “all at once") methods.

Note that the right-bottom 2×2 block of the matrix F^{k_m} in (3.11)

$$\begin{pmatrix} \beta M & \mathcal{P}_{\mathcal{A}_{k_m}}^\top \\ \mathcal{P}_{\mathcal{A}_{k_m}} & 0 \end{pmatrix} \tag{3.12}$$

is well-conditioned and hence the variables $(\boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m})$ and $(\boldsymbol{\eta}^{k_{m+1}})_{\mathcal{A}_{k_m}}$ can be computed easily once $\boldsymbol{r}^{k_{m+1}}$ is obtained. We are thus inspired to choose the Schur complement reduction in [6], which is a major segregated approach, to convert the Newtonian system (3.10) to a linear system with only respect to the variable $\boldsymbol{r}^{k_{m+1}}$, by using the block factorization of the coefficient matrix F^{k_m} in (3.11). For the matrix F^{k_m}, it can be factorized as

$$F^{k_m} = \begin{pmatrix} I & \frac{1}{\beta}N_{k_m}^\top M^{-1} & R_{k_m} \\ 0 & I & 0 \\ 0 & \frac{1}{\beta}\mathcal{P}_{\mathcal{A}_{k_m}} M^{-1} & I \end{pmatrix} \begin{pmatrix} S_{k_m} & 0 & 0 \\ 0 & \beta M & 0 \\ 0 & 0 & -\frac{1}{\beta}(\mathcal{P}_{\mathcal{A}_{k_m}} M^{-1} \mathcal{P}_{\mathcal{A}_{k_m}}^\top) \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ -\frac{1}{\beta}M^{-1}N_{k_m} & I & \frac{1}{\beta}M^{-1}\mathcal{P}_{\mathcal{A}_{k_m}} \\ -R_{k_m}^\top & 0 & I \end{pmatrix}, \tag{3.13}$$

where

$$R_{k_m} = N_{k_m}^\top M^{-1} \mathcal{P}_{\mathcal{A}_{k_m}}^\top (\mathcal{P}_{\mathcal{A}_{k_m}} M^{-1} \mathcal{P}_{\mathcal{A}_{k_m}}^\top)^{-1}$$

and

$$S_{k_m} = A_{k_m} + \frac{1}{\beta} N_{k_m}^\top M^{-1} N_{k_m} - \frac{1}{\beta} N_{k_m}^\top M^{-1} \mathcal{P}_{\mathcal{A}_{k_m}}^\top (\mathcal{P}_{\mathcal{A}_{k_m}} M^{-1} \mathcal{P}_{\mathcal{A}_{k_m}}^\top)^{-1} \mathcal{P}_{\mathcal{A}_{k_m}} M^{-1} N_{k_m}$$

is the Schur complement of (3.12).

It is just seen that the Schur complement S_{k_m} requires computing $(\mathcal{P}_{\mathcal{A}_{k_m}} M^{-1} \mathcal{P}_{\mathcal{A}_{k_m}}^\top)^{-1}$. Note that the dimension of M^{-1} is of order $O(h^{-d})$. Therefore, it is extremely expensive for fine mesh cases to compute M^{-1} and hence the Schur complement S_{k_m}. To tackle this issue, we consider the lumped mass matrix (see [40]) to approximate the mass matrix M, which is a diagonal matrix with the row sums of the mass matrix M on the diagonal. That is, we have

$$W_{i,i} := \sum_{j=1}^{N} M_{i,j} = \sum_{j=1}^{N} (\phi_h^j, \phi_h^i), \quad i = 1, 2, \cdots, N, \tag{3.14}$$

with ϕ_h^i being the finite element basis functions in V_h. Then the inverse of the diagonal matrix W is easy to compute. Accordingly, the block matrix

$$\hat{F}^{k_m} := \begin{pmatrix} A_{k_m} & N_{k_m}^\top & 0 \\ -N_{k_m} & \beta W & \mathcal{P}_{\mathcal{A}_{k_m}}^\top \\ 0 & \mathcal{P}_{\mathcal{A}_{k_m}} & 0 \end{pmatrix}$$

is an approximation of F^{k_m} and its block diagonal decomposition is

$$\hat{F}^{k_m} = \underbrace{\begin{pmatrix} I & \frac{1}{\beta}N_{k_m}^\top W^{-1} & G_{k_m} \\ 0 & I & 0 \\ 0 & \frac{1}{\beta}\mathcal{P}_{\mathcal{A}_{k_m}} W^{-1} & I \end{pmatrix}}_{L_{k_m}} \underbrace{\begin{pmatrix} H_{k_m} & 0 & 0 \\ 0 & \beta W & 0 \\ 0 & 0 & -\frac{1}{\beta}(\mathcal{P}_{\mathcal{A}_{k_m}} W^{-1} \mathcal{P}_{\mathcal{A}_{k_m}}^\top) \end{pmatrix}}_{C_{k_m}} \underbrace{\begin{pmatrix} I & 0 & 0 \\ -\frac{1}{\beta}W^{-1}N_{k_m} & I & \frac{1}{\beta}W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}} \\ -G_{k_m}^\top & 0 & I \end{pmatrix}}_{R_{k_m}}, \tag{3.15}$$

where $G_{k_m} = N_{k_m}^\top W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top(\mathcal{P}_{\mathcal{A}_{k_m}}W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top)^{-1}$ and

$$H_{k_m} = A_{k_m} + \frac{1}{\beta}N_{k_m}^\top W^{-1}N_{k_m} - \frac{1}{\beta}N_{k_m}^\top W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top(\mathcal{P}_{\mathcal{A}_{k_m}}W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top)^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}W^{-1}N_{k_m}.$$

Also, it is easy to verify that the matrix $\mathcal{P}_{\mathcal{A}_{k_m}}W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top$ is diagonal. Thus, it is easy to compute $(\mathcal{P}_{\mathcal{A}_{k_m}}W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top)^{-1}$ and it holds that

$$W^{-1} - W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top(\mathcal{P}_{\mathcal{A}_{k_m}}W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top)^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}W^{-1} = \Pi_{\mathcal{I}_{k_m}}W^{-1}\Pi_{\mathcal{I}_{k_m}}.$$

Therefore, computational cost for the explicit formulation of the matrix G_{k_m} is negligible and the matrix H_{k_m} can be simplified as

$$H_{k_m} = A_{k_m} + \frac{1}{\beta}N_{k_m}^\top \Pi_{\mathcal{I}_{k_m}}W^{-1}\Pi_{\mathcal{I}_{k_m}}N_{k_m}. \tag{3.16}$$

Thus, computing M^{-1} is not required for the Schur complement H_{k_m} in (3.16) and it becomes easy to compute the block factorization (3.15) of $\hat{F}^{k_m}$. These features suggest us to relax the Newtonian system (3.10) to a linear system with the coefficient matrix $\hat{F}^{k_m}$.

Note that the Newtonian system (3.10) can be rewritten as

$$\hat{F}^{k_m}\begin{pmatrix} \boldsymbol{r}^{k_m+1} \\ \boldsymbol{q}^{k_m+1} - \boldsymbol{q}^{k_m} \\ (\boldsymbol{\eta}^{k_m+1})_{\mathcal{A}_{k_m}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \\ \boldsymbol{d}_3 \end{pmatrix} + \begin{pmatrix} 0 \\ \beta(W-M)(\boldsymbol{q}^{k_m+1} - \boldsymbol{q}^{k_m}) \\ 0 \end{pmatrix}. \tag{3.17}$$

The equation (3.17) is implicit because $\boldsymbol{q}^{k_m+1}$ appears in both sides. We consider a semi-implicit scheme for the Newtonian system (3.17) by replacing $\boldsymbol{q}^{k_m+1}$ in the right-hand side with the known last outer iterate $\boldsymbol{q}^k$, and obtain

$$\hat{F}^{k_m}\begin{pmatrix} \boldsymbol{r}^{k_m+1} \\ \boldsymbol{q}^{k_m+1} - \boldsymbol{q}^{k_m} \\ (\boldsymbol{\eta}^{k_m+1})_{\mathcal{A}_{k_m}} \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{d}}_1 \\ \hat{\boldsymbol{d}}_2 \\ \hat{\boldsymbol{d}}_3 \end{pmatrix}, \text{ where } \begin{pmatrix} \hat{\boldsymbol{d}}_1 \\ \hat{\boldsymbol{d}}_2 \\ \hat{\boldsymbol{d}}_3 \end{pmatrix} := \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \\ \boldsymbol{d}_3 \end{pmatrix} + \begin{pmatrix} 0 \\ \beta(W-M)(\boldsymbol{q}^{k} - \boldsymbol{q}^{k_m}) \\ 0 \end{pmatrix}. \tag{3.18}$$

That is, for numerical implementation purpose, we relax the Newtonian system (3.10) as the much easier linear system (3.18). Indeed, it follows from (3.15) that

$$\begin{pmatrix} \boldsymbol{r}^{k_m+1} \\ \boldsymbol{q}^{k_m+1} - \boldsymbol{q}^{k_m} \\ (\boldsymbol{\eta}^{k_m+1})_{\mathcal{A}_{k_m}} \end{pmatrix} = R_{k_m}^{-1}C_{k_m}^{-1}L_{k_m}^{-1}\begin{pmatrix} \hat{\boldsymbol{d}}_1 \\ \hat{\boldsymbol{d}}_2 \\ \hat{\boldsymbol{d}}_3 \end{pmatrix}.$$

Hence, the procedure of solving the linear system (3.18) via its Schur complement reduction can be summarized in Subroutine 2.

Subroutine 2 Solver for the Newtonian system (3.18).

```
1: function NEWTON-SOLVER($\hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3$)
2:     $(\hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3)^\top \leftarrow L_{k_m}^{-1}(\hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3)^\top$.
3:     $(\boldsymbol{r}^{k_m+1}, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3)^\top \leftarrow C_{k_m}^{-1}(\hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3)^\top$.
4:     $(\boldsymbol{r}^{k_m+1}, \boldsymbol{q}^{k_m+1} - \boldsymbol{q}^{k_m}, (\boldsymbol{\eta}^{k_m+1})_{\mathcal{A}_{k_m}})^\top \leftarrow R_{k_m}^{-1}(\boldsymbol{r}^{k_m+1}, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3)^\top$.
5:     return $(\boldsymbol{q}^{k_m+1} - \boldsymbol{q}^{k_m}, (\boldsymbol{\eta}^{k_m+1})_{\mathcal{A}_{k_m}})$.
6: end function
```

In Subroutine 2, the inverses of L_{k_m} and R_{k_m} are easy to compute as their permutation matrices are triangular. The computation of $C_{k_m}^{-1}(\hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3)^\top$ requires solving the linear system

$$H_{k_m}\boldsymbol{r} = \hat{\boldsymbol{d}}_1, \tag{3.19}$$

as well as computing $W^{-1}\hat{\boldsymbol{d}}_2$ and $(\mathcal{P}_{\mathcal{A}_{k_m}}W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top)^{-1}\hat{\boldsymbol{d}}_3$. Since both W and $\mathcal{P}_{\mathcal{A}_{k_m}}W^{-1}\mathcal{P}_{\mathcal{A}_{k_m}}^\top$ are diagonal matrices, the computational load of Subroutine 2 is dominated by solving (3.19). Note that (3.19) is easy because its coefficient matrix H_{k_m} is positive definite and its dimension is the same as that of (3.5), which is much less than that of (3.10). Thus, via Subroutine 2, the indefinite linear system (3.10) in higher dimension is significantly alleviated.

3.5 Optimization insights

As just shown, the indefinite linear system (3.10) is relaxed to (3.18) by its Schur complement reduction. Then it is interesting to analyze the corresponding relaxation of the underlying optimization problem and discern its difference from the desired one (3.2). Indeed, the linear system (3.18) is equivalent to:

$$F^{k_m}\begin{pmatrix}\boldsymbol{r}^{k_{m+1}}\\ \boldsymbol{q}^{k_{m+1}}-\boldsymbol{q}^{k_m}\\ (\boldsymbol{\eta}^{k_{m+1}})_{\mathcal{A}_{k_m}}\end{pmatrix}+\begin{pmatrix}0\\ \beta(W-M)(\boldsymbol{q}^{k_{m+1}}-\boldsymbol{q}^k)\\ 0\end{pmatrix}=\begin{pmatrix}\boldsymbol{d}_1\\ \boldsymbol{d}_2\\ \boldsymbol{d}_3\end{pmatrix}, \tag{3.20}$$

which differs from the Newtonian system (3.10) in the extra term $\beta(W-M)(\boldsymbol{q}^{k_{m+1}}-\boldsymbol{q}^k)$. Then, following the steps reversely in subsection 3.3, it is easy to see that (3.20) is exactly the corresponding Newtonian system if the active-set Newton method is applied to the following optimization problem:

$$\begin{aligned}&\min_{\boldsymbol{q}\in\mathbb{R}^N}\left\{J(\boldsymbol{q})+\frac{\beta}{2}\|\boldsymbol{q}-\boldsymbol{p}^k+\frac{\boldsymbol{\lambda}^k}{\beta}\|_M^2+\frac{\beta}{2}\|\boldsymbol{q}-\boldsymbol{q}^k\|_{W-M}^2\right\},\\ &\text{s.t.}\quad \boldsymbol{a}_0\le\boldsymbol{q}\le\boldsymbol{a}_1.\end{aligned} \tag{3.21}$$

Since each entry of the mass matrix M is positive [40], together with the definition of W in (3.14), it is easy to verify that $W-M$ is positive semidefinite. Then, the problem (3.21) is still convex and the solution of (3.18) converges to the solution of the problem (3.21). Note that we slightly abuse the notation and define $\frac{\beta}{2}\|\boldsymbol{q}-\boldsymbol{q}^k\|_{W-M}^2 := (\boldsymbol{q}-\boldsymbol{q}^k)^\top(W-M)(\boldsymbol{q}-\boldsymbol{q}^k)$ in (3.21), despite that $W-M$ is positive semidefinite. Furthermore, because of the equivalence between the Euclidean space and the space V_h, (3.21) can be rewritten as

$$\min_{q_h\in K_h}\left\{J(q_h)+\frac{\beta}{2}\|q_h-p_h^k+\frac{\lambda_h^k}{\beta}\|_{L^2(\Omega)}^2+\frac{\beta}{2}\|q_h-q_h^k\|_T^2\right\}. \tag{3.22}$$

In (3.22), the semi-norm

$$\|q_h\|_T^2 := \sum_{\tau\in\mathcal{T}_h}Q_{\tau,h}(q_h^2)-(q_h,q_h) \text{ with } Q_{\tau,h}(g)=\frac{|\tau|}{d+1}\sum_{j=1}^{d+1}g(x_j^\tau)$$

and $\{x_j^\tau\}_{j=1}^{d+1}$ are the vertices of the d-simplex $\tau\in\mathcal{T}_h$, and $\mathcal{T}_h$ is a regular partition of Ω. Hence, our numerical technique for tackling the difficult q_h-subproblem via solving (3.18) can be represented as replacing the problem (3.1) with (3.22), in which the objective function is regularized by a semi-proximal regularization term. In other words, applying the active-set Newton method along with the Schur complement reduction can be explained as replacing the optimization problem (3.1) by the proximally regularized one (3.21). Replacing the q_h-subproblem (1.16a) with (3.22) in the ADMM (1.16) hence results in the so-called proximal ADMM, which has been well studied in the optimization area. We refer to, e.g., [21, 22], for convergence of various proximal versions of the ADMM.

Remark 1. *For the implicit equation* (3.17)*, we can alternatively consider replacing the unknown $\boldsymbol{q}^{k_{m+1}}$ in the right-hand of* (3.17) *with the last inner iterate $\boldsymbol{q}^{k_m}$, instead of the last outer iterate $\boldsymbol{q}^k$. The resulting semi-implicit equation remains the coefficient matrix $\hat{F}^{k_m}$ and the right-hand side in* (3.10)*. In our numerical experiments, we* *use the warm start technique, meaning the initial iterate is set as $\boldsymbol{q}^{k_0} := \boldsymbol{q}^k$, and as to be shown in numerical results, usually each inner loop only requires executing the active-set Newton method by one iteration. Hence, using $\boldsymbol{q}^{k_m}$ or $\boldsymbol{q}^k$ makes very little difference numerically. On the other hand, an advantage of using $\boldsymbol{q}^k$ in (3.17) is that the resulting scheme can be theoretically explained as a proximal version of the ADMM with well known theoretical results as studied in the optimization area.*

3.6 Implementation of the active-set Newton method for the q_h-subproblem (3.18)

Now, we present the active-set Newton method for solving the q_h-subproblem (3.18) in Subroutine 3.

Subroutine 3 An active-set Newton method for the q_h-subproblem.

```
1:  function ASNEWTON($\boldsymbol{q}^k, \boldsymbol{p}^k, \boldsymbol{\lambda}^k$)
2:      Set initial values: $0 \leftarrow m$, $(\boldsymbol{q}^{k_m}, J'(\boldsymbol{q}^{k_m})) \leftarrow (\boldsymbol{q}^k, J'(\boldsymbol{q}^k))$; $\epsilon > 0$; "Tol" $> 0$.
3:      while $m \leq$ MaxIter do
4:          Compute the active and inactive indices: $\mathcal{A}^+_{k_m}, \mathcal{A}^-_{k_m}, \mathcal{A}_{k_m}, \mathcal{I}_{k_m}$.
5:          $\begin{pmatrix} \hat{\boldsymbol{d}}_1 \\ \hat{\boldsymbol{d}}_2 \\ \hat{\boldsymbol{d}}_3 \end{pmatrix} \leftarrow \begin{pmatrix} 0 \\ -J'(\boldsymbol{q}^{k_m}) - \beta W(\boldsymbol{q}^{k_m} - \boldsymbol{q}^k) - \beta M(\boldsymbol{q}^k - \boldsymbol{p}^k + \frac{\boldsymbol{\lambda}^k}{\beta}) \\ \mathcal{P}_{\mathcal{A}^+_{k_m}} \boldsymbol{a}_1 + \mathcal{P}_{\mathcal{A}^-_{k_m}} \boldsymbol{a}_0 - \mathcal{P}_{\mathcal{A}_{k_m}} \boldsymbol{q}^{k_m} \end{pmatrix}$.
6:          $(\boldsymbol{\eta}^{k_{m+1}})_{\mathcal{I}_{k_m}} \leftarrow 0$.
7:          $(\boldsymbol{q}^{k_{m+1}} - \boldsymbol{q}^{k_m}, (\boldsymbol{\eta}^{k_{m+1}})_{\mathcal{A}_{k_m}}) \leftarrow$ NEWTON-SOLVER($\hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2, \hat{\boldsymbol{d}}_3$), (See Subroutine 2).
8:          $\boldsymbol{q}^{k_{m+1}} = \max(\epsilon, \boldsymbol{q}^{k_{m+1}})$.
9:          $m \leftarrow m + 1$.
10:         $J'(\boldsymbol{q}^{k_m}) \leftarrow$GRADIENT($\boldsymbol{q}^{k_m}$), (See Subroutine 1).
11:         if $\|F(\boldsymbol{q}^{k_m}, \boldsymbol{\eta}^{k_m})\| <$ Tol then
12:             return $\boldsymbol{q}^{k+1} \leftarrow \boldsymbol{q}^{k_m}$, break.
13:         end if
14:     end while
15: end function
```

For each step of Subroutine 3, we need to solve two linear systems: (3.5) in Subroutine 1 and (3.19) in Subroutine 2. Since both of these linear systems are positive definite, we can use the preconditioned conjugate gradient (PCG) method to solve them. As (3.5) is a discretized formulation of the elliptic equation (1.1), a popular way to construct the preconditioner $\tilde{A}_{k_m}$ is using the multigrid (MG) method (see, e.g. [7]), which uses the MG V-cycles associated with its coefficient matrix A_{k_m} to approximate $A^{-1}_{k_m}$. For (3.19) in Subroutine 2, we still use the preconditioner $\tilde{A}_{k_m}$ of (3.5) for solving (3.19), though the MG V-cycles associated with H_{k_m} may be closer to $H^{-1}_{k_m}$. Note that we do not use the MG V-cycles associated with H_{k_m} because it is expensive to compute the explicit formulation of H_{k_m} and MG V-cycles require more computation if the explicit formulation of H_{k_m} is unknown.

Remark 2. *The update of $\boldsymbol{q}^{k_{m+1}}$ in Line 8 of Subroutine 3 is to ensure that $\boldsymbol{q}$ is positive. The norm $\|F(\boldsymbol{q}, \boldsymbol{\eta})\|$ is defined as*

$$\max(\|error_1\|_{W^{-1}}, \|error_2\|)$$

with

$$\begin{cases} error_1 := -J'(\boldsymbol{q}) - \beta W(\boldsymbol{q} - \boldsymbol{q}^k) - \beta M(\boldsymbol{q}^k - \boldsymbol{p}^k + \dfrac{\boldsymbol{\lambda}^k}{\beta}) + \boldsymbol{\eta}, \\ error_2 := \boldsymbol{\eta} - \max\{0, \boldsymbol{\eta} + c(\boldsymbol{q} - \boldsymbol{a}_1)\} - \min\{0, \boldsymbol{\eta} + c(\boldsymbol{q} - \boldsymbol{a}_0)\}. \end{cases}$$

Remark 3. *Since the convexity of the functional $J(\cdot)$ is kept, the active-set Newton method summarized as Subroutine 3 is guaranteed to be convergent to a solution of (3.2). We refer to, e.g., [24], for rigorous analysis. Also, as analyzed in [24], Subroutine 3 is superlinearly convergent.*

Remark 4. *It is clear that the computational cost of Subroutine 3 is dominated by computing the gradient $J'(\boldsymbol{q}^{k_m})$ (Subroutine 1) and the Newton step (Subroutine 2). To compute the gradient $J'(\boldsymbol{q}^{k_m})$, its main computation is computing the discrete matrix A_{k_m} and then solving the discretized elliptic equation (3.5). For the Newton step, its main computation is calculating the discrete matrix N_{k_m} and then solving the linear system (3.19). As just analyzed, the linear system (3.19) is positive definite and its dimension is the same as that of (3.5). Hence, the Newton step does not require too much additional computation, compared with the computation of the gradient $J'(\boldsymbol{q}^{k_m})$. Recall that implementing a first-order algorithm usually requires discerning an appropriate step size (e.g., via line-search techniques) for the sake of ensuring the convergence, hence multiple objective function values are usually required. As mentioned, computing these functional values is equally expensive as that of computing the gradient. Therefore, it is encouraging to consider the active-set Newton method in Subroutine 3 whose computation is not much more than that of implementing a first-order algorithm, yet its convergence is guaranteed to be superlinear.*

4 Deep CNN for the p_h-subproblem

In this section, we discuss how to solve the p_h-subproblem (1.16b). This subproblem can be specified as

$$p_h^{k+1} = \arg\min_{p_h \in V_h} \left\{ \alpha \|\nabla p_h\|_{L^1(\Omega)} + \frac{\beta}{2} \|q_h^{k+1} - p_h + \frac{\lambda_h^k}{\beta}\|^2_{L^2(\Omega)} \right\}. \tag{4.1}$$

Note that the original TV term is kept and hence the objective functional in (4.1) is nonsmooth. Obviously, (4.1) has no closed-form solution and it should be solved iteratively by a certain algorithm. Also, the dimension of p_h is the same as that of q_h, and it may be high for a higher-dimensional space and fine mesh discretization. For instance, it is of order 10^6 if the mesh size $h = 1/1024$ for the unit square $\Omega \subset \mathbb{R}^2$. Hence, it is also necessary to consider how to solve the p_h-subproblem (1.16b) efficiently for implementing the ADMM (1.16). We reiterate that it is always more preferable to choose some model-tailored algorithms in accordance with the structure of the problem under consideration. For the p_h-subproblem (4.1), certainly it can be treated as a generic optimization problem and then some generic-purpose or less structure-exploiting algorithms can be applied. But it turns out that the deep convolutional neural network (CNN), which has been significantly enhanced in recent literatures (e.g., [38, 45]), is a much better choice for the p_h-subproblem (4.1). Below is the detail.

Let $R(x) := \|\nabla p_h\|_{L^1(\Omega)}$ and $\theta > 0$ be constant. The proximal operator of $R(x)$ is given by

$$\text{Prox}_{\theta R(x)}(z) = \arg\min_x \left\{ \theta R(x) + \frac{1}{2}\|x - z\|^2_{L^2(\Omega)} \right\}. \tag{4.2}$$

Then, the solution of the p_h-subproblem (4.1) can be presented by

$$p_h^{k+1} = \text{Prox}_{\frac{\alpha}{\beta}R(x)}(q_h^{k+1} + \frac{\lambda_h^k}{\beta}).$$

Following the standard Rudin-Osher–Fatemi model in [44], the operator $\text{Prox}_{\frac{\alpha}{\beta}R(x)}$ can be interpreted as the denoising operator for the standard image denoising model. In the last few years, the literature of algorithms for various image denoising models has been phenomenally upgraded by contemporary deep neural networks, see, e.g., [41, 47, 48, 49]. An advantage of applying a deep neural network to denoising models is that it avoids iterations in its testing phase, and hence computation can be largely saved. We are thus inspired to consider some pre-trained deep neural network, rather than some iterative scheme, for the p_h-subproblem (4.1).

To see why the deep CNN is chosen for the case where $\Omega \subset \mathbb{R}^2$ is a rectangular domain and it is triangulated into the uniform mesh, there exists a one-to-one mapping between $p_h \in V_h$ and an $m \times n$ raster image ($m \times n = N$) where the gray value at pixel (i, j) of the image I corresponds to the value of the function p_h at node (i, j). Thus, there is a mapping between a discrete two-dimensional function and a gray-scale raster image. Then, the pre-trained deep CNN which has been widely used for various image denoising problems can be applied.

Among various networks in the literatures such as [41, 47, 48, 49], we choose the deep CNN in [49]. Given the depth D, the architecture of the deep CNN network for a gray image in [49] is shown in Figure 1. There are three types of layers: (i) Conv+ReLU: in the first layer, convolutions (Conv) with 64 filters of size 3×3 are used to generate 64 feature maps, and rectified linear units (ReLU, $\max(0, \cdot)$) are then utilized for nonlinearity. (ii) Conv+BN+ReLU: in layers 2 to $D-1$, convolutions with 64 filters of size 3×3 are used, and batch normalization (BN) [29] is added between convolution and ReLU. (iii) Conv: in the last layer, convolutions with filters of size $3 \times 3 \times 64$ are used to reconstruct the output. Note that the output of the network is the residual image. The denoised image should be "Noisy Image" minus "Residual Image".

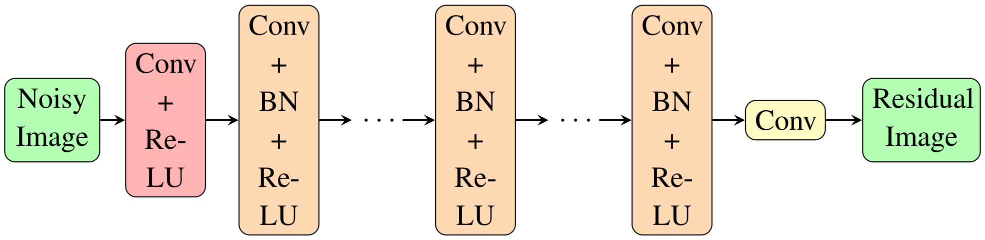

Figure 1: The architecture of the deep CNN network in [49].

Let $\mathcal{M}$ denote the mapping from $\boldsymbol{p}$ to a raster image, and $\mathcal{C}_\sigma$ the pre-trained deep CNN with σ the variance of the noise used for training CNN. Solving the p_h-subproblem (4.1) by a pre-trained deep CNN can be summarized in Subroutine 4.

Subroutine 4 A deep CNN based method for the p_h-subproblem.

```
1: function DENOISER($\boldsymbol{q}^{k+1}, \boldsymbol{\lambda}^k$)
2:     $I^{k+1}_{input} := \mathcal{M}(\boldsymbol{q}^{k+1} + \frac{\boldsymbol{\lambda}^k}{\beta})$.
3:     $I^{k+1}_{output} := I^{k+1}_{input} - \mathcal{C}_\sigma(I^{k+1}_{input})$.
4:     $\boldsymbol{p}^{k+1} = \mathcal{M}^{-1}(I^{k+1}_{output})$.
5: end function
```

Remark 5. *Our primary interest is the case where $\Omega \subset \mathbb{R}^2$ is a rectangular domain and it is partitioned by the uniform triangulation mesh. For other cases such as Ω is not rectangular, the mesh is not uniform, or the deep CNN is not trained based on raster images, the mapping $\mathcal{M}$ should be redefined. For the case where $\Omega \subset \mathbb{R}^3$, one may employ a deep 3D CNN (see, e.g., [30]). These much more complicated situations should be discussed case by case with significantly more techniques, and they are beyond the scope of this paper.*

5 The ADMM-Newton-CNN numerical approach

With the discussions in Sections 3 and 4, we are ready to present the complete version of the ADMM-Newton-CNN numerical approach to the TV model (1.2). We show the flowchart of its implementation in Figure 2.

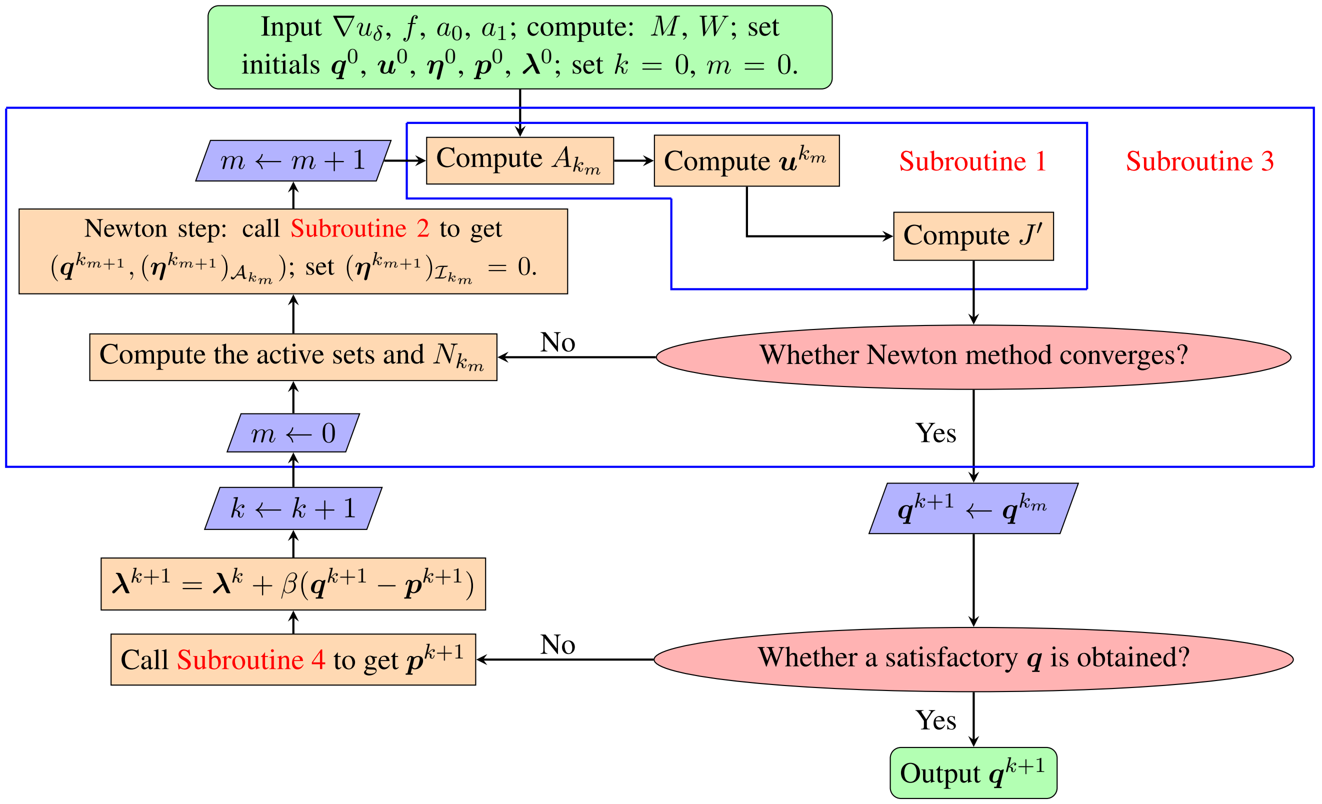
Input ∇u_δ, f, a_0, a_1; compute: M, W; set initials $\boldsymbol{q}^0$, $\boldsymbol{u}^0$, $\boldsymbol{\eta}^0$, $\boldsymbol{p}^0$, $\boldsymbol{\lambda}^0$; set $k=0$, $m=0$.
$m \leftarrow m+1$
Compute A_{k_m}
Compute $\boldsymbol{u}^{k_m}$
Subroutine 1
Subroutine 3
Newton step: call Subroutine 2 to get $(\boldsymbol{q}^{k_{m+1}}, (\boldsymbol{\eta}^{k_{m+1}})_{\mathcal{A}_{k_m}})$; set $(\boldsymbol{\eta}^{k_{m+1}})_{\mathcal{I}_{k_m}} = 0$.
Compute J'
Compute the active sets and N_{k_m}
No
Whether Newton method converges?
$m \leftarrow 0$
Yes
$k \leftarrow k+1$
$\boldsymbol{q}^{k+1} \leftarrow \boldsymbol{q}^{k_m}$
$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta(\boldsymbol{q}^{k+1} - \boldsymbol{p}^{k+1})$
Call Subroutine 4 to get $\boldsymbol{p}^{k+1}$
No
Whether a satisfactory $\boldsymbol{q}$ is obtained?
Yes
Output $\boldsymbol{q}^{k+1}$

Figure 2: Flowchart of implementation of the proposed ADMM-Newton-CNN approach.

6 Numerical results

In this section, we show efficiency of the proposed ADMM-Newton-CNN numerical approach by preliminary numerical results. All codes were written in MATLAB R2020b and numerical experiments were conducted on a desktop with Windows 10, Intel(R) Core(TM) i9-9900KF CPU (3.60 GHz), and 128 GB RAM. We notice that there are some limited numerical studies in the literature [11, 51], which are focused on the smoothing TV_ϵ model (1.5) with $d = 1$ and coarse mesh discretization. But the proposed ADMM-Newton-CNN approach is for the original TV model (1.2) with the focus on the higher-dimensional space of $d = 2$ as well as fine mesh discretization. Hence, it seems difficult to make any numerical comparison with the mentioned existing works, because of the essentially different natures in both modeling and philosophy of algorithmic design.

6.1 Experiment setups

We fix Ω as $(0,1)\times(0,1)$ and $f = 10$ in Ω. The domain Ω is partitioned by the uniform triangulation mesh in the iFEM package [10]. The lower and upper bounds a_0 and a_1 in the constrained set K in (1.4) are taken as 0.1 and 5.0, respectively. We follow [11, 33] and construct examples for the test in the following way.

(1.) Choose a discontinuous diffusion coefficient $q(x) \in L^1(\Omega)$.

(2.) Compute the finite element solution u_h of (1.1).

(3.) Take the noisy observation data as $\nabla u_\delta(x) = \nabla u_h + \delta\|\nabla u_h\|_h \mathrm{rand}(x)$, where $\mathrm{rand}(x)$ is a uniformly distributed random vector-valued function in $[-1,1]$ with $\delta > 0$ the noise level.

Recall that, for the q_h-subproblem (1.16a), we use the warm start technique for the PCG executions, and the implementation of MG V-circles is based on the iFEM package developed in [10] with Jacobi splitting. Moreover, for the p_h-subproblem (1.16b), we use the same network architecture, the same training dataset, and the original code as [49] (see https://github.com/cszn/DnCNN) to train totally 25 CNN networks for the cases where $\sigma = 1, 2, \cdots, 25$. Hence, C_σ in Subroutine 4 is chosen as one of these pre-trained 25 CNN networks with a specified value of σ. The mapping $\mathcal{M}$ in Subroutine 4 is specified as $\mathcal{M}(\boldsymbol{p}) = (\boldsymbol{p} - \boldsymbol{a}_0)/(a_1 - a_0) \times 256$, where $\boldsymbol{p}$ is the coefficient of p_h and I is a raster image. The initial guess of the Lagrange multiplier $\boldsymbol{\lambda}^0$ is always set to be $\mathbf{0}$; the initial guess of $\boldsymbol{q}^0$, $\boldsymbol{u}^0$ and $\boldsymbol{p}^0$ are set to be $\mathbf{1}$, $\mathbf{1}$ and $\mathbf{0}$, respectively. For the stopping criterion to solve the linear systems (3.5) and (3.19), relative errors are controlled with the tolerances of 10^{-10} and 10^{-5}, respectively. In addition, the value of "Tol" in Subroutine 3 is 10^{-3}.

6.2 Experimental results

Example 1. *We take the discontinuous coefficient $q(x, y)$ in $(0, 1) \times (0, 1)$ as*

$$q(x,y) = \begin{cases} 1, & y \in [0, 0.5], \\ 2, & y \in (0.5, 1], \end{cases}$$

whose discontinuous points form a straight line.

For the penalty parameter β and denoising parameter σ, generally they should be tuned according to the noise level δ. According to the Morozov's discrepancy principle [14], the value of α in (1.2) is positively correlated with the noise level δ, and the parameters σ in $\mathcal{C}_\sigma$ and $\theta = \frac{\alpha}{\beta}$ in (4.2) play the same role of controlling the rate of denoising. Hence, σ should be proportional to $\theta = \frac{\alpha}{\beta}$ and $\beta\sigma$ should be positively correlated with the noise level δ. In our numerical experiments, we tune the parameters σ and β such that $\beta\sigma$ is proportional to the noise level of the observation, i.e., $\beta\sigma \sim \delta$. In Table 1, we list the tuned values of β and σ for the cases where the noise levels are $\delta = 0.01$, 0.05 and 0.1, respectively. These parameters are kept as constants for different finite element meshes.

Table 1: Parameters δ, β and σ for Example 1.

δ	β	σ	$(\beta\sigma)/\delta$
0.01	0.1	9	90
0.05	0.5	9	90
0.1	0.6	15	90

As discussed in subsection 3.5 and Remark 3, the proposed ADMM-Newton-CNN approach is guaranteed to be convergent and our main interest is to show how numerically efficient this scheme could be. We have observed that the iterative sequence tends to be convergent after about 30 iterations. Hence, we record the numerical performance in Table 2 for the first 50 iterations. For succinctness, only several choices of the noise levels and the meshes are listed. It is encouraging to see that total numbers of Newton steps, and PCG numbers for solving the linear systems (3.5) and (3.19), as well as the relative error to the true solution $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)}$, are all very robust to the mesh. Since the dimension of the resulting subproblems is increased when the mesh is refined, this feature is particularly favorable for fine mesh discretization.

To take a closer look into computing time of individual subtasks, we focus on the case of $h = 1/256$ and report the respective computing times of various subtasks of the first 50 iterations in Table 3. According to this table, we see that computing time for the linear system (3.19) accounts for about 20-40% of the entire time. Especially, for the cases where $\delta = 0.01$ and 0.05, computing time for (3.19) is less than that of the CNN implementation. This fact well explains that the preconditioner $\tilde{A}_{k_m}$ is a good choice for the linear system (3.19). The Newton step is hence computationally cheap because the linear system (3.19) can be well solved with the preconditioner $\tilde{A}_{k_m}$. Recall that the computation of both $J'(\boldsymbol{q}^{k_m})$ and the objective function value mainly consists of computing A_{k_m} and solving the linear system (3.5). Also, the Newton step needs to compute N_{k_m} and solve the linear system (3.19). Based on Table 3, it is easy to estimate that the computation time of the Newton step is only about three

Table 2: Numerical results for Example 1 after the first 50 iterations.

δ	h	Total Newton No.	Total PCG No. for (3.5)/(3.19)	CPU Time (s)	$\|q_h^{50} - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)}$
0.01	1/64	57	661 / 2238	2.425	0.0068
	1/128	58	705 / 2722	9.327	0.0055
	1/256	60	764 / 3239	45.847	0.0053
	1/512	62	814 / 3635	199.392	0.0044
	1/1024	63	855 / 3825	881.665	0.0052
0.05	1/64	55	636 / 1211	2.093	0.0106
	1/128	55	660 / 1388	7.550	0.0122
	1/256	55	679 / 1533	35.839	0.0139
	1/512	55	705 /1659	148.891	0.0273
	1/1024	55	724 / 1705	649.436	0.0486
0.1	1/64	55	638 / 1075	2.063	0.0405
	1/128	55	666 / 1232	7.363	0.0374
	1/256	55	683 / 1348	34.702	0.0369
	1/512	55	716 / 1448	144.838	0.0478
	1/1024	55	741 / 1500	631.207	0.0706

Table 3: Computing time of various subtasks for Example 1 with $h = 1/256$ after the first 50 iterations.

	δ=0.01			δ=0.05			δ=0.1		
Subtasks	No.	Total time (s)	%Time	No.	Total time (s)	%Time	No.	Total time (s)	%Time
Linear system (3.19)	60	16.901	36.9%	55	8.300	23.2%	55	7.178	20.7%
Implementation of CNN	50	9.256	20.8%	50	9.810	27.4%	50	9.556	27.5%
N_{k_m}	60	4.578	10.0%	55	4.099	11.4%	55	4.179	12.0%
A_{k_m}	60	3.789	8.3%	55	3.440	9.6%	55	3.478	10.0%
Linear system (3.5)	60	3.196	7.0%	55	2.842	7.9%	55	2.897	8.3%
Others		7.857	17.0%		7.347	20.5%		7.414	21.5%
Total		45.847	100%		35.839	100%		34.702	100%

times of that of computing $J'(\boldsymbol{q}^{k_m})$. Hence, choosing the active-set Newton method in Subroutine 3 for (3.2), instead of some first-order algorithm which generally requires computing the objective function values repeatedly to find an appropriate step size, is verified.

In Figure 3, we plot the curves of $\|\nabla u_h^k - \nabla u_h\|_h$ and $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)}$ for Example 1 with $h = 1/256$. Also, differences between the ground-truth solution q and the numerical solutions q_h^k at the 30-th iteration are plotted in Figure 4. These curves further display the efficiency of the proposed ADMM-Newton-CNN approach for Example 1. In this figure, "expectation" means $E[\|\nabla u_h^k - \nabla u_h\|_h]$.

Example 2. *The discontinuous coefficient $q(x, y)$ is taken as*

$$\begin{aligned} q(x, y) &= 1 + 0.5 I_{\Omega_1} + I_{\Omega_2}, \\ \Omega_1 &= \{(x, y)|(x - 0.5)^2 + (y - 0.5)^2 \le 1/8\}, \\ \Omega_2 &= \{(x, y)|1/3 \le x \le 2/3, 1/3 \le y \le 2/3\}, \end{aligned}$$

where I_{Ω_k} denotes the characteristic function over Ω_k, $k = 1, 2$. Its discontinuous points form a circle and a square. This example has right-angled and curved discontinuous points, and it is more complicated.

Values of the parameters β and σ for various noise levels δ are listed in Table 4. Again, values of σ are set such that $\beta\sigma$ is proportional to the noise level of the observation, i.e., $\beta\sigma \sim \delta$, and these parameters are kept as constants for different finite element meshes.

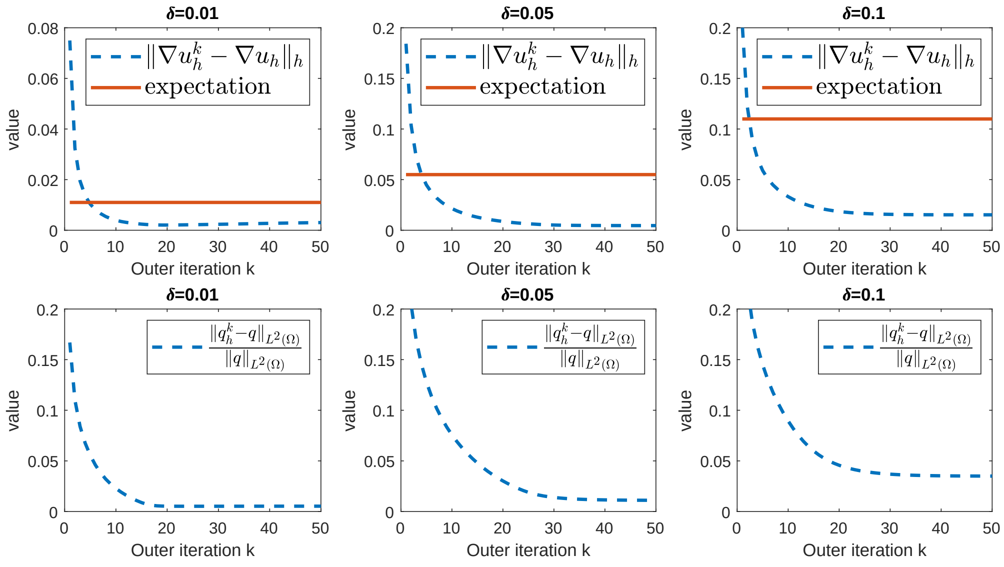

Figure 3: Plots of $\|\nabla u_h^k - \nabla u_h\|_h$ and $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)}$ for Example 1 with $h = 1/256$.

Table 4: Parameters δ, β and σ for Example 2.

δ	β	σ	$(\beta\sigma)/\delta$
0.01	0.06	12	84
0.05	0.3	12	84
0.1	0.3	24	84

Numerical results are reported in Table 5, for the first 50 iterations when the proposed ADMM-Newton-CNN approach is applied to Example 2. We list computing time of various subtasks individually in Table 6, for the first 50 iterations and $h = 1/256$. Moreover, in Figure 5, $\|\nabla u_h^k - \nabla u_h\|_h$ and $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)}$ are plotted for the first 50 iterations when $h = 1/256$. In this figure, "expectation" means $E[\|\nabla u_h^k - \nabla u_h\|_h]$. In Figure 6, difference between the ground-truth solution q and q_h^k at the 30-th iteration are plotted for the case where $h = 1/256$. Similar conclusions as those for Example 1 can be drawn, and efficiency of the proposed ADMM-Newton-CNN approach is further verified for Example 2.

7 Conclusions

We focus on a well-known model with the total variational (TV) regularization for identifying the diffusion coefficient in an elliptic equation with observation data of the gradient of the solution. We consider the original TV-regularized model without any relaxation so that the favorable nonsmoothness and convexity properties can be both kept. We propose to solve this model by the alternating direction method of multipliers (ADMM), and show that the resulting subproblems can be solved effectively by the active-set Newton method and the convolutional neural network (CNN), respectively. The proposed ADMM-Newton-CNN approach is validated to be very efficient for the 2-dimensional space case with fine mesh discretization. This work enhances the current literatures in which only the 1-dimensional space case with coarse mesh discretization can be numerically tackled for some smoothing and thus inaccurate surrogate models. As [49], we use the training datasets which consist of 400 piecewise constant images in size of 180×180 for training the CNNs in our experiments. The generalization property of the trained CNNs seems to still guarantee its efficiency for identifying the discontinuous diffusion coefficients under discussion, although these datasets are generic-purpose. If some specific datasets for the discontinuous diffu-

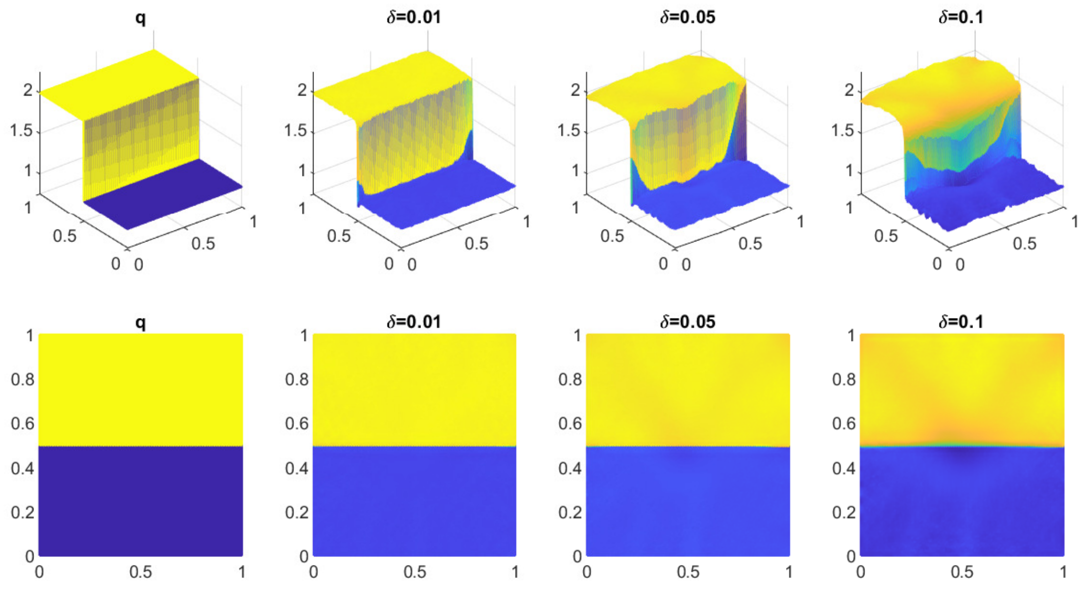

Figure 4: Numerical solutions q_h^k at the 30-th iteration for Example 1 with $h = 1/256$. Column 1: the true coefficient q; Column 2: $\delta = 0.01$ and the relative error $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)} = 0.0051$; Column 3: $\delta = 0.05$ and the relative error $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)} = 0.0138$; Column 4: $\delta = 0.1$ and the relative error $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)} = 0.0368$; Bottom: the projection of q or q_h^k onto the domain Ω.

sion coefficient problem are available, then it is much preferred to fine-tune existing networks on smaller but more specific datasets.

A relevant yet much more challenging problem is to solve TV-regularized models for identifying the diffusion coefficient in an elliptic equation with observation data of function values of the solution, as studied in [9, 11]. This problem is nonconvex and thus intrinsically different from the convex model (1.2). To extend the proposed ADMM-Newton-CNN approach to this nonconvex problem, it is keen to consider how to handle the nonconvex subproblems both theoretically and numerically. It is also interesting to extend the philosophy of algorithmic design, as well as the numerical techniques initiated in this paper, to other parameter identification problems for diffusion coefficients and advection coefficients arising in some elliptic systems with other types of objective functionals, or in some complicated PDE systems.

Acknowledgment

The first author was partially supported by the National Natural Science Foundation of China (No. 12071343). The second author was supported by Seed Fund for Basic Research (Project No. 202011159106) from the University of Hong Kong. The third author was partially supported by the Institute of Mathematical Research, Department of Mathematics, the University of Hong Kong.

References

[1] R. Acar and C. R. Vogel, *Analysis of bounded variation penalty methods for ill-posed problems*, Inverse Problems, 10 (1994), pp. 1217–1229.

[2] H. Attouch, G. Buttazzo, and G. Michaille, *Variational analysis in Sobolev and BV spaces: Applications to PDEs and Optimization*, SIAM/MPS, Philadelphia, PA, 2006.

Table 5: Numerical performance for Example 2 after the first 50 iterations.

δ	h	Total Newton No.	Total PCG No. for (3.5)/(3.19)	CPU Time (s)	$\Vert q_h^{50} - q\Vert_{L^2(\Omega)}/\Vert q\Vert_{L^2(\Omega)}$
0.01	1/64	53	617 / 3156	2.650	0.0222
	1/128	53	636 / 3543	9.815	0.0136
	1/256	53	662 / 3801	46.936	0.0104
	1/512	53	673 / 3973	193.944	0.0094
	1/1024	53	710 / 4059	841.498	0.0096
0.05	1/64	54	655 / 1718	2.268	0.0852
	1/128	53	656 / 1824	7.964	0.0545
	1/256	53	679 / 1932	37.295	0.0388
	1/512	53	696 / 1985	154.104	0.0404
	1/1024	53	722 / 2012	666.666	0.0604
0.1	1/64	54	651 / 1591	2.278	0.2160
	1/128	54	682 / 1772	8.030	0.1903
	1/256	53	690 / 1829	36.437	0.1558
	1/512	53	715 / 1915	152.632	0.1419
	1/1024	51	738 / 1967	662.879	0.1462

Table 6: Computation time for Example 2 with $h = 1/256$ after the first 50 iterations.

	δ=0.01			δ=0.05			δ=0.1		
Subtasks	No.	Total time (s)	%Time	No.	Total time (s)	%Time	No.	Total time (s)	%Time
Newtonian system (3.19)	53	19.773	42.1%	53	10.182	27.3%	53	9.542	26.2%
Implementation of CNN	50	9.729	20.7%	50	9.730	26.1%	50	9.425	25.9%
N_{k_m}	53	4.064	8.7%	53	4.035	10.8%	53	4.025	11.0%
A_{k_m}	53	3.374	7.2%	53	3.367	9.0%	53	3.361	9.2%
Linear system (3.5)	53	2.792	5.9%	53	2.833	7.6%	53	2.906	8.0%
Others		7.205	15.4%		7.149	19.2%		7.178	19.7%
Total		46.936	100%		37.295	100%		36.437	100%

[3] H. T. BANKS AND K. KUNISCH, *Estimation Techniques for Distributed Parameter Systems*, Birkhäuser, Boston, 1989.

[4] S. BARTELS, *Total variation minimization with finite elements: Convergence and iterative solution*, SIAM J. Numer. Anal., 50 (2012), pp. 1162–1180.

[5] S. BARTELS AND M. MILICEVIC, *Stability and experimental comparison of prototypical iterative schemes for total variation regularized problems*, Comput. Methods Appl. Math., 16 (2016), pp. 361–388.

[6] M. BENZI, G.H GOLUB AND J. LIESEN, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[7] W.L. BRIGGS, V.E. HENSON AND S.F. MCCORMICK, *A multigrid tutorial*, SIAM, 2000.

[8] T. F. CHAN AND X.-C. TAI, *Augmented Lagrangian and total variational methods for recovering discontinuous coefficients from elliptic equations*, CAM Report 97-2, UCLA, Los Angeles, CA, 1997.

[9] T. F. CHAN AND X.-C. TAI, *Identification of discontinuous coefficients in elliptic problems using total variation regularization*, SIAM J. Sci. Comput., 25 (2003), pp. 881–904.

[10] L. CHEN, *iFEM: an integrated finite element method package in MATLAB*, Tech. Report, University of California at Irvine, 2009.

[11] Z. CHEN AND J. ZOU, *An augmented Lagrangian method for identifying discontinuous parameters in elliptic systems*, SIAM J. Control Optim., 37 (1999), pp. 892–910.

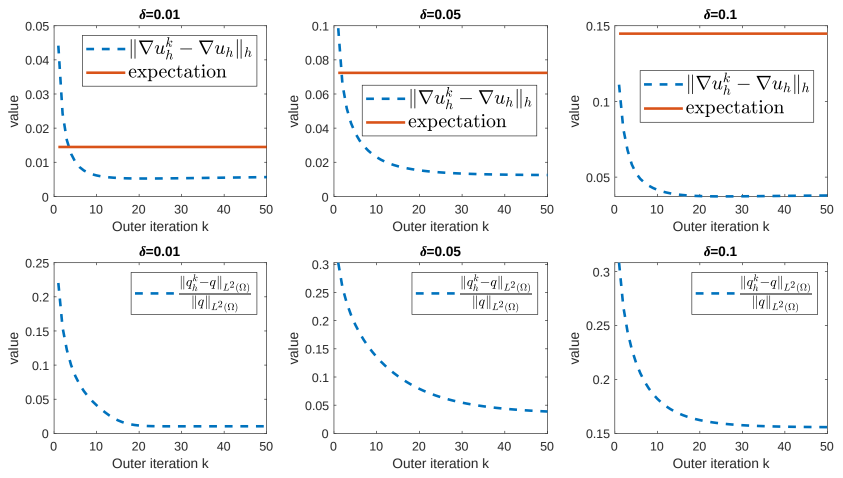

Figure 5: Plots of $\|\nabla u_h^k - \nabla u_h\|_h$ and $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)}$ for Example 2 with $h = 1/256$.

[12] P. Clément, *Approximation by finite element functions using local regularization*, R.A.I.R.O. Anal. Numér., 9 (1975), pp. 77-84.

[13] F.H. Clarke, *Optimization and nonsmooth analysis*, Wiley, New York, 1983.

[14] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*, Kluwer Academic Publishers Group, Dordrecht, 1996.

[15] L. C. Evans, *Partial differential equations*, American Mathematical Society, Providence, RI, second ed., 2010.

[16] R. Glowinski and A. Marrocco, *Sur l'approximation par éléments finis et la résolution par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires*, R.A.I.R.O. Anal. Numér., 9 (1975), pp. 41–76.

[17] R. Guenther and R. Hudspeth and W. McDougal and J. Gerlach, *Remarks on parameter identification. I*, Numer. Math., 47 (1985), pp. 355-361.

[18] D. N. Hào and T. N. T. Quyen, *Convergence rates for Tikhonov regularization of coefficient identification problems in Laplace-type equations*, Inverse Problems, 26 (2010), p. 125014 (23pp).

[19] D. N. Hào and T. N. T. Quyen, *Convergence rates for total variation regularization of coefficient identification problems in elliptic equations I*, Inverse Problems, 27 (2011), p. 075008 (28pp).

[20] D. N. Hào and T. N. T. Quyen, *Finite element methods for coefficient identification in an elliptic equation*, Appl. Anal., 93 (2014), pp. 1533-1566.

[21] B. He, L.Z Liao, D. Han and H. Yang, *A new inexact alternating directions method for monotone variational inequalities*, Math. Program., 92 (2002), pp. 103–118.

[22] B. He and X. Yuan, *On the O(1/n) convergence rate of the Douglas–Rachford alternating direction method*, SIAM J. Numer. Anal., 50 (2012), pp. 700–709.

[23] M. R. Hestenes, *Multiplier and gradient methods*, J. Optim. Theory Appl., 4 (1969), pp. 303–320.

[24] M. Hinze, R. Pinnau, M. Ulbrich and S. Ulbrich, *Optimization with PDE constraints*. Springer Science & Business Media, 2008.

[25] M. Hinze and T. N. T. Quyen, *Matrix coefficient identification in an elliptic equation with the convex energy functional method*, Inverse Problems, 32 (2016), p. 085007 (29pp).

[26] K. Ito, M. Kroller, and K. Kunisch, *A numerical study of an augmented Lagrangian method for the estimation of parameters in elliptic systems*, SIAM J. Sci. and Stat. Comput., 12 (1991), pp. 884–910.

[27] K. Ito and K. Kunisch, *The augmented Lagrangian method for parameter estimation in elliptic systems*, SIAM J. Control Optim., 28 (1990), pp. 113–136.

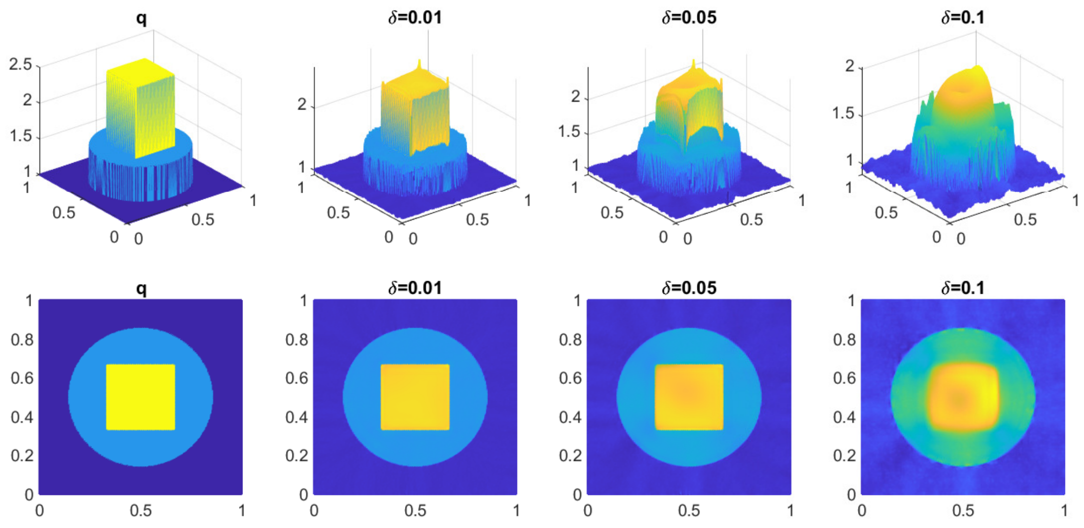

Figure 6: Numerical solutions q_h^k at the 30-th iteration for Example 2 with $h = 1/256$. Column 1: the true coefficient q; Column 2: $\delta = 0.01$ and the relative error $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)} = 0.0104$; Column 3: $\delta = 0.05$ and the relative error $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)} = 0.0544$; Column 4: $\delta = 0.1$ and the relative error $\|q_h^k - q\|_{L^2(\Omega)}/\|q\|_{L^2(\Omega)} = 0.1575$; Bottom: the projection of q or q_h^k onto the domain Ω.

[28] K. Ito and K. Kunisch, *Augmented Lagrangian-SQP-methods in Hilbert spaces and application to control in the coefficients problems*, SIAM J. Optim., 6 (1996), pp. 96–125.

[29] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in International Conference on Machine Learning, 2015, pp. 448–456.

[30] S. Ji, W. Xu, M. Yang, and K. Yu, *3D convolutional neural networks for human action recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2012), pp. 221–231.

[31] B. Kaltenbacher and J. Schberl, *A saddle point variational formulation for projection-regularized parameter identification*, Numer. Math., 91 (2002), pp. 675-697.

[32] Y. L. Keung and J. Zou, *Numerical identifications of parameters in parabolic systems*, Inverse Problems, 14 (1998), pp. 83–100.

[33] Y. L. Keung and J. Zou, *An efficient linear solver for nonlinear parameter identification problems*, SIAM J. Sci. Comput., 22 (2001), pp. 1511-1526.

[34] I. Knowles, *Parameter identification for elliptic problems*, J. Comput. Appl. Math., 131 (2001), pp. 175–194.

[35] R. Kohn and M. Vogelius, *Determining conductivity by boundary measurements*, Commun. Pure Appl. Math., 37 (1984), pp. 289–298.

[36] R. V. Kohn and M. Vogelius, *Relaxation of a variational method for impedance computed tomography*, Commun. Pure Appl. Math., 40 (1987), pp. 745–777.

[37] R. V. Kohn and B. D. Lowe, *A variational method for parameter identification*, ESAIM Math. Model. Numer. Anal., 22 (1988), pp. 119–158.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[39] K. Kunisch, and A. Rösch, *Primal-dual active set strategy for a general class of constrained optimal control problems*, SIAM J. Optim., 13 (2002), pp. 321–334.

[40] M. G. Larson and F. Bengzon, *The Finite Element Method: Theory, Implementation, and Applications*, Springer, Berlin, Heidelberg, 2013.

[41] S. LEFKIMMIATIS, *Universal denoising networks: a novel CNN architecture for image denoising*, in Proceedings of the IEEE conference on computer vision and pattern recognition, IEEE, 2018, pp. 3204–3213.

[42] M. J. D. POWELL, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., Academic Press, New York, 1969, pp. 283–298.

[43] T. N. T. QUYEN, *Variational method for multiple parameter identification in elliptic PDEs*, J. Math. Anal. Appl., 461 (2018), pp. 676-700.

[44] L. RUDIN, S. OSHER, AND E. FATEMI, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[45] K. SIMONYAN AND A. ZISSERMAN, *Very deep convolutional networks for large-scale image recognition*, in International Conference for Learning Representations, 2015.

[46] X.-C. TAI, J. FRØ YEN, M. S. ESPEDAL, AND T. F. CHAN, *Overlapping domain decomposition and multigrid methods for inverse problems*, in Domain decomposition methods, 10 (Boulder, CO, 1997), vol. 218 of Contemp. Math., Amer. Math. Soc., Providence, RI, 1998, pp. 523–529.

[47] G. WANG, G. T. WANG, Z. PAN AND Z. ZHANG, *Multiplicative noise removal using deep CNN denoiser prior*, in 2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), IEEE, 2017, pp. 1–6.

[48] J. Y. XIE, L. L. XU AND E. H. CHEN, *Image denoising and inpainting with deep neural networks*, Advances in neural information processing systems, 2012, pp. 341–349.

[49] K. ZHANG, W. ZUO, Y. CHEN, D. MENG, AND L. ZHANG, *Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising*, IEEE Trans. Image Process., 26 (2017), pp. 3142–3155.

[50] W. P. ZIEMER, *Weakly differentiable functions: Sobolev spaces and functions of bounded variation*, Springer-Verlag, New York, 1989.

[51] J. ZOU, *Numerical methods for elliptic inverse problems*, Int. J. Comput. Math., 70 (1998), pp. 211-232.