Journal of Inequalities and Applications
a SpringerOpen Journal

RESEARCH Open Access

Accurate and efficient numerical solutions for elliptic obstacle problems

Philku Lee[1*], Tai Wan Kim[2] and Seongjai Kim[3]

*Correspondence:
leepk@sogang.ac.kr
[1]Department of Mathematics, Sogang University, Ricci Building R1416, 35 Baekbeom-ro, Mapo-gu, Seoul, 04107, South Korea
Full list of author information is available at the end of the article

Abstract
Elliptic obstacle problems are formulated to find either superharmonic solutions or minimal surfaces that lie on or over the obstacles, by incorporating inequality constraints. In order to solve such problems effectively using finite difference (FD) methods, the article investigates simple iterative algorithms based on the successive over-relaxation (SOR) method. It introduces subgrid FD methods to reduce the accuracy deterioration occurring near the free boundary when the mesh grid does not match with the free boundary. For nonlinear obstacle problems, a method of gradient-weighting is introduced to solve the problem more conveniently and efficiently. The iterative algorithm is analyzed for convergence for both linear and nonlinear obstacle problems. An effective strategy is also suggested to find the optimal relaxation parameter. It has been numerically verified that the resulting obstacle SOR iteration with the optimal parameter converges about one order faster than state-of-the-art methods and the subgrid FD methods reduce numerical errors by one order of magnitude, for most cases. Various numerical examples are given to verify the claim.

Keywords: elliptic obstacle problem; successive over-relaxation (SOR) method; gradient-weighting method; obstacle relaxation; subgrid finite difference (FD)

1 Introduction

Variational inequalities have been extensively studied as one of key issues in calculus of variations and in the applied sciences. The basic prototype of such inequalities is represented by the so-called obstacle problem, in which a minimization problem is often solved. The obstacle problem is, for example, to find the equilibrium position u of an elastic membrane whose boundary is held fixed, with an added constraint that the membrane lies above a given obstacle φ in the interior of the domain $\Omega \subset \mathbb{R}^d$:

$$\min_u \int_\Omega \sqrt{1+|\nabla u|^2}\,d\mathbf{x}, \quad \text{s.t. } u \geq \varphi \text{ in } \Omega, u = f \text{ on } \Gamma, \tag{1.1}$$

where $\Gamma = \partial\Omega$ denotes the boundary of Ω and f is the fixed value of u on the boundary. The problem is deeply related to the study of minimal surfaces and the capacity of a set in potential theory as well. Other classical applications of the obstacle problem include the study of fluid filtration in porous media, constrained heating, elasto-plasticity, optimal control, financial mathematics, and surface reconstruction [1–7].

The problem in (1.1) can be linearized in the case of small perturbations by expanding the energy functional in terms of its Taylor series and taking the first term, in which case the energy to be minimized is the standard Dirichlet energy

$$\min_u \int_\Omega |\nabla u|^2 \, d\mathbf{x}, \quad \text{s.t. } u \geq \varphi \text{ in } \Omega, u = f \text{ on } \Gamma. \tag{1.2}$$

A variational argument [2] shows that, away from the contact set $\{\mathbf{x}|u(\mathbf{x}) = \varphi(\mathbf{x})\}$, the solution to the obstacle problem (1.2) is harmonic. A similar argument (which restricts itself to variations that are positive) shows that the solution is superharmonic on the contact set. Thus both arguments imply that the solution is a superharmonic function. As a matter of fact, it follows from an application of the maximum principle that the solution to the obstacle problem (1.2) is the least superharmonic function in the set of admissible functions. The Euler-Lagrange equation for (1.2) reads

$$\begin{array}{ll} \left.\begin{array}{l} -\Delta u \geq 0, \\ u \geq \varphi, \\ (-\Delta u) \cdot (u - \varphi) = 0, \end{array}\right\} & \text{in } \Omega, \\ u = f, & \text{on } \Gamma. \end{array} \tag{1.3}$$

In modern computational mathematics and engineering, the obstacle problems are not extremely difficult to solve numerically any more, as shown in numerous publications; see [8–14], for example. However, most of those known methods are either computationally expensive or yet to be improved for higher accuracy and efficiency of the numerical solution. In this article, we consider accuracy-efficiency issues and their remedies for the numerical solution of elliptic obstacle problems. This article makes the following contributions.

– *Accuracy improvement through subgrid finite differencing of the free boundary*: It can be verified either numerically or theoretically that the numerical solution easily involve a large error near the free boundary (the edges of obstacles), particularly when the grid mesh does not match with the obstacle edges. We suggest a post-processing algorithm which can reduce the error (by about a digit) by detecting accurate free boundary in subgrid level and introducing nonuniform *finite difference* (FD) method. The main goal of the subgrid FD algorithm is to produce a numerical solution of a higher accuracy u_h, which guarantees $u_h(\mathbf{x}) \geq \varphi(\mathbf{x})$ for *all* points $\mathbf{x} \in \Omega$.
– *Obstacle SOR*: The iterative algorithm for solving the linear system of the obstacle problem is implemented based on one of simplest iterative algorithms, the successive over-relaxation (SOR) method. Convergence of the obstacle SOR method is analyzed and compared with modern sophisticated methods. We also suggest an effective way to set the optimal relaxation parameter ω. Our simple obstacle SOR method with the optimal parameter performs better than state-of-the-art methods in both accuracy and efficiency.
– *Effective numerical methods for nonlinear problems*: For the nonlinear obstacle problem (1.1), a method of *gradient-weighting* is introduced to solve the problem more conveniently and efficiently. In particular, the suggested numerical schemes for the gradient-weighting problem produce an algebraic system of a symmetric and diagonally dominant M-matrix of which the main diagonal entries are all the same

positive constant. Thus the resulting system is easy to implement and presumably converges fast; as one can see from Section 5, the obstacle SOR algorithm for nonlinear problems converges in a similar number of iterations as for linear problems.

The article is organized as follows. The next section presents a brief review for state-of-the-art methods for elliptic obstacle problems focusing the one in [14]. Also, accuracy deterioration of the numerical solution (underestimation) is discussed by exemplifying an obstacle problem in 1D where the mesh grid does not match with edges of the free boundary. In Section 3, the SOR is applied for both linear and nonlinear problems and analyzed for convergence; the limits of iterates are proved to satisfy discrete obstacle problems. A method of gradient-weighting and second-order FD schemes are introduced for nonlinear problems. An effective strategy is suggested to find the optimal relaxation parameter. Section 4 introduces subgrid FD schemes near the free boundary in order to reduce accuracy deterioration of the numerical solution. In Section 5, various numerical examples are included to verify the claims we just made. Section 6 concludes the article summarizing our experiments and findings.

2 Preliminaries

As preliminaries, we first present a brief review for state-of-the-art methods for elliptic obstacle problems and certain accuracy issues related to the free boundary.

2.1 State-of-the-art methods for elliptic obstacle problems

This subsection summarizes state-of-the-art methods for elliptic obstacle problems focusing on the *primal-dual method incorporating L^1-like penalty term* (PDL1P) studied by Zosso *et al.* [14]. Primal-dual splitting methods have a great deal of attention, particularly in the context of total variation (TV) minimization and L^1-type problems in image processing [15–19].

In the literature of optimization problems, one of common practices is to reformulate a constrained optimization problem for a unconstrained problem by incorporating the constraint as a penalty term. Recently, Tran *et al.* [13] proposed the following minimization problem of a L^1-like penalty term:

$$\min_u \int_\Omega |\nabla u|^2 + \mu(\varphi - u)_+, \quad \text{s.t. } u|_\Gamma = f, \tag{2.1}$$

where μ is a Lagrange multiplier and $(\cdot)_+ = \max(\cdot, 0)$. It is shown that, for sufficiently large but finite μ, the minimizer of the unconstrained problem (2.1) is also the minimizer of the original, constrained problem (1.2).

The PDL1P [14] is a hybrid method which combines primal-dual splitting algorithm and the L^1-like penalty method in (2.1); it can be summarized as follows.

```
Initialize $u^0, \overline{u}^0, p^0 \leftarrow 0$.
Repeat
    (a) $p^{n+1} = (p^n + r_1 \nabla_h \overline{u}^n)/(1 + r_1)$,
    (b) $u^* = u^n + r_2 \nabla_h \cdot p^{n+1}$,                (PDL1P [14])        (2.2)
    (c) $u^{n+1} = \mathcal{P}_\varphi(u^*)$,
    (d) $\overline{u}^{n+1} = 2u^{n+1} - u^n$,
until $\|u^{n+1} - u^n\|_\infty < \varepsilon$,
```

where ∇_h denotes the numerical approximation of the gradient ∇, associated with the mesh size h, r_1 and r_2 are constants to be determined, p^n is the dual variable representing the gradient of the primal variable (u^n), and u^* is an intermediate solution. Here $\mathcal{P}_\varphi$ is an obstacle projection defined by

$$\mathcal{P}_\varphi\bigl(u^*\bigr)(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \in \Gamma, \\ u^*(\mathbf{x}) + r_2\mu & \text{if } \mathbf{x} \notin \Gamma \text{ and } u^*(\mathbf{x}) < \varphi(\mathbf{x}) - r_2\mu, \\ \varphi(\mathbf{x}) & \text{if } \mathbf{x} \notin \Gamma \text{ and } \varphi(\mathbf{x}) - r_2\mu \le u^*(\mathbf{x}) \le \varphi(\mathbf{x}), \\ u^*(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{2.3}$$

The above algorithm can be implemented effectively. It follows from (2.2)(a) that

$$\nabla_h \cdot p^{n+1} = \frac{\nabla_h \cdot p^n + r_1 \Delta_h \overline{u}^n}{1 + r_1}, \tag{2.4}$$

where Δ_h is the discrete Laplacian. Thus $S^{n+1} \equiv \nabla_h \cdot p^{n+1}$ can be considered as a variable and updated in each iteration, averaging its previous iterate S^n and $\Delta_h \overline{u}^n$ as in (2.4). As analyzed in [14], PDL1P (away from the obstacle) can be compared to either the forward Euler (explicit) scheme for discrete heat equation or a three-level time stepping method for a damped acoustic wave equation, where $r_1 r_2$ plays the role of the time-step size. PDL1P converges when

$$r_1 r_2 \|\Delta_h\| \le 1, \tag{2.5}$$

where $\|\Delta_h\|$ is the operator/induced norm of the discrete Laplacian Δ_h (= 8, when the mesh size $h = 1$). The authors in [14] claimed that '[Their] results achieve state-of-the-art precision in much shorter time; the speed up is one-two orders of magnitude with respect to the method in [13], and even larger compared to older methods [20–22].' Thus, in this article our suggested method would be compared mainly with PDL1P (the best-known method), in order to show its superiority.

2.2 Accuracy issues

The solution of obstacle problems must lie on or over the obstacle ($u \ge \varphi$), which is also one of requirements for numerical solutions. For FD methods and *finite element* (FE) methods for the obstacle problem (1.3), for example, this requirement can easily be violated when edges of the free boundary does not match with mesh grids. See Figure 1, where the shaded rectangle indicates the obstacle defined on one-dimensional (1D) interval $[x_0, x_5]$:

$$\varphi(x) = \begin{cases} 0 & \text{if } x_0 \le x < p, \\ 1 & \text{if } p \le x \le x_5, \end{cases} \tag{2.6}$$

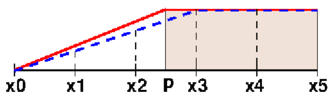

Figure 1 A non-matching grid: The true solution u (red solid curve) and the numerical solution on the non-matching grid u_h (blue dashed curve). Here the obstacle is the shaded region, $u(x_0) = u_h(x_0) = \varphi(x_0) = 0$, and $u(x_5) = u_h(x_5) = \varphi(x_5) = 1$.

which is not matching with the mesh grids $\{x_i : x_i = i \cdot h_x, i = 0, \ldots, 5\}$. The figure shows the true solution u (red solid curve) and a numerical solution u_h (blue dashed curve) of the linear obstacle problem (1.3) in 1D. The numerical solution is clearly underestimated and the magnitude of the error $|u_h - u|$ is maximized at $x = p$:

$$\max_x \bigl|u_h(x) - u(x)\bigr| = \bigl|u_h(p) - u(p)\bigr| = \frac{x_3 - p}{x_3 - x_0}, \tag{2.7}$$

which is $\mathcal{O}(h_x)$.

Let C_h denote the numerical contact set:

$$C_h \equiv \bigl\{x \in \Omega_h^0 : u_h(x) = \varphi(x)\bigr\}, \tag{2.8}$$

where Ω_h^0 is the set of *interior* grid points. Define an interior grid point is a *neighboring point* if it is not in the contact set but one of its adjacent grid points is in the contact set. Let the set of neighboring points be called the *neighboring set* N_h. Then, for the example in Figure 1, $C_h = \{x_3, x_4\}$ and $N_h = \{x_2\}$.

The accuracy of the numerical solution u_h can be improved by applying a post-processing in which a subgrid FD method is applied at grid points in the neighboring set. For example, at $x = x_2$, $-u_{xx}$ can be approximated by employing nonuniform FD schemes over the grid points $[x_1, x_2, p]$, given as

$$-u_{xx}(x_2) \approx \frac{2}{h_x^2}\left(-\frac{u_1}{1+r} + \frac{u_2}{r} - \frac{\varphi(p)}{r(1+r)}\right), \tag{2.9}$$

where $r = (p - x_2)/h_x \in (0, 1]$, and therefore numerical solution of $-u_{xx} = 0$ at $x = x_2$ must satisfy

$$u_2 = \frac{ru_1 + \varphi(p)}{1 + r}. \tag{2.10}$$

As r is approaching 0 (*i.e.*, $(p - x_2)$ becomes smaller proportionally), the obstacle value $\varphi(p)$ is more weighted. On the other hand, when $r = 1$, $\varphi(p) = u_3$ and the scheme in (2.9) becomes the standard second-order FD scheme. Let the numerical solution $\widetilde{u}$ be obtained from

$$\varphi(x_j) \le \widetilde{u}_j = \begin{cases} (r\widetilde{u}_{j-1} + \varphi(p))/(1+r) & \text{if } j = 2, \\ (\widetilde{u}_{j-1} + \widetilde{u}_{j+1})/2 & \text{if } j = 1, 3, 4, \end{cases} \tag{2.11}$$

where $\widetilde{u}_0 = 0$ and $\widetilde{u}_5 = 1$. Then it is not difficult to prove that $\widetilde{u}$ is *exactly* the same as the true solution u at all grid points (except numerical rounding error), regardless of the grid size h_x.

The above example has motivated the authors to develop an effective numerical algorithm for elliptic obstacle problems in 2D which detects the neighboring set of the free boundary, determines the subgrid proportions (r's), and updates the solution for an improved accuracy using subgrid FD schemes. Here the main goal is to try to guarantee $u(\mathbf{x}) \ge \varphi(\mathbf{x})$ for all $\mathbf{x} \in \Omega$ (whether $\mathbf{x}$ is a grid point or not). Since it is often the case that the free boundary is determined only after solving the problem, the algorithm must be a post-process. Details are presented in Section 4.

3 Obstacle relaxation methods

This section introduces and analyzes effective relaxation methods for solving (1.3) and its nonlinear problem as shown in (3.10) below.

3.1 The linear obstacle problem

We will begin with second-order approximation schemes for $-\Delta u$. For simplicity, we consider a rectangular domain in $\mathbb{R}^2$, $\Omega = (a_x, b_x) \times (a_y, b_y)$. Then the following second-order FD scheme can be formulated on the grid points:

$$\mathbf{x}_{pq} := (x_p, y_q), \quad p = 0, 1, \ldots, n_x, q = 0, 1, \ldots, n_y, \tag{3.1}$$

where, for some positive integers n_x and n_y,

$$x_p = a_x + p \cdot h_x, \qquad y_q = a_y + q \cdot h_y; \quad h_x = \frac{b_x - a_x}{n_x}, h_y = \frac{b_y - a_y}{n_y}.$$

Let $u_{pq} = u(x_p, y_q)$. Then, at each of the interior points $\mathbf{x}_{pq}$, the five-point FD approximation of $-\Delta u$ reads

$$-\Delta_h u_{pq} = \frac{-u_{p-1,q} + 2u_{pq} - u_{p+1,q}}{h_x^2} + \frac{-u_{p,q-1} + 2u_{pq} - u_{p,q+1}}{h_y^2}. \tag{3.2}$$

Multiply both sides of (3.2) by h_x^2 to have

$$(-\Delta_h u_{pq})h_x^2 = \left(2 + 2r_{xy}^2\right)u_{pq} - u_{p-1,q} - u_{p+1,q} - r_{xy}^2 u_{p,q-1} - r_{xy}^2 u_{p,q+1}, \tag{3.3}$$

where $r_{xy} = h_x/h_y$ and $u_{st} = f_{st}$ at boundary grid points (x_s, y_t).

Now, consider the following Jacobi iteration for simplicity. Given an initialization u^0, find u^n iteratively as follows.

Algorithm $\mathcal{L}_J$

```
For $n = 1, 2, \ldots$
  For $q = 1 : n_y - 1$
  For $p = 1 : n_x - 1$
    (a) $u_{J,pq} = \frac{1}{2+2r_{xy}^2}(u_{p-1,q}^{n-1} + u_{p+1,q}^{n-1} + r_{xy}^2 u_{p,q-1}^{n-1} + r_{xy}^2 u_{p,q+1}^{n-1})$;
    (b) $u_{pq}^n = \max(u_{J,pq}, \varphi_{pq})$;                                  (3.4)
  end
  end
end
```

where $u_{st}^{n-1} = f_{st}$ at boundary grid points (x_s, y_t).

Note that Algorithm $\mathcal{L}_J$produces a solution u of which the function value at a point is a simple average of four neighboring values, satisfying the constraint $u \geq \varphi$.

Theorem 1 *Let $\widehat{u}$ be the limit of the iterates u^n of Algorithm $\mathcal{L}_J$. Then $\widehat{u}$ satisfies the FD discretization of* (1.3). *That is,*

$$\left.\begin{aligned} &-\Delta_h\widehat{u}_{pq} \ge 0, \\ &\widehat{u}_{pq} \ge \varphi_{pq}, \\ &(-\Delta_h\widehat{u}_{pq}) \cdot (\widehat{u}_{pq} - \varphi_{pq}) = 0, \end{aligned}\right\} \quad (x_p, y_q) \in \Omega_h^0, \qquad \widehat{u}_{st} = f_{st}, \quad (x_s, y_t) \in \Gamma_h, \tag{3.5}$$

where Ω_h^0 denotes the set of interior grid points and Γ_h is the set of boundary grid points.

Proof It is clear to see from Algorithm $\mathcal{L}_J$that

$$\widehat{u}_{pq} \ge \varphi_{pq} \quad \text{for } (x_p, y_q) \in \Omega_h^0 \quad \text{and} \quad \widehat{u}_{st} = f_{st} \quad \text{for } (x_s, y_t) \in \Gamma_h.$$

Let $\widehat{u}_{pq} = \varphi_{pq}$ at an interior point (x_p, y_q). Then it follows from (3.4)(b) that

$$\widehat{u}_{J,pq} = \frac{1}{2 + 2r_{xy}^2}\big(\widehat{u}_{p-1,q} + \widehat{u}_{p+1,q} + r_{xy}^2\widehat{u}_{p,q-1} + r_{xy}^2\widehat{u}_{p,q+1}\big) \le \varphi_{pq} = \widehat{u}_{pq}, \tag{3.6}$$

which implies that

$$0 \le \big(2 + 2r_{xy}^2\big)\widehat{u}_{pq} - \widehat{u}_{p-1,q} - \widehat{u}_{p+1,q} - r_{xy}^2\widehat{u}_{p,q-1} - r_{xy}^2\widehat{u}_{p,q+1} = (-\Delta_h\widehat{u}_{pq}) \cdot h_x^2. \tag{3.7}$$

On the other hand, let $\widehat{u}_{pq} > \varphi_{pq}$ at (x_p, y_q). Then, since $\widehat{u}_{pq} = \max(\widehat{u}_{J,pq}, \varphi_{pq})$, we must have

$$\widehat{u}_{pq} = \widehat{u}_{J,pq}, \tag{3.8}$$

which implies that $-\Delta_h\widehat{u}_{pq} = 0$. This completes the proof. □

One can easily prove that the algebraic system obtained from (3.3) is irreducibly diagonally dominant and symmetric positive definite. Since its off-diagonal entries are all nonpositive, the matrix must be a Stieltjes matrix and therefore an M-matrix [23]. Thus relaxation methods of regular splittings (such as the Jacobi, the Gauss-Seidel (GS), and the successive over-relaxation (SOR) iterations) are all convergent and their limits are the same as $\widehat{u}$ and therefore satisfy (3.5). In this article, variants of Algorithm $\mathcal{L}_J$for the GS and the SOR would be denoted, respectively, by $\mathcal{L}_{\mathrm{GS}}$ and $\mathcal{L}_{\mathrm{SOR}}(\omega)$, where ω is an over-relaxation parameter for the SOR, $1 < \omega < 2$. For example, $\mathcal{L}_{\mathrm{SOR}}(\omega)$ is formulated as

Algorithm $\mathcal{L}_{\mathrm{SOR}}(\omega)$

```
For $n = 1, 2, \ldots$
  For $q = 1 : n_y - 1$
  For $p = 1 : n_x - 1$
    (a) $u_{\mathrm{GS},pq} = \frac{1}{2+2r_{xy}^2}(u_{p-1,q}^n + u_{p+1,q}^{n-1} + r_{xy}^2 u_{p,q-1}^n + r_{xy}^2 u_{p,q+1}^{n-1})$;
    (b) $u_{\mathrm{SOR},pq} = \omega \cdot u_{\mathrm{GS},pq} + (1-\omega) \cdot u_{pq}^{n-1}$;                (3.9)
    (c) $u_{pq}^n = \max(u_{\mathrm{SOR},pq}, \varphi_{pq})$;
  end
  end
end
```

where $u_{st}^{n-1} = u_{st}^n = f_{st}$ at boundary grid points (x_s, y_t).

Note that the right side of (3.9)(a) involves updated values wherever available. When $\omega = 1$, Algorithm $\mathcal{L}_{\text{SOR}}(\omega)$ becomes Algorithm $\mathcal{L}_{\text{GS}}$; that is, $\mathcal{L}_{\text{SOR}}(1) = \mathcal{L}_{\text{GS}}$.

3.2 The nonlinear obstacle problem

Applying the same arguments for the linear problem (1.3), the Euler-Lagrange equation for the nonlinear minimization problem (1.1) can be formulated as

$$\left.\begin{array}{l}\mathcal{N}(u) \geq 0,\\ u \geq \varphi,\\ \mathcal{N}(u)\cdot(u-\varphi) = 0,\end{array}\right\} \quad \text{in } \Omega, \qquad u = f, \quad \text{on } \Gamma, \tag{3.10}$$

where

$$\mathcal{N}(u) = -\nabla\cdot\left(\frac{\nabla u}{\sqrt{1+|\nabla u|^2}}\right). \tag{3.11}$$

Thus the solution to the nonlinear problem (3.10) can be considered as a minimal surface satisfying the constraint given by the obstacle function φ.

Since $\sqrt{1+|\nabla u|^2} \geq 1$, the nonlinear obstacle problem (3.10) can equivalently be formulated as

$$\left.\begin{array}{l}\mathcal{M}(u) \geq 0,\\ u \geq \varphi,\\ \mathcal{M}(u)\cdot(u-\varphi) = 0,\end{array}\right\} \quad \text{in } \Omega, \qquad u = f, \quad \text{on } \Gamma, \tag{3.12}$$

where

$$\mathcal{M}(u) = -\sqrt{1+|\nabla u|^2}\,\nabla\cdot\left(\frac{\nabla u}{\sqrt{1+|\nabla u|^2}}\right). \tag{3.13}$$

Such a method of gradient-weighting will make algebraic systems simpler and better conditioned, as to be seen below. In order to introduce effective FD schemes for $\mathcal{M}(u)$, we first rewrite $\mathcal{M}(u)$ as

$$\mathcal{M}(u) = -\left(\sqrt{1+|\nabla u|^2}\right)_1\left(\frac{u_x}{\sqrt{1+|\nabla u|^2}}\right)_x - \left(\sqrt{1+|\nabla u|^2}\right)_2\left(\frac{u_y}{\sqrt{1+|\nabla u|^2}}\right)_y, \tag{3.14}$$

where both $(\sqrt{1+|\nabla u|^2})_1$ and $(\sqrt{1+|\nabla u|^2})_2$ are the same as $\sqrt{1+|\nabla u|^2}$; however, they will be approximated in a slightly different way. The following numerical schemes are of second-order accuracy and specifically designed for the resulting algebraic system to be simpler and better conditioned.

For the FD scheme at the (p,q)th pixel, we first compute second-order FD approximations of $\sqrt{1+|\nabla u|^2}$ at $\mathbf{x}_{p-1/2,q}(W)$, $\mathbf{x}_{p+1/2,q}(E)$, $\mathbf{x}_{p,q-1/2}(S)$, and $\mathbf{x}_{p,q+1/2}(N)$:

$$\begin{aligned} d_{pq,W} &= \bigl[1 + (u_{pq} - u_{p-1,q})^2/h_x^2 \\ &\qquad + (u_{p-1,q+1} + u_{p,q+1} - u_{p-1,q-1} - u_{p,q-1})^2/\bigl(16h_y^2\bigr)\bigr]^{1/2}, \\ d_{pq,E} &= d_{p+1,q,W}, \\ d_{pq,S} &= \bigl[1 + (u_{pq} - u_{p,q-1})^2/h_y^2 \\ &\qquad + (u_{p+1,q} + u_{p+1,q-1} - u_{p-1,q} - u_{p-1,q-1})^2/\bigl(16h_x^2\bigr)\bigr]^{1/2}, \\ d_{pq,N} &= d_{p,q+1,S}. \end{aligned} \tag{3.15}$$

Then the directional-derivative terms at the pixel point $\mathbf{x}_{pq}$ can be approximated by

$$\begin{aligned} \left(\frac{u_x}{\sqrt{1+|\nabla u|^2}}\right)_x(\mathbf{x}_{pq}) &\approx \frac{1}{h_x^2}\left[\frac{1}{d_{pq,W}}u_{p-1,q} + \frac{1}{d_{pq,E}}u_{p+1,q} - \left(\frac{1}{d_{pq,W}} + \frac{1}{d_{pq,E}}\right)u_{pq}\right], \\ \left(\frac{u_y}{\sqrt{1+|\nabla u|^2}}\right)_y(\mathbf{x}_{pq}) &\approx \frac{1}{h_y^2}\left[\frac{1}{d_{pq,S}}u_{p,q-1} + \frac{1}{d_{pq,N}}u_{p,q+1} - \left(\frac{1}{d_{pq,S}} + \frac{1}{d_{pq,N}}\right)u_{pq}\right]. \end{aligned} \tag{3.16}$$

Now, we discretize the surface element as follows:

$$\begin{aligned} \bigl(\sqrt{1+|\nabla u|^2}\bigr)_1(\mathbf{x}_{pq}) &\approx \left[\frac{1}{2}\left(\frac{1}{d_{pq,W}} + \frac{1}{d_{pq,E}}\right)\right]^{-1} = \frac{2d_{pq,W}d_{pq,E}}{d_{pq,W} + d_{pq,E}}, \\ \bigl(\sqrt{1+|\nabla u|^2}\bigr)_2(\mathbf{x}_{pq}) &\approx \left[\frac{1}{2}\left(\frac{1}{d_{pq,S}} + \frac{1}{d_{pq,N}}\right)\right]^{-1} = \frac{2d_{pq,S}d_{pq,N}}{d_{pq,S} + d_{pq,N}}, \end{aligned} \tag{3.17}$$

where the right-hand sides are harmonic averages of FD approximations of $\sqrt{1+|\nabla u|^2}$ in x- and y-coordinate directions, respectively. Then it follows from (3.14), (3.16), and (3.17) that

$$\begin{aligned} \mathcal{M}(u)(\mathbf{x}_{pq}) \cdot h_x^2 \approx{}& \bigl(2 + 2r_{xy}^2\bigr)u_{pq} - a_{pq,W}u_{p-1,q} - a_{pq,E}u_{p+1,q} \\ & - r_{xy}^2 a_{pq,S}u_{p,q-1} - r_{xy}^2 a_{pq,N}u_{p,q+1}, \end{aligned} \tag{3.18}$$

where

$$\begin{aligned} a_{pq,W} &= \frac{2d_{pq,E}}{d_{pq,W} + d_{pq,E}}, \qquad a_{pq,E} = \frac{2d_{pq,W}}{d_{pq,W} + d_{pq,E}}, \\ a_{pq,S} &= \frac{2d_{pq,N}}{d_{pq,S} + d_{pq,N}}, \qquad a_{pq,N} = \frac{2d_{pq,S}}{d_{pq,S} + d_{pq,N}}. \end{aligned} \tag{3.19}$$

Note that $a_{pq,W} + a_{pq,E} = a_{pq,S} + a_{pq,N} = 2$. As for the linear problem, it is easy to prove that the algebraic system obtained from (3.18) is an M-matrix.

Given FD schemes for $\mathcal{M}(u)$ as in (3.18), the nonlinear obstacle problem (3.12) can be solved iteratively by the Jacobi iteration.

Algorithm $\mathcal{N}_J$

```
For $n = 1, 2, \ldots$
  For $q = 1 : n_y - 1$
  For $p = 1 : n_x - 1$
    (a) $u_{J,pq} = \frac{1}{2+2r_{xy}^2}(a_{pq,W}^{n-1}u_{p-1,q}^{n-1} + a_{pq,E}^{n-1}u_{p+1,q}^{n-1} + r_{xy}^2 a_{pq,S}^{n-1}u_{p,q-1}^{n-1} + r_{xy}^2 a_{pq,N}^{n-1}u_{p,q+1}^{n-1})$;        (3.20)
    (b) $u_{pq}^n = \max(u_{J,pq}, \varphi_{pq})$;
  end
  end
end
```

where $u_{st}^{n-1} = f_{st}$ at boundary grid points (x_s, y_t).

The superscript $(n-1)$ on the coefficients $a_{pq,D}$, $D = W, E, S, N$, indicate that they are obtained using the last iterate u^{n-1}. Algorithm $\mathcal{N}_J$ produces a solution u of which the function value at a point is a *weighted* average of four neighboring values, satisfying the constraint $u \geq \varphi$. One can prove the following corollary, using the same arguments introduced in the proof of Theorem 1.

Corollary 1 *Let $\widehat{u}$ be the limit of the iterates u^n of Algorithm $\mathcal{N}_J$. Then $\widehat{u}$ satisfies the FD discretization of* (3.12). *That is,*

$$\left.\begin{array}{l} \mathcal{M}_h(\widehat{u})_{pq} \geq 0, \\ \widehat{u}_{pq} \geq \varphi_{pq}, \\ \mathcal{M}_h(\widehat{u})_{pq} \cdot (\widehat{u}_{pq} - \varphi_{pq}) = 0, \end{array}\right\} \quad (x_p, y_q) \in \Omega_h^0, \\ \widehat{u}_{st} = f_{st}, \qquad (x_s, y_t) \in \Gamma_h, \tag{3.21}$$

where $\mathcal{M}_h(\widehat{u})_{pq}$ denotes the FD scheme of $\mathcal{M}(u)(\mathbf{x}_{pq})$ as defined in (3.18) *with $u = \widehat{u}$.*

Variants of Algorithm $\mathcal{N}_J$ for the GS and the SOR can be formulated similarly as for the linear obstacle problem; they would be denoted respectively by $\mathcal{N}_{\mathrm{GS}}$ and $\mathcal{N}_{\mathrm{SOR}}(\omega)$. In practice, such symmetric coercive optimization problems, the SOR methods are much more efficient than the Jacobi and Gauss-Seidel methods. We will exploit $\mathcal{L}_{\mathrm{SOR}}(\omega)$ and $\mathcal{N}_{\mathrm{SOR}}(\omega)$ for numerical comparisons with state-of-the-art methods, by setting the relaxation parameter ω optimal.

3.3 The optimal relaxation parameter $\widehat{\omega}$

Consider the standard Poisson equation with a Dirichlet boundary condition

$$\begin{aligned} -\Delta u &= g \quad \text{in } \Omega, \\ u &= f \quad \text{on } \Gamma = \partial\Omega, \end{aligned} \tag{3.22}$$

for prescribed functions f and g. Let $\Omega = [0,1]^2$, for simplicity, and apply the second-order FD method for the second derivatives on a uniform grid: $h = h_x = h_y = 1/(m+1)$, for some positive integer. The its algebraic system can be written as

$$A\mathbf{u} = \mathbf{b} \in \mathbb{R}^{m^2}. \tag{3.23}$$

Then the theoretically optimal relaxation parameter for the SOR method can be determined as [23], Section 4.3,

$$\widehat{\omega} = \frac{2}{1 + \sqrt{1 - \rho(T_J)^2}}, \tag{3.24}$$

where $\rho(T_J)$ is the spectral radius of the iteration matrix of the Jacobi method T_J. The iteration matrix T_J can be explicitly presented as a block tridiagonal matrix

$$T_J = \frac{1}{4}\,\mathrm{tridiag}(I_m, B_m, I_m), \tag{3.25}$$

where I_m is the m-dimensional identity matrix and

$$B = \mathrm{tridiag}(1,0,1) = \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{m\times m}.$$

For such a matrix T_J, it is well known that

$$\rho(T_J) = 1 - ch^2, \quad \text{for some } c > 0. \tag{3.26}$$

Thus it follows from (3.24) and (3.26) that the optimal SOR parameter corresponding to the mesh size h, $\widehat{\omega}_h$, can be expressed as

$$\widehat{\omega}_h = \frac{2}{1 + \sqrt{1 - (1 - ch^2)^2}} = \frac{2}{1 + \sqrt{2ch^2 - c^2h^4}} \approx \frac{2}{1 + c_0 h}, \tag{3.27}$$

where $c_0 = \sqrt{2c}$. Hence, for general mesh size h, the corresponding optimal SOR parameter $\widehat{\omega}_h$ can be found as follows.

$$\left\{\begin{array}{l} \text{(a) Determine } \widehat{\omega}_{h_0} \text{ for a prescribed mesh size } h = h_0, \textit{heuristically}. \\ \text{(b) Find } c_0 \text{ by solving (3.27) for } c_0\text{:} \\ \qquad c_0 = (2/\widehat{\omega}_{h_0} - 1)/h_0. \\ \text{(c) Use (3.27) with the above } c_0 \text{ to determine } \widehat{\omega}_h \text{ for general } h. \end{array}\right. \tag{3.28}$$

It is often the case that the calibration (3.28)(a)-(3.28)(b) can be carried out with a small problem, *i.e.*, with h_0 of a very low resolution.

4 Subgrid FD schemes for the free boundary: a post-process

This section describes subgrid FD schemes for the free boundary, focusing on the linear obstacle problem; the arguments to be presented can be applied the same way for nonlinear problems. Again, we assume for simplicity that $h = h_x = h_y$.

Let $\widehat{u}$ be the numerical solution of an obstacle problem. Then it would satisfy the discrete obstacle problem (3.5), particularly $\widehat{u}_{pq} \geq \varphi_{pq}$ at all (interior) grid points $\mathbf{x}_{pq} \in \Omega_h^0$.

However, when the mesh grid is not matching with the free boundary, the obstacle constraint $\widehat{u} \geq \varphi$ may not be satisfied at all points $\mathbf{x} \in \Omega$. This implies that when the mesh is not fine enough, the numerical solution can be underestimated near the free boundary, as shown in Figure 1 in Section 2.2. Note that the error introduced by non-matching grids is in $\mathcal{O}(h)$, while the numerical truncation error is in $\mathcal{O}(h^2)$ for second-order FD schemes. That is, the underestimation is in $\mathcal{O}(h)$, which can be much larger than the truncation error. The strategy below can be considered as a post-processing algorithm designed in order to reduce the underestimation without introducing a mesh refinement. The post-processing algorithm consists of three steps: (a) finding the numerical contact set and the neighboring set, (b) subgrid determination of the free boundary, and (c) nonuniform FD schemes on the neighboring set.

4.1 The contact set and the neighboring set

Finding the numerical contact set is an easy task. Let $\widehat{u}$ and φ be the numerical solution and the lower obstacle, respectively. Then, for example, the characteristic set of contact points C_h can be determined as follows.

$$\left[\begin{array}{l} C_h = \widehat{u} - \varphi; \\ \text{if } C_h(\mathbf{x}_{pq}) > 0, \quad \text{then } C_h(\mathbf{x}_{pq}) = 1; \quad \text{for all points } \mathbf{x}_{pq}; \\ C_h = \mathbf{1} - C_h. \end{array}\right. \tag{4.1}$$

As defined in Section 2.2, an interior grid point is a neighboring point when it is not in the contact set but one of its adjacent grid points is in the contact set. Thus the neighboring points can be found more effectively as follows. Visit each point in the contact set; if any one of its four adjacent points is not in the contact set, then the non-contacting point is a neighboring point. The set of all neighboring points is the neighboring set N_h.

4.2 Subgrid determination of the free boundary

Let $\mathbf{x}_{pq}$ be a neighboring point with two of its adjacent points are contact points ($C_h(p+1,q) = C_h(p,q-1) = 1$), as in Figure 2. Then we may assume that the *real* free boundary passes somewhere between the contact points and the neighboring points. We will suggest an effective strategy for the determination of the free boundary in subgrid level.

We first focus on the horizontal line segment connecting $\mathbf{x}_{pq}$ and $\mathbf{x}_{p+1,q}$ in the east (E) direction. Define

$$\mathbf{x}_E(r) = (1-r)\mathbf{x}_{pq} + r\mathbf{x}_{p+1,q}, \quad r \in [0,1]. \tag{4.2}$$

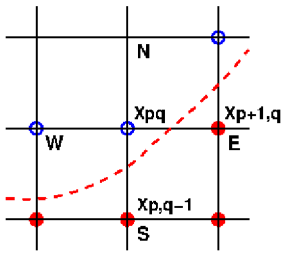

Figure 2 Contact points (red solid circle) and neighboring points (blue open circle). The red dashed curve indicates a possible free boundary.

Then the corresponding linear interpolation between $\mathbf{u}_{pq}$ and $\mathbf{u}_{p+1,q}$ over the line segment is formulated as

$$L_E(r) = (1-r)\mathbf{u}_{pq} + r\mathbf{u}_{p+1,q}, \quad r \in [0,1]. \tag{4.3}$$

Let

$$F_E(r) = \varphi\bigl(\mathbf{x}_E(r)\bigr) - L_E(r), \quad r \in [0,1]. \tag{4.4}$$

Since $\mathbf{x}_{pq}$ and $\mathbf{x}_{p+1,q}$ are a neighboring point and a contact point, respectively, we have

$$F_E(0) < 0 \quad \text{and} \quad F_E(1) = 0. \tag{4.5}$$

If the free boundary passes between $\mathbf{x}_{pq}$ and $\mathbf{x}_{p+1,q}$, then there must exist $r \in (0,1)$ such that $F_E(r) > 0$. Let r_E be such that $\mathbf{x}_E(r_E)$ represents the intersection between the line segment $\mathbf{x}_E(\cdot)$ and the free boundary. Then it can be approximated as follows.

$$r_E = \max_{r\in(0,1]} F_E(r). \tag{4.6}$$

The maximization problem in (4.6) can be solved easily (using the Newton method, for example), when the obstacle is defined as a smooth function. A more robust method can be formulated as a combination of a line search algorithm and the bisection method.

```
set $k_0$, $k_1$;
$r_E = 1$;    $F_{\max} = 0$;
for $k = 1 : k_0 - 1$    % line search
    if $F_E(k/k_0) > F_{\max}$
        $r_E = k/k_0$;    $F_{\max} = F_E(k/k_0)$;
    end
end
if $r_E < 1$    % refine it through bisection
    $r_b = 1/k_0$;
    for $k = 1 : k_1$
        $r_b = r_b/2$;
        if $F_E(r_E - r_b) > F_{\max}$
            $r_E = r_E - r_b$;    $F_{\max} = F_E(r_E - r_b)$;
        end
        if $F_E(r_E + r_b) > F_{\max}$
            $r_E = r_E + r_b$;    $F_{\max} = F_E(r_E + r_b)$;
        end
    end
    $B_E = \varphi(\mathbf{x}_E(r_E))$;
end
```
(4.7)

Remarks

- The last evaluation of φ (and saving) is necessary for the nonuniform FD schemes on the neighboring set, which will be discussed in Section 4.3. The quantity B_E will be used as the Dirichlet value on the free boundary.

- For other directions D (= W, S, or N), one can define corresponding difference functions F_D as shown in (4.2)-(4.4) for $D = E$. Then r_D can be obtained by applying (4.7) with F_E being replaced with F_D. When the adjacent point $\mathbf{x}_D(1)$ is not a contact point, you may simply set $r_D = 1$. Thus each neighboring point produces an array of four values $[r_W, r_E, r_S, r_N]$ and free boundary values for the directions D where $r_D < 1$.
- Assuming that (4.6) has a unique solution and the obstacle is given as a smooth function, the maximum error for the detection of the free boundary using (4.7) is

$$\left(\frac{1}{k_0}\cdot\frac{1}{2^{k_1}}\right)h, \tag{4.8}$$

where h is mesh size. It has been numerically verified that the choice $(k_0, k_1) = (10, 4)$ is enough for an accurate detection of the free boundary, for which the upper bound of the error becomes $h/160 = 0.00625h$.

4.3 Nonuniform FD schemes on the neighboring set

Let $\mathbf{x}_{pq} = (x_p, y_q)$ be a neighboring point. Then $r_{pq,D} \in (0,1]$ would be available for each $D \in \{W, E, S, N\}$; $B_{pq,D}$ is also available for $r_D < 1$. Thus the FD scheme for $-u_{xx}(\mathbf{x}_{pq})$ can be formulated over three points $\{(x_p - r_{pq,W}h_x, y_q), (x_p, y_q), (x_p + r_{pq,E}h_x, y_q)\}$ as follows.

$$-u_{xx}(\mathbf{x}_{pq}) \approx \frac{2}{h_x^2}\left(-\frac{u_{pq,W}}{r_{pq,W}(r_{pq,W}+r_{pq,E})} + \frac{u_{pq}}{r_{pq,W}\cdot r_{pq,E}} - \frac{u_{pq,E}}{r_{pq,E}(r_{pq,W}+r_{pq,E})}\right), \tag{4.9}$$

where

$$u_{pq,W} = \begin{cases} u_{p-1,q} & \text{if } r_{pq,W} = 1, \\ B_{pq,W} & \text{if } r_{pq,W} < 1, \end{cases} \qquad u_{pq,E} = \begin{cases} u_{p+1,q} & \text{if } r_{pq,E} = 1, \\ B_{pq,E} & \text{if } r_{pq,E} < 1. \end{cases}$$

Similarly, the FD scheme for $-u_{yy}(\mathbf{x}_{pq})$ can be formulated over three points in the y-direction $\{(x_p, y_q - r_{pp,S}h_y), (x_p, y_q), (x_p, y_q + r_{pp,N}h_y)\}$:

$$-u_{yy}(\mathbf{x}_{pq}) \approx \frac{2}{h_y^2}\left(-\frac{u_{pq,S}}{r_{pq,S}(r_{pq,S}+r_{pq,N})} + \frac{u_{pq}}{r_{pq,S}\cdot r_{pq,N}} - \frac{u_{pq,N}}{r_{pq,N}(r_{pq,S}+r_{pq,N})}\right), \tag{4.10}$$

where

$$u_{pq,S} = \begin{cases} u_{p,q-1} & \text{if } r_{pq,S} = 1, \\ B_{pq,S} & \text{if } r_{pq,S} < 1, \end{cases} \qquad u_{pq,N} = \begin{cases} u_{p,q+1} & \text{if } r_{pq,N} = 1, \\ B_{pq,N} & \text{if } r_{pq,N} < 1. \end{cases}$$

Thus, the post-processing algorithm of the obstacle SOR (3.9), $\mathcal{L}_{\mathrm{SOR}}(\omega)$, can be formulated by replacing the two terms in the right side of (3.2) with the right sides of (4.9) and (4.10), and computing $u_{\mathrm{GS},pq}$ in (3.9.a) correspondingly at all neighboring points.

5 Numerical experiments

In this section, we apply the obstacle SOR method and the post-processing schemes to various obstacles to verify their effectiveness and accuracy. We mainly concern 2-D obstacle problems of Dirichlet boundary conditions. The algorithms are implemented, for both one and double obstacles, in Matlab and carried out on a Desktop computer of an

Intel i5-3450S 2.80 GHz processor. The optimal relaxation parameter is calibrated with the lowest resolution to find a constant c_0 (3.28) and the constant is used for all other cases. For a comparison purpose, we implemented a state-of-the-art method, PDL1P [14], and its parameters (r_1 and r_2 in (2.2)) are found heuristically for cases where the parameters are not suggested in [14]. The iterations are stopped when the maximum difference of consecutive iterates becomes smaller than the tolerance ε:

$$\left\| u^n - u^{n-1} \right\|_\infty < \varepsilon, \tag{5.1}$$

where $\varepsilon = 10^{-6}$ mostly; Section 5.3 uses $\varepsilon = 10^{-7}$ for an accurate estimation of the error. For all examples, the numerical solution is initialized from φ (the lower obstacle) and the boundary condition f.

$$u^0(\mathbf{x}) = \begin{cases} \varphi(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega_h^0, \\ f(\mathbf{x}) & \text{if } \mathbf{x} \in \Gamma_h. \end{cases} \tag{5.2}$$

5.1 Linear obstacle problems

We first consider a non-smooth obstacle $\varphi_1 : \Omega \to \mathbb{R}$ with $\Omega = [0,1]^2$, defined by

$$\varphi_1(x,y) = \begin{cases} 5 & \text{if } |x-0.6| + |y-0.6| < 0.04, \\ 4.5 & \text{if } (x-0.6)^2 + (y-0.25)^2 < 0.001, \\ 4.5 & \text{if } y = 0.57 \text{ and } 0.075 < x < 0.13, \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

We solve the linear obstacle problem varying resolutions. The tolerance is set $\varepsilon = 10^{-6}$ hereafter except for examples in Section 5.3. Table 1 presents the number of iterations and CPU (the elapsed time, measured in second) for the linear problem of the non-smooth obstacle (5.3). One can see from the table that our suggested method requires less iterations and converges about one order faster in the computation time than the PDL1P, a state-of-the-art method. We have also implemented the primal-dual hybrid gradient (PDHD) algorithm in [15, 18, 19] for obstacle problems. The PDL1P turns out to be a simple adaptation of the PDHD and their performances are about the same, particularly when μ is set large. For the resolution 64 × 64, Figure 3 depicts the numerical solutions of the PDL1P and the obstacle SOR and their contour lines. For this example, both the PDL1P and the obstacle SOR resulted in almost identical solutions.

Table 1 The number of iterations and CPU for the linear problem of the non-smooth obstacle φ_1 (5.3)

($\varepsilon = 10^{-6}$)	PDL1P		Obstacle SOR	
Resolution	Iter	CPU	Iter	CPU
32 × 32	1,284	0.13	84	0.005
64 × 64	1,744	0.71	148	0.03
128 × 128	2,111	3.58	268	0.22
256 × 256	2,097	14.09	525	1.70

For PDL1P [14], set $\mu = 10^8$, $r_1 = 0.01$, and $r_2 = 12.5$.

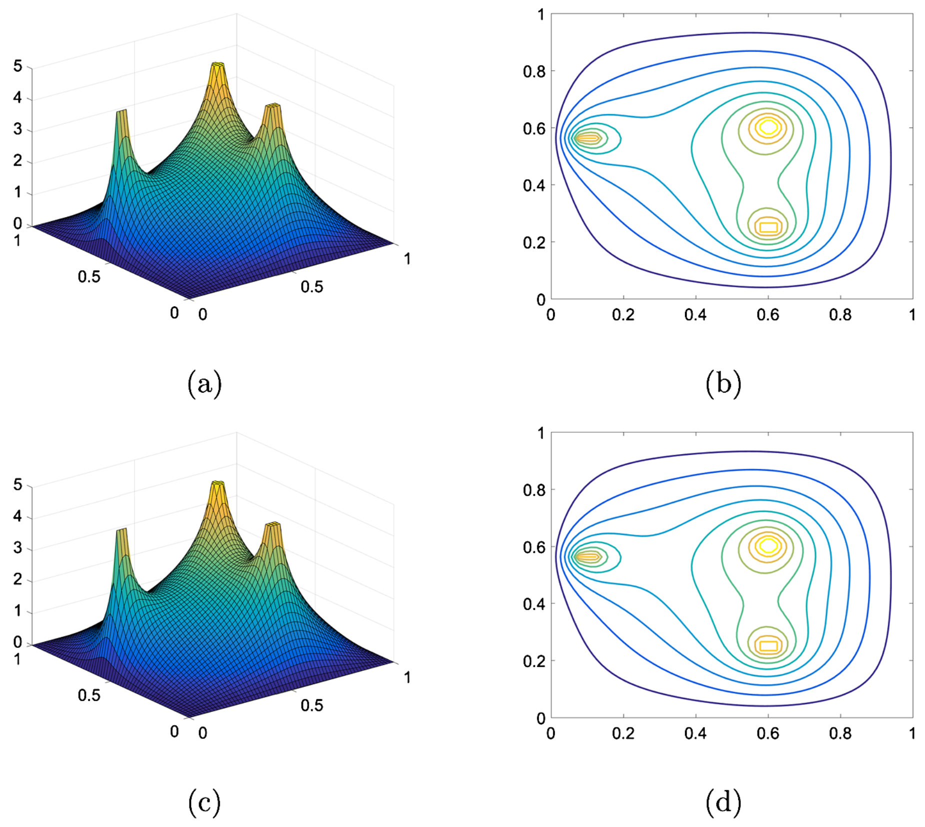

Figure 3 Solutions to the linear problem for the obstacle φ_1 (5.3) at resolution 64 × 64. (a) The numerical solution by the PDL1P, **(b)** its contour plot, **(c)** the numerical solution by the obstacle SOR, and **(d)** its contour plot.

As the second example, we consider the radially symmetric obstacle $\varphi_2 : \Omega \to \mathbb{R}$ with $\Omega = [-2,2]^2$ defined by

$$\varphi_2(r) = \begin{cases} \sqrt{1-r^2} & \text{if } r \le r^*, \\ -1 & \text{otherwise,} \end{cases} \tag{5.4}$$

where $r^* = 0.6979651482233\ldots$, the solution of

$$\left(r^*\right)^2\left(1 - \log\left(r^*/2\right)\right) = 1. \tag{5.5}$$

For the obstacle φ_2, the analytic solution to the linear obstacle problem can be defined as

$$u^*(r) = \begin{cases} \sqrt{1-r^2} & \text{if } r \le r^*, \\ -(r^*)^2 \ln(r/2)/\sqrt{1-(r^*)^2} & \text{otherwise,} \end{cases} \tag{5.6}$$

when the boundary condition is set appropriately using u^*. See Figure 4, in which we give plots of φ_2 and the true solution u^*.

In Table 2, we compare performances of the PDL1P and the obstacle SOR applied for the linear obstacle problem with (5.4). The PDL1P uses the parameters suggested in [14] ($\mu = 0.1$, $r_1 = 0.008$, $r_2 = 15.625$). As one can see from the table, our suggested method takes about one order less CPU time than the PDL1P for the computation of the numerical solution. In Figure 5, we show the numerical solutions u_h and the errors $u_h - u^*$ produced

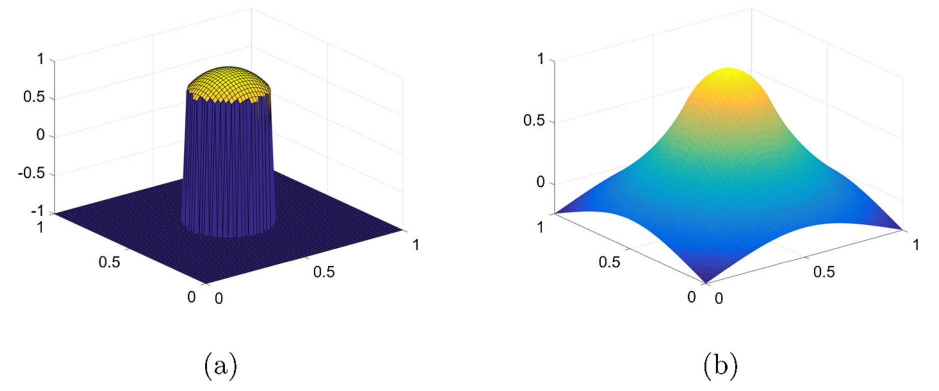

Figure 4 The true solution to the linear problem for the obstacle φ_2 (5.4) at resolution 64 × 64. (a) The obstacle φ_2 and **(b)** the true solution u^* (5.6)

Table 2 L^∞-errors, the number of iterations, and the CPU for linear obstacle problem with φ_2 (5.4)

($\varepsilon = 10^{-6}$)	PDL1P		Obstacle SOR	
Resolution	L^∞-error	Iter (CPU)	L^∞-error	Iter (CPU)
32 × 32	$8.91 \cdot 10^{-3}$	715 (0.09)	$8.69 \cdot 10^{-3}$	62 (0.02)
64 × 64	$3.01 \cdot 10^{-3}$	1,340 (0.60)	$3.05 \cdot 10^{-3}$	122 (0.04)
128 × 128	$7.66 \cdot 10^{-4}$	1,971 (3.51)	$7.64 \cdot 10^{-4}$	244 (0.22)
256 × 256	$1.86 \cdot 10^{-4}$	2,072 (14.85)	$1.88 \cdot 10^{-4}$	489 (1.63)

The PDL1P uses the parameters suggested in [14] ($\mu = 0.1$, $r_1 = 0.008$, $r_2 = 15.625$).

by the PDL1P and the obstacle SOR at the 64 × 64 resolution. The solutions are almost identical and the errors are nonpositive. This implies that the numerical solutions of the obstacle problem are underestimated.

As a more general obstacle problem, we consider the elastic-plastic torsion problem in [22]. The problem is to find the equilibrium position of the membrane between two obstacles φ, ψ that a force v is acting on:

$$\min_u \int_\Omega |\nabla u|^2 \, d\mathbf{x} - \int_\Omega uv \, d\mathbf{x}, \quad \text{s.t. } \psi \geq u \geq \varphi \text{ in } \Omega, u = f \text{ on } \Gamma. \tag{5.7}$$

Let $\Omega = [0,1]^2$ and the problem consist of two obstacles $\varphi_3 : \Omega \to \mathbb{R}$, $\psi_3 : \Omega \to \mathbb{R}$ and the force $v : \Omega \to \mathbb{R}$ defined by $\varphi_3(x,y) = -\operatorname{dist}(x, \partial\Omega)$, $\psi_3(x,y) = 0.2$ and

$$v(x,y) = \begin{cases} 300 & \text{if } (x,y) \in S = \{(x,y) : |x-y| \leq 0.1 \wedge x \leq 0.3\}, \\ -70e^y g(x) & \text{if } x \leq 1-y \text{ and } (x,y) \notin S, \\ 15e^y g(x) & \text{if } x > 1-y \text{ and } (x,y) \notin S, \end{cases} \tag{5.8}$$

where

$$g(x) = \begin{cases} 6x & \text{if } 0 \leq x \leq 1/6, \\ 2(1-3x) & \text{if } 1/6 < x \leq 1/3, \\ 6(x-1/3) & \text{if } 1/3 < x \leq 1/2, \\ 2(1-3(x-1/3)) & \text{if } 1/2 < x \leq 2/3, \\ 6(x-2/3) & \text{if } 2/3 < x \leq 5/6, \\ 2(1-3(x-2/3)) & \text{if } 5/6 < x \leq 1. \end{cases} \tag{5.9}$$

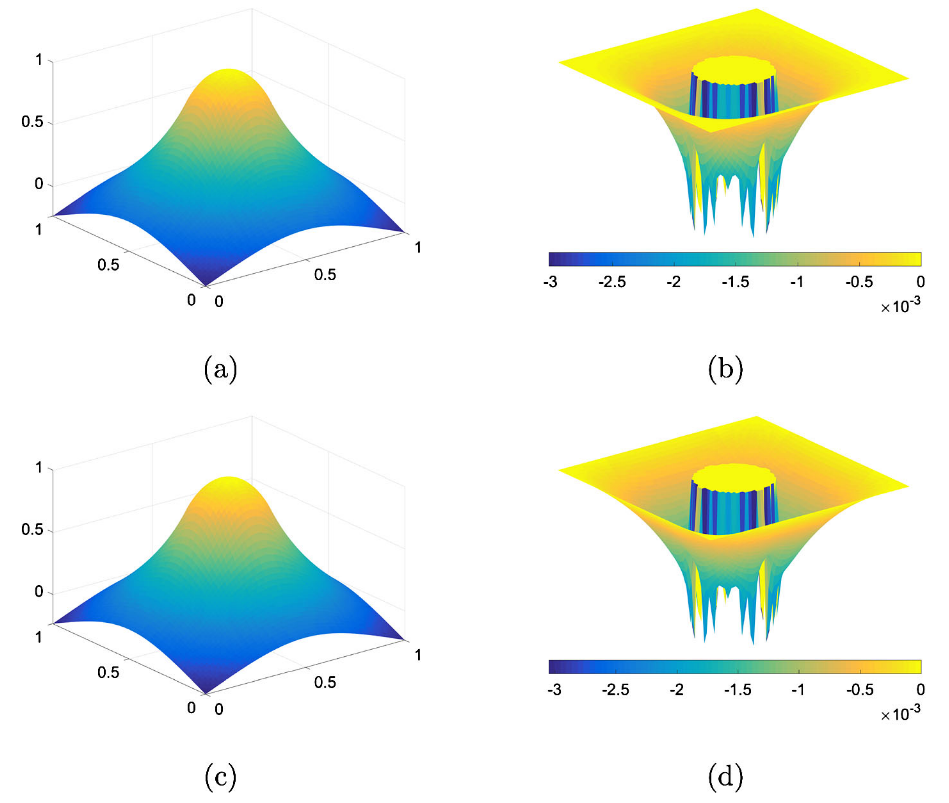

Figure 5 Numerical solutions u_h and errors $u_h - u^*$ at the 64 × 64 resolution. (a)-(b) by the PDL1P and **(c)-(d)** by the obstacle SOR.

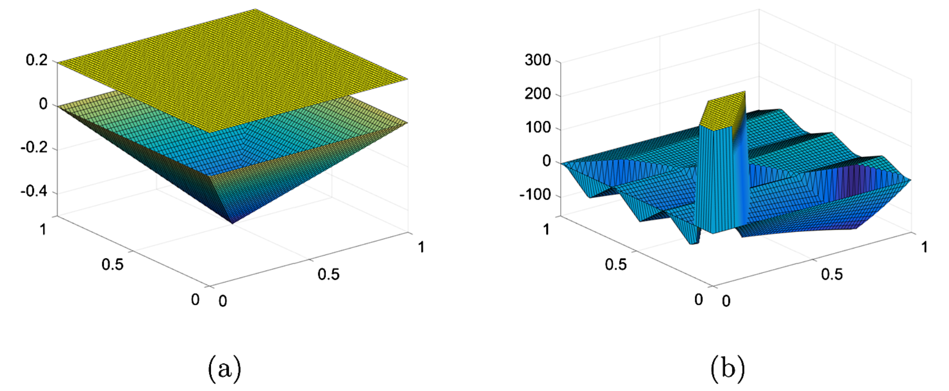

Figure 6 Elastic-plastic torsion problem. (a) The obstacles (ψ_3 and φ_3) and **(b)** the force v.

See Figure 6, where the obstacles and the force are depicted.

In Table 3, we present performances of the PDL1P and the obstacle SOR applied for the elastic-plastic torsion problem (5.7). For the PDL1P, we use the parameters suggested in [14] ($\mu = 0.1$, $r_1 = 0.008$, $r_2 = 15.625$). As one can see from the table, our suggested method again resulted in the numerical solution about one order faster than the PDL1P measured in the CPU time. In Figure 7, we illustrate the simulated membranes in the equilibrium satisfying (5.7) and their contact sets at resolution 64 × 64. In Figures 7(b)

Table 3 The number of iterations and the CPU time for the elastic-plastic torsion problem (5.7)

($\varepsilon = 10^{-6}$)	PDL1P		Obstacle SOR	
Resolution	Iter	CPU	Iter	CPU
32×32	887	0.13	47	0.02
64×64	1,287	0.68	98	0.04
128×128	1,609	3.43	193	0.22
256×256	1,866	17.27	368	1.58

For the PDL1P, we use the parameters suggested in [14] ($\mu = 0.1$, $r_1 = 0.008$, $r_2 = 15.625$).

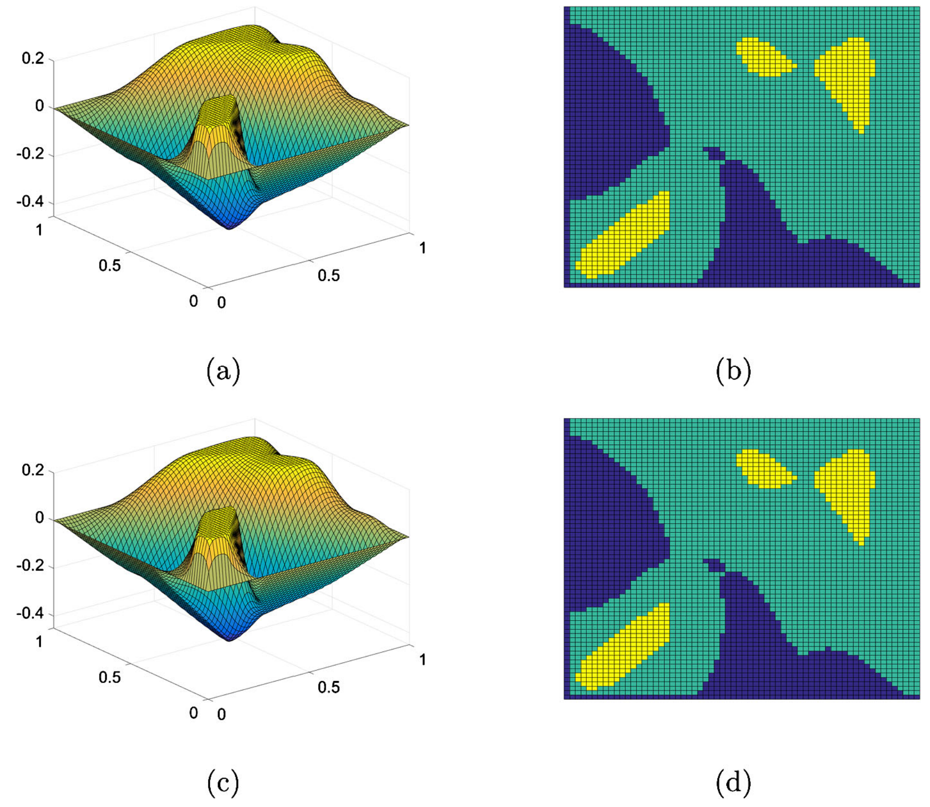

Figure 7 The numerical solutions and the contact sets for the elastic-plastic torsion problem at the 64×64 resolution. (a)-(b) by the PDL1P and **(c)-(d)** by the obstacle SOR.

and 7(d), the upper and lower contact sets are colored in yellow (brightest in gray scale) and blue (darkest in gray scale), respectively. The results produced by the two methods are *apparently* the same.

5.2 Nonlinear obstacle problems

The obstacle SOR is implemented for nonlinear obstacle problems as described in Section 3.2.

In Table 4, we present experiments for which the obstacle SOR is applied for nonlinear obstacle problems with $\varphi = \varphi_i$, $i = 1, 2, 3$. From a comparison with linear cases presented in Tables 1, 2, and 3, we can see for each of the obstacles that the obstacle SOR iteration for the nonlinear problem converges in a similar number of iterations as for the linear problem.

Table 4 The performance of the obstacle SOR applied for nonlinear obstacle problems with $\varphi = \varphi_i$, $i = 1, 2, 3$

($\varepsilon = 10^{-6}$)	φ_1		φ_2		φ_3	
Resolution	Iter	CPU	Iter	CPU	Iter	CPU
32 × 32	79	0.02	62	0.04	47	0.04
64 × 64	133	0.16	121	0.17	98	0.16
128 × 128	257	1.25	239	1.16	192	1.09
256 × 256	513	10.19	477	9.20	368	8.24

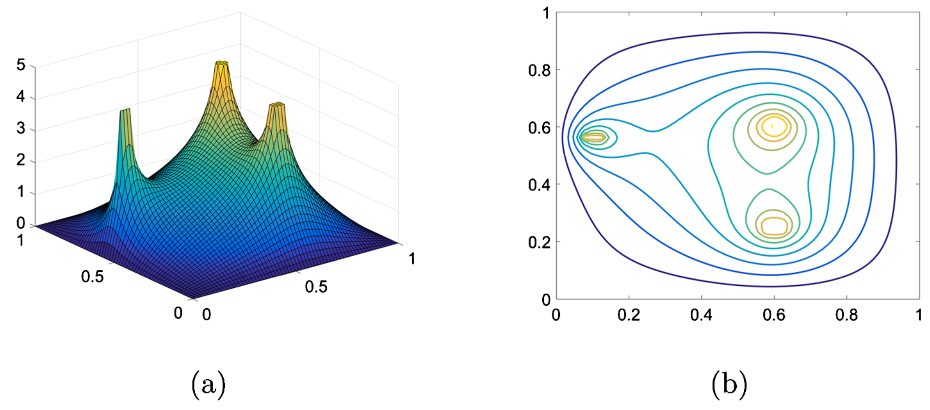

Figure 8 Nonlinear obstacle problem with $\varphi = \varphi_1$ at resolution 64 × 64. (a) The nonlinear numerical solution and **(b)** its contour plot.

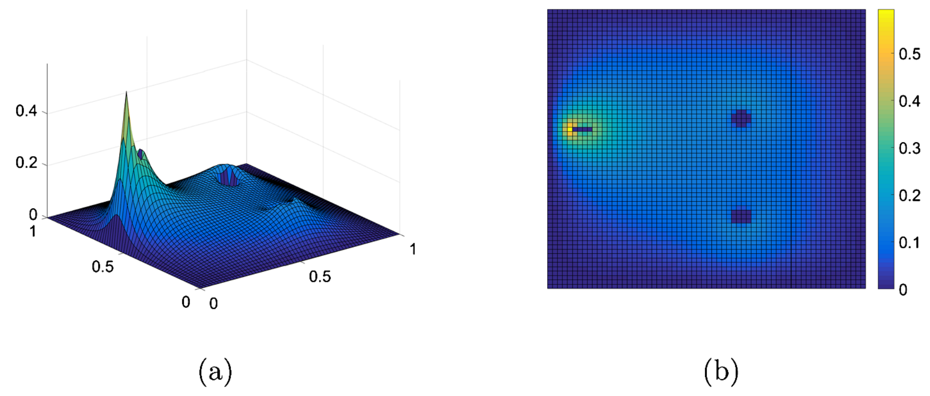

Figure 9 The difference between the linear solution and the nonlinear solution, at the 64 × 64 resolution, for the obstacle problem with $\varphi = \varphi_1$. (a) $(u_{h,\text{linear}} - u_{h,\text{nonlinear}})$ and **(b)** its density plot.

Only the apparent difference is the CPU time; an iteration of the nonlinear solver is about as six time expensive as that of the linear solver, due to the computation of coefficients as in (3.19). For $\varphi = \varphi_1$, the nonlinear solution is plotted in Figure 8. Compared with the linear solutions in Figure 3, the nonlinear solution shows slightly lower function values, which is expected. As the grid point approaches the obstacles, the solution shows an increasing gradient magnitude. This may enlarge weights for far-away grid values as shown in (3.19), which in return acts as a force to reduce function values. The difference between the linear solution and the nonlinear solution, at the 64 × 64 resolution, is depicted in Figure 9.

5.3 Post-processing algorithm

In Figure 5, one have seen that the error, the difference between the numerical solution and the analytic solution, shows its highest values near the free boundary. The larger error is due to the result of mismatch between the mesh grid and the obstacle edges. In order to eliminate the error effectively, we apply the subgrid FD schemes in Section 4 as a post-processing (PP) algorithm. For the examples presented in this subsection, the numerical solutions are solved as follows: (a) the problem is solved with $\varepsilon = 10^{-5}$ (pre-processing), (b) the free boundary is estimated with $(k_0, k_1) = (10, 4)$ and subgrid FD schemes are applied at neighboring grid points as in Section 4, and (c) another round of iterations is applied to satisfy the tolerance $\varepsilon = 10^{-7}$.

First, we consider a step function for an one-dimensional (1D) obstacle, as in Section 2.2. Let $\Omega = [0,1]$ and $\varphi_4 : \Omega \to \mathbb{R}$ defined by

$$\varphi_4(x) = \begin{cases} 0 & \text{if } 0 \le x < \pi/6, \\ 1 & \text{if } \pi/6 \le x \le 1. \end{cases} \tag{5.10}$$

The analytic solution to the linear problem is given as

$$u_{4,true}(x) = \begin{cases} 6x/\pi & \text{if } 0 \le x < \pi/6, \\ 1 & \text{if } \pi/6 \le x \le 1. \end{cases} \tag{5.11}$$

Figure 10 shows the numerical solutions to the linear problem associated to (5.10) with and without the post-process, and their errors. The numerical solutions without and with the post-process are obtained iteratively satisfying the tolerance $\varepsilon = 10^{-7}$. Notice that the solution without post-process is underestimated and shows a relatively high error: $\|u - u_{4,\text{true}}\|_\infty = 0.066$. The error is reduced to $\|u_{\text{pp}} - u_{4,\text{true}}\|_\infty = 8.57 \times 10^{-7}$ after the post-process.

The post-processing algorithm is applied to the linear obstacle problem in 2-D involving $\varphi = \varphi_2$. Table 5 contains efficiency results that compare performances of the PDL1P, the obstacle SOR (without post-process), and the obstacle SOR with the post-process

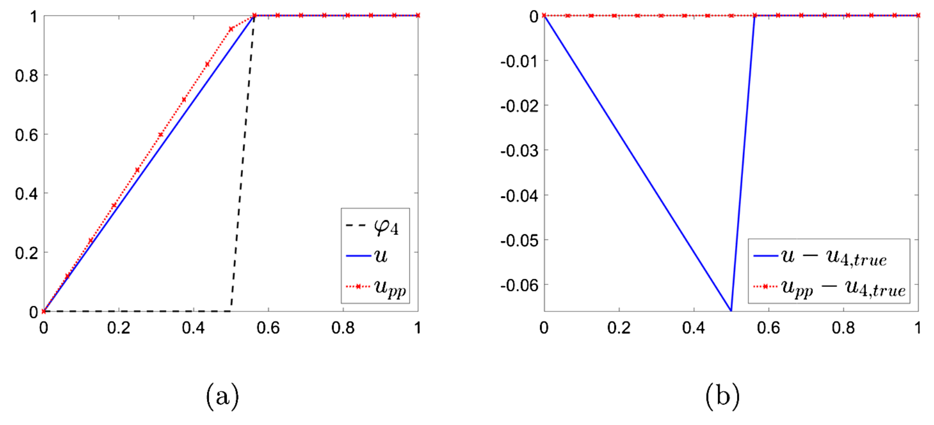

Figure 10 The post-processing algorithm applied for an obstacle problem in 1D at resolution 16. **(a)** The computed solutions without the post-process (blue solid curve) and with the post-process (red dotted curve with × marks) and **(b)** their errors. Here the subscript pp indicates the post-process.

Table 5 CPU time and iteration comparisons for the suggested post-process, applied to the linear problem with $\varphi = \varphi_2$

($\varepsilon = 10^{-7}$)	PDL1P		Obstacle SOR		Obstacle SOR+PP	
Resolution	Iter	CPU	Iter	CPU	Iter	CPU
25 × 25	814	0.07	56	0.02	92	0.07
50 × 50	1,486	0.40	108	0.03	147	0.10
100 × 100	2,092	2.26	213	0.13	299	0.26
200 × 200	2482	10.47	416	0.84	570	1.34

Table 6 L^∞ and L^2 error comparisons for the suggested post-process, applied to the linear problem with $\varphi = \varphi_2$

($\varepsilon = 10^{-7}$)	PDL1P		Obstacle SOR		Obstacle SOR+PP	
Resolution	L^∞ error	L^2 error	L^∞ error	L^2 error	L^∞ error	L^2 error
25 × 25	$1.94 \cdot 10^{-2}$	$4.35 \cdot 10^{-3}$	$1.94 \cdot 10^{-2}$	$4.38 \cdot 10^{-3}$	$8.44 \cdot 10^{-4}$	$2.80 \cdot 10^{-4}$
50 × 50	$4.38 \cdot 10^{-3}$	$8.10 \cdot 10^{-4}$	$4.39 \cdot 10^{-3}$	$8.41 \cdot 10^{-4}$	$2.01 \cdot 10^{-4}$	$6.93 \cdot 10^{-5}$
100 × 100	$1.25 \cdot 10^{-3}$	$2.73 \cdot 10^{-4}$	$1.25 \cdot 10^{-3}$	$2.87 \cdot 10^{-4}$	$5.16 \cdot 10^{-5}$	$1.79 \cdot 10^{-5}$
200 × 200	$5.45 \cdot 10^{-4}$	$7.29 \cdot 10^{-5}$	$5.46 \cdot 10^{-4}$	$7.76 \cdot 10^{-5}$	$1.40 \cdot 10^{-5}$	$4.74 \cdot 10^{-6}$

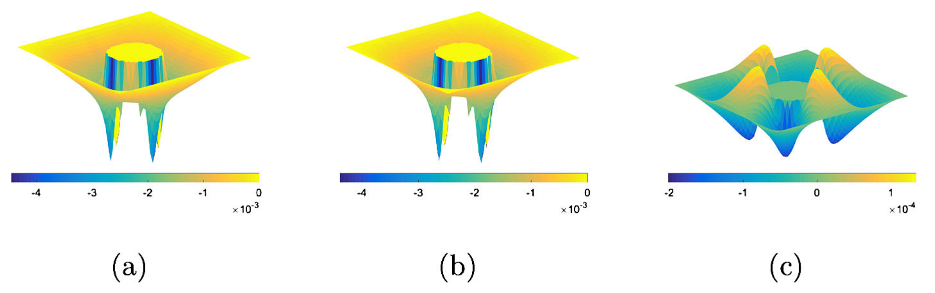

Figure 11 Plots of the error $(u_h - u^*)$ at the 50 × 50 resolution for the linear obstacle problem with $\varphi = \varphi_2$. (a) by the PDL1P, **(b)** by the obstacle SOR, and **(c)** by the obstacle SOR+PP.

(Obstacle SOR+PP) at various resolutions; while Table 6 presents an accuracy comparison for those methods. According to Table 5, the post-processed solution requires about 40% more iterations than the non-post-processed one; the incorporation of the post-process makes the iterative algorithm as twice expensive measured in CPU time as the original iteration. However, one can see from Table 6 that the post-process makes the error reduced by a factor of 15∼20. Thus in order to achieve a three-digit accuracy in the maximum-norm, for example, the PDL1P requires 10.47 seconds and the obstacle SOR completes the task in 0.84 seconds; when the obstacle SOR+PP takes only 0.1 second.

Figure 11 includes plots of the error $(u_h - u^*)$ at the 50 × 50 resolution for the linear obstacle problem with $\varphi = \varphi_2$, produced by the PDL1P, the obstacle SOR, and the obstacle SOR+PP. The numerical solutions of the PDL1P and the obstacle SOR are almost identical to each other and clearly underestimated, with the maximum discrepancy occurring around the free boundary due to the misfit between the mesh grid and the free boundary. It can be seen from Figure 11(c) that the post-process can eliminate the misfit error very effectively; the remaining error is the truncation error introduced by the second-order FD schemes.

5.4 Parameter choices

Finally, we present experimental results for parameter choices, when the obstacle SOR is applied for the linear problem with $\varphi = \varphi_3$. For an effective calibration of the optimal relaxation parameter as suggested in (3.28), we first select $h_0 = 1/25$. Then by using a trial-by-error method, we found the calibrated optimal relaxation parameter $\widehat{\omega}_{h_0} = 1.61$, which results in the following calibrated constant:

$$c_0 \approx 6.0559. \tag{5.12}$$

Thus it follows from (3.27) that the calibrated optimal relaxation parameter reads

$$\widehat{\omega}_{\mathrm{cal},h} \approx \begin{cases} 1.6817 & \text{when } h = 1/32, \\ 1.8371 & \text{when } h = 1/64, \\ 1.9097 & \text{when } h = 1/128, \\ 1.9538 & \text{when } h = 1/256, \end{cases} \tag{5.13}$$

which is used for the results of the obstacle SOR included in Table 3.

In order to verify effectiveness of the calibration, we implement a line search algorithm to find a relaxation parameter $\widehat{\omega}$ that converges in the smallest number of iterations with $\varepsilon = 10^{-6}$, the same tolerance as for the results in Table 3. For $h = 1/64$ and $h = 1/128$, the line search algorithm returned the curves as shown in Figure 12 with

$$[\widehat{\omega}, \mathrm{iter}] \approx \begin{cases} [1.6732, 45] & \text{when } h = 1/32, \\ [1.8217, 95] & \text{when } h = 1/64, \\ [1.9009, 189] & \text{when } h = 1/128. \end{cases} \tag{5.14}$$

Note that when the calibrated parameters are used, the iteration counts of the obstacle SOR presented in Table 3 are 47, 98, and 193, respectively, for $h = 1/32$, $h = 1/64$, and $h = 1/128$. Thus the calibrated optimal parameters in (5.13) are quite accurate for the optimal convergence.

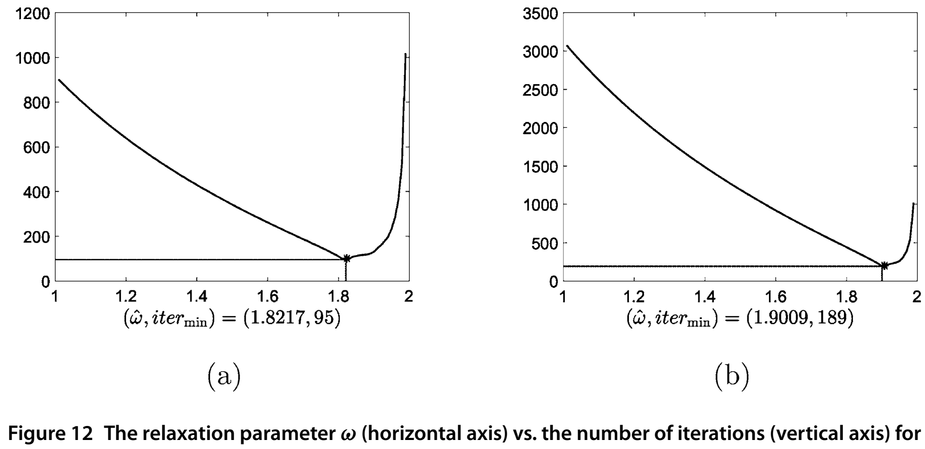

Figure 12 The relaxation parameter ω (horizontal axis) vs. the number of iterations (vertical axis) for solving the linear obstacle problem with $\varphi = \varphi_3$ by the obstacle SOR. (a) when $h = 1/64$ and **(b)** when $h = 1/128$.

6 Conclusions

Although various numerical algorithms have been suggested for solving elliptic obstacle problems effectively, most of the algorithms presented in the literature are yet to be improved for both accuracy and efficiency. In this article, the authors have studied obstacle relaxation methods in order to get second-order finite difference (FD) solutions of obstacle problems more accurately and more efficiently. The suggested iterative algorithm is based on one of the simplest relaxation methods, the successive over-relaxation (SOR). The iterative algorithm is incorporated with subgrid FD methods to reduce accuracy deterioration occurring near the free boundary when the mesh grid does not match with the free boundary. For nonlinear obstacle problems, a method of gradient-weighting has been introduced to solve the problem more conveniently and efficiently. The iterative algorithm has been analyzed for convergence for both linear and nonlinear obstacle problems. An effective strategy is also presented to find the optimal relaxation parameter. The resulting obstacle SOR has converged about one order faster than state-of-the-art methods and the subgrid FD methods could reduce the numerical errors by one order of magnitude.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
All authors have participated in the research and equally contributed to the writing of this manuscript. All authors read and approved the final manuscript.

Author details
[1]Department of Mathematics, Sogang University, Ricci Building R1416, 35 Baekbeom-ro, Mapo-gu, Seoul, 04107, South Korea. [2]Centennial Christian School International, 20 Shin Heung Ro 26-Gil, Yongsan Gu, Seoul, 140-833, South Korea. [3]Mississippi State University, Mississippi State, MS 39762-5921, USA.

Acknowledgements
S. Kim's work is supported in part by NSF grant DMS-1228337. Tai Wan (the second author) is a high school student who has spent many hours working on mathematical analysis and computational algorithms. S. Kim much appreciates his efforts for the project. Constructive comments by two anonymous reviewers improved the clarity of the paper and are much appreciated.

Received: 5 November 2016 Accepted: 25 January 2017 Published online: 03 February 2017

References
1. Bartels, S: The obstacle problem. In: Numerical Methods for Nonlinear Partial Differential Equations. Springer Series in Computational Mathematics, vol. 47, pp. 127-152. Springer, Berlin (2015)
2. Caffarelli, L: The obstacle problem revisited. J. Fourier Anal. Appl. **4**(4-5), 383-402 (1998)
3. Cha, Y, Lee, GY, Kim, S: Image zooming by curvature interpolation and iterative refinement. SIAM J. Imaging Sci. **7**(2), 1284-1308 (2014)
4. Kim, H, Cha, Y, Kim, S: Curvature interpolation method for image zooming. IEEE Trans. Image Process. **20**(7), 1895-1903 (2011)
5. Kim, H, Willers, J, Kim, S: Digital elevation modeling via curvature interpolation for LiDAR data. Electron. J. Differ. Equ. **23**, 47-57 (2016)
6. Petrosyan, A, Shahgholian, H, Uraltseva, N: Regularity of Free Boundaries in Obstacle-Type Problems. Graduate Studies in Mathematics. Am. Math. Soc., Providence (2012)
7. Rodrigues, JF: Obstacle Problems in Mathematical Physics. Notas de Matematica, vol. 114. Elsevier Science, Amsterdam (1987)
8. Arakelyan, A, Barkhudaryan, R, Poghosyan, M: Numerical solution of the two-phase obstacle problem by finite difference method. Armen. J. Math. **7**(2), 164-182 (2015)
9. Brugnano, L, Casulli, V: Iterative solution of piecewise linear systems. SIAM J. Sci. Comput. **30**(1), 463-472 (2008)
10. Brugnano, L, Sestini, A: Iterative solution of piecewise linear systems for the numerical solution of obstacle problems. arXiv:0912.3222 (2009)
11. Hoppe, RHW, Kornhuber, R: Adaptive multilevel methods for obstacle problems. SIAM J. Numer. Anal. **31**(2), 301-323 (1994)
12. Gräser, C, Kornhuber, R: Multigrid methods for obstacle problems. J. Comput. Math. **27**(1), 1-44 (2009)
13. Tran, G, Schaeffer, H, Feldman, WM, Osher, SJ: An L^1 penalty method for general obstacle problems. SIAM J. Appl. Math. **75**(4), 1424-1444 (2015)
14. Zosso, D, Osting, B, Xia, M, Osher, SJ: A fast primal-dual method for the obstacle problem. CAM Report 15-48, Department of Mathematics, Computational Applied Mathematics, University of California, Los Angeles, CA (2015, to appear in J. Sci. Comput.)

15. Chambolle, A, Pock, T: A first-order primal-dual algorithm for convex problems with applications to imaging. J. Math. Imaging Vis. **40**(1), 120-145 (2011)
16. Esser, E, Zhang, X, Chan, TF: A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. SIAM J. Imaging Sci. **3**(4), 1015-1046 (2010)
17. Sochen, N, Kimmel, R, Malladi, R: A general framework for low level vision. IEEE Trans. Image Process. **7**(3), 310-318 (1998)
18. Zhu, M, Chan, T: An efficient primal-dual hybrid gradient algorithm for total variation image restoration. CAM Report 08-34, Department of Mathematics, Computational Applied Mathematics, University of California, Los Angeles, CA (2008)
19. Zhu, M, Wright, SJ, Chan, TF: Duality-based algorithms for total-variation-regularized image restoration. Comput. Optim. Appl. **47**(3), 377-400 (2010)
20. Lions, PL, Mercier, B: Splitting algorithms for the sum of two nonlinear operators. SIAM J. Numer. Anal. **16**(6), 964-979 (1979)
21. Majava, K, Tai, XC: A level set method for solving free boundary problems associated with obstacles. Int. J. Numer. Anal. Model. **1**(2), 157-171 (2004)
22. Wang, F, Cheng, X: An algorithm for solving the double obstacle problems. Appl. Math. Comput. **201**(1-2), 221-228 (2008)
23. Varga, R: Matrix Iterative Analysis. Prentice-Hall, Englewood Cliffs (1962)