This paper presents a high-order deferred correction algorithm combined with penalty iteration for solving free and moving boundary problems, using a fourth-order finite difference method. Typically, when free boundary problems are solved on a fixed computational grid, the order of the solution is low due to the discontinuity in the solution at the free boundary, even if a high order method is used. Using a detailed error analysis, we observe that the order of convergence of the solution can be increased to fourth-order by solving successively corrected finite difference systems, where the corrections are derived from the previously computed lower order solutions. The penalty iterations converge quickly given a good initial guess. We demonstrate the accuracy and efficiency of our algorithm using several examples. Numerical results show that our algorithm gives up to fourth-order convergence for both the solution and the free boundary location. We also test our algorithm on challenging the American put option pricing problem. Our algorithm gives the expected high-order convergence.

A high-order deferred correction method for the solution of free boundary problems using penalty iteration, with an application to American option pricing

Dawei Wang[†], Kirill Serkh[‡ ◇], and Christina Christara[†]

May 4, 2022

[◇] This author's work was supported in part by the NSERC Discovery Grants RGPIN-2020-06022 and DGECR-2020-00356.

[†] Dept. of Computer Science, University of Toronto, Toronto, ON M5S 2E4
[‡] Dept. of Math. and Computer Science, University of Toronto, Toronto, ON M5S 2E4

1 Introduction

Free boundary problems, in which both the solution to a partial differential equation (PDE) and the domain on which it is defined are unknowns to be solved for, arise in numerous applications of practical importance. Many free boundary problems, both with boundaries that move over time (called moving boundary problems) and boundaries that are invariant with time, can be reformulated as linear complementarity problems (LCPs) (see, for example, §8.5 of [3]), a few well-known examples of which are the elliptic obstacle problem (see, for example, [20]) and the American option pricing problem (see, for example, [23]).

Two important categories for solving free boundary problems are the font tracking methods and the fixed domain methods. Front tracking methods directly compute an approximation to the free boundary, either at each time step in time-dependent problems, or iteratively in time-independent problems (as in, for example, [22]). While the free boundary can be tracked parametrically or as an indicator function of some set, the most common approaches are the level set method (see, for example, [21] for a survey), in which the free boundary is represented as the zero level-set of a function which obeys an evolution equation, and the phase-field method, in which the free boundary is approximated by a finite-width region where a phase-field function smoothly changes sign across the region (see, for example, [1] for a survey). The front-fixing method, in which the free boundary PDE is transformed into a nonlinear PDE with a fixed boundary, is also considered to be a front tracking method (see, for example, [26]). Whatever the particular method used, front tracking requires the construction of a separate algorithm for approximating the front, derived from the underlying equations and constraints of the free boundary problem.

In contrast, fixed domain methods reformulate the problem over the whole of a fixed domain, and solve the new equations in such a way that the position of the free boundary is returned simultaneously with the solution to the PDE, and appears a posteriori as part of the solution process. Such methods have a reputation for being robust and relatively straightforward to implement. The most widely-used fixed-domain method is the penalty method, which incorporates the inequality constraint of the LCP into the PDE by adding a nonlinear penalty term (see, for example, Ch. 1, §8 of [11]). The resulting penalized equation can be solved by successive over-relaxation (SOR), which can be fairly expensive (see, for example, [4]); this process can be accelerated by the multigrid method (see, for example, [2]). Remarkably, under certain conditions, when Newton's method is applied to the penalized PDE, the solution converges monotonically and exhibits the rapid convergence characteristic of Newton's method. However, while the discretized equation is solved to high accuracy, the approximation of the discrete solution to the true solution of the LCP is of low order, due to the disagreement between the free boundary and the fixed computational grid.

The problem of reconciling a nonconforming boundary with a fixed computational grid has been studied extensively, particularly in the context of front tracking methods for fluids and PDEs with smooth, fixed boundaries. One of the earliest such methods is the immersed boundary method (IBM) of Peskin, in which the boundary exerts an effect on a fluid, represented on a rectangular mesh, using approximations to delta functions located on the nonconforming smooth boundary (see [19]). This method was extended by Leveque and Li to the immersed interface method (IIM), which modifies the finite difference stencil in the vicinity of the boundary to correct for error terms derived from the underlying Taylor expansions (see [14]). In the explicit jump immersed interface method (EJIIM) proposed by Wiegmann and Bube, the corrections of the IIM are applied directly in terms of the jumps in the solution and its derivatives. Importantly, when the jumps are known a priori, the corrections are applied to the right hand side of the discretized system of equations; when they are unknown, they are simultaneously solved for and used to correct the solution in the same spirit as the IIM (see [24]). We note that the idea of applying corrections to the right hand side of the system was also suggested earlier by Fornberg and Meyer-Spasche in [9], in which they proposed a method for eliminating the first term in the expansion of the error near the nonconforming boundary. The ghost cell method (GSM) proposed by Gibou, Fedkiw, and others [13], based on the ghost fluid method (GFM) [7], is an alternative way of applying the jump corrections, in which ghost points are defined near the boundary, and equations for their values are adjoined to the discretized system. In [13], the authors observe that a second-order scheme can be constructed in which the discretized system is symmetric, however, they also observe that the resulting finite difference matrices becomes nonsymmetric for orders higher than two (see, for example, [12]). All of the aforementioned methods assume that the jumps at the nonconforming interface are either known beforehand, or are determined by augmenting the finite difference system with additional equations.

We present a method that does not augment or alter the finite difference matrix, and does not assume that the jumps are known in advance. We describe a high order deferred correction type algorithm for computing both the solution and the free boundary of an LCP. The idea is to derive the correction from the solution itself, after it has already been computed without any correction, or with a correction of a lower order. The correction is then applied to the right hand side, and the problem is re-solved to one order of accuracy higher than before. Two key ideas which we use to rigorously justify this procedure are the smoothness of the error away from the free boundary, which justifies the numerical differentiation and extrapolation of the solution to obtain the jumps, and the fact that the Green's function describing the error near the free boundary decreases like $O(h)$ as the gridsize h goes to zero, which is needed to show that the jump corrections are computed to a sufficiently high order. Since the corrections are computed separately and are applied exclusively to the right hand side, the matrix of the system to be solved is identical to the original finite difference matrix at each correction stage. In fact, since the solution at the previous correction stage can be used as an initial guess to penalty iteration at the subsequent stage, only one or two iterations are required for all correction stages after the first. The jump corrections are computed to high order by one-sided finite differences and extrapolation, and the location of the free boundary is determined, also to high order, from the solution by a combination of Lagrange interpolation and Newton's method. The deferred correction procedure can, at least in principle, be continued to indefinitely high orders, although we only apply it to fourth-order. We also note that the principles behind our deferred correction method are completely general, in the sense that they could be applied to essentially any free boundary problem formulated as an LCP. We demonstrate the effectiveness of the method on several examples with a one-dimensional space component, with and without a time component. We also apply our method to the well-known problem of the pricing of the American put option.

2 Preliminaries

2.1 The LCP formulation of free and moving boundary problems

One form of the variational inequality representation of a free boundary problem in one dimension is

$$\begin{cases} \partial_t \hat{V} - \mathcal{L}\hat{V} - g \geq 0, \\ \hat{V} - V^* \geq 0, \\ (\partial_t \hat{V} - \mathcal{L}\hat{V} - g) \cdot (\hat{V} - V^*) = 0, \end{cases} \tag{1}$$

see, for example [11], where $\mathcal{L}$ is a second-order differential operator

$$\mathcal{L} = p(t,S)\frac{\partial^2}{\partial S^2} + w(t,S)\frac{\partial}{\partial S} + z(t,S),$$

and V^* is a given function, sometimes called the obstacle function or the payoff function. Problem (1) is also called a linear complementarity problem. Note that all three relations in (1) need to be satisfied. The solution of the (1) is separated into two parts by a moving boundary $S_f(t)$. The goal is to find the solution $\hat{V} = \hat{V}(t,S)$ such that either $\hat{V} - V^* > 0$ and $\partial_t \hat{V} - \mathcal{L}\hat{V} - g = 0$, on what we call the PDE region of the solution, or $\partial_t \hat{V} - \mathcal{L}\hat{V} - g \geq 0$ and $\hat{V} - V^* = 0$, on what we call the penalty region of the solution.

In elliptic obstacle problems, the ∂_t term disappears in the above formulation, and $\mathcal{L}$ is an elliptic operator. In the American option pricing problems, $\partial_t - \mathcal{L}$ is the famous Black-Scholes operator and $\mathcal{L} = \mathcal{L}_{BS}$ with

$$\mathcal{L}_{BS} \equiv \frac{\sigma^2 S^2}{2}\partial_{SS} + (r-d)S\partial_S - r, \tag{2}$$

where S is the underlying asset price, r is the risk-free rate, d is the dividend rate of the underlying asset, σ is the volatility, and t is the backward time from expiry. Typical payoff functions are

$$V^*(S) = \max\{S-K, 0\} \;\text{ or } V^*(S) = \max\{K-S, 0\}$$

for the American call and put options, respectively. In the following discussion, we focus our analysis on moving boundary problems. When not explicitly stated, conclusions for free boundary problems, where the time derivative term disappears, follow identically.

Moreover, it can be shown that the solution is piecewise smooth, and the value matching and smooth pasting conditions

$$\hat{V}(t, S_f(t)) = V^*(S_f(t)), \quad \frac{\partial \hat{V}}{\partial S}(t, S_f(t)) = \frac{\partial V^*}{\partial S}(S_f(t)), \tag{3}$$

hold at the moving boundary (see, for example [25]). We see that the solution is only C^1 in space.

There has been much work that tries to achieve high-order convergence for solving the American option pricing problems. Nonuniform-mesh techniques have been proposed to deal with the nonsmoothness in the solution at the undertermined exercise boundary [18, 5]. In the work by Oosterlee and Leentvaar [18], the authors propose to use fourth-order finite differences in space and BDF4 in time, together with time-dependent grid-stretching in a predictor-corrector type scheme to achieve fourth-order accuracy. However, the authors of [18] did not provide numerical results on the convergence order for American options. In this paper, we develop a general deferred correction algorithm using fourth-order finite difference method in space and BDF4 in time for solving the free and moving boundary problems.

2.2 Penalty method for solving the LCP

In this paper, we solve the LCP using the penalty method as discussed in [10]. We approximate (1) by the penalized nonlinear PDE

$$\partial_t V = \mathcal{L}V + \rho \max\{V^* - V, 0\}, \tag{4}$$

for moving boundary problems, and

$$\mathcal{L}V + \rho \max\{V^* - V, 0\} = 0, \tag{5}$$

for free boundary problems, where ρ is a large positive penalty parameter, $\mathcal{L}$ is defined in (2), V is the solution, and V^* is the payoff function, as defined in Section 2.1, which also serves as the initial condition for PDE in (4). When $\rho \to \infty$, either $V - V^* \geq 0$ or $V^* = V + \epsilon$ for $0 < \epsilon \ll 1$, where $\epsilon = \mathcal{O}(\rho^{-1})$ and $\rho \max\{V^* - V, 0\}$ is bounded, see [11]. Using a finite volume discretization and applying the generalized Newton's iteration, also referred to as discrete penalty iteration, to the discretized PDE, [10] is able to prove monotonic convergence and finite termination of the algorithm under certain conditions. Moreover, second-order convergence can be obtained with an adaptive time step selector.

3 Discretization, error analysis and jump corrections

3.1 Discretization of the penalized equation

In this section, we describe an extension of the second-order penalty iteration method introduced in [10] for solving both (5) and (4). Unlike [10], we use fourth-order finite difference space discretization and BDF4 time stepping in order to obtain high-order accuracy. To approximate the second derivative, our fourth-order finite difference scheme uses five-point stencils for the interior grid points and six-point stencils for the near-boundary points. For the approximation of the first derivative, five-point stencils are used for both the interior and near-boundary points.

Consider a discretized domain $S_0 < S_1 < \cdots < S_{M+1}$ where S_0 and S_{M+1} represent the left and right boundary respectively. Let $\tilde{V}_j^n \approx V(t_n, S_j)$ be the finite difference approximation to the true solution $V(t, S)$ of (4) at time t_n, and space point S_j. We drop the superscript n when time is irrelevant. On a uniform grid with grid size h, the fourth-order finite difference approximation to $\frac{\partial^2 V}{\partial S^2}(t, S_j)$ is given by the operator

$$D_4^2 V_j \equiv \frac{1}{12h^2}(-V_{j-2} + 16V_{j-1} - 30V_j + 16V_{j+1} - V_{j+2}),$$

for $2 \leq j \leq M-1$, and

$$D_4^2 V_1 \equiv \frac{1}{12h^2}(10V_0 - 15V_1 - 4V_2 + 14V_3 - 6V_4 + 5V_5),$$

for $j = 1$, and similarly for $j = M$. On a nonuniform grid, the finite difference weights can be obtained by the method of undetertermined coefficients in a stable way (see, for example, [8]). We denote the generic fourth-order finite difference approximation to $\frac{\partial^2 V}{\partial S^2}$ at point S_j to be

$$D_4^2 V_j \equiv c_{-2}V_{j-2} + c_{-1}V_{j-1} + c_0 V_j + c_1 V_{j+1} + c_2 V_{j+2},$$

where we abuse notation here and denote the finite difference coefficients at the points $x_{j-2},\ x_{j-1},\ x_j,\ x_{j+1},\ x_{j+2}$ by $c_{-2},\ c_{-1},\ c_0,\ c_1,\ c_2$, respectively, for the finite difference approximation at x_j. Fourth-order finite difference discretization of the first derivative $\frac{\partial V}{\partial S}(t, S_j)$ can be obtained similarly using a five-point stencil, which we omit here.

Let $\mathbf{S}$ denote the vector the interior grid points, i.e. $\mathbf{S} = [S_1, \ldots, S_M]^T$. Assuming Dirichlet boundary conditions, the fourth-order finite differences above give us the space discretization of $\partial_{SS}V$ and $\partial_S V$

$$\frac{\partial V}{\partial S}(t, \mathbf{S}) \approx \bar{\mathbf{L}}_1 \tilde{\mathbf{V}}_{\text{aug}}, \quad \frac{\partial^2 V}{\partial S^2}(t, \mathbf{S}) \approx \bar{\mathbf{L}}_2 \tilde{\mathbf{V}}_{\text{aug}},$$

where $\tilde{\mathbf{V}}_{\text{aug}} = [\tilde{V}_0,\ \tilde{V}_1,\ \ldots,\ \tilde{V}_{M+1}]^T$ is the finite difference solution vector, $\bar{\mathbf{L}}_1$ and $\bar{\mathbf{L}}_2$ are $M \times (M+2)$ matrices with the coefficients of the corresponding finite difference stencil on each row. Let $\mathbf{L}$ be an $M \times M$ matrix defined by

$$\mathbf{L} = \mathbf{P}\mathbf{L}_2 + \mathbf{W}\mathbf{L}_1 + \mathbf{Z},$$

where $\mathbf{L}_2$ and $\mathbf{L}_1$ are $M \times M$ matrices from the interior columns of $\bar{\mathbf{L}}_2$ and $\bar{\mathbf{L}}_1$, respectively, and $\mathbf{P}$, $\mathbf{W}$, and $\mathbf{Z}$ are diagonal matrices with diagonal entries $[\mathbf{P}]_{jj} = p(t, S_j)$, $[\mathbf{W}]_{jj} = w(t, S_j)$, and $[\mathbf{Z}]_{jj} = z(t, S_j)$ for $j = 1, \ldots, M$. Then the discretization of $\mathcal{L}V$ becomes

$$\mathcal{L}V(t, \mathbf{S}) \approx \mathbf{L}\tilde{\mathbf{V}} + \mathbf{b},$$

where $\tilde{\mathbf{V}} = [\tilde{V}_1,\ \tilde{V}_2,\ \ldots,\ \tilde{V}_M]^T$, and

$$\begin{aligned}\mathbf{b} =& p(t, S_0)V(t, S_0)\bar{\mathbf{L}}_2[:, 1] + w(t, S_0)V(t, S_0)\bar{\mathbf{L}}_1[:, 1] \\ &+ p(t, S_{M+1})V(t, S_{M+1})\bar{\mathbf{L}}_2[:, M+2] + w(t, S_{M+1})V(t, S_{M+1})\bar{\mathbf{L}}_1[:, M+2],\end{aligned}$$

which is a vector that incorporates the boundary conditions, where $\bar{\mathbf{L}}_1[:, j]$ and $\bar{\mathbf{L}}_2[:, j]$ denote the j-th columns of $\bar{\mathbf{L}}_1$ and $\bar{\mathbf{L}}_2$, respectively. The penalty term in (5) can be discretized by

$$\mathbf{q}(\tilde{\mathbf{V}}) \equiv \rho \boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}}(\mathbf{V}^* - \tilde{\mathbf{V}}) \tag{6}$$

where $\mathbf{V}^* = [V_1^*,\ V_2^*,\ \ldots,\ V_M^*]^T$ is the vector of the payoff function values on the grid points S_1 to S_M, and $\boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}}$ is a diagonal matrix whose diagonal entries are

$$[\boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}}]_{i,i} = \begin{cases} 1, & V_i^* > \tilde{V}_i, \\ 0, & \text{else.} \end{cases} \tag{7}$$

Therefore, we obtain the discretization of the left-hand side of (5),

$$\mathcal{L}V(t,\mathbf{S}) + \rho \max\{V^*(\mathbf{S}) - V(t,\mathbf{S}), 0\} \approx \mathbf{L}\tilde{\mathbf{V}} + \mathbf{b} + \mathbf{q}(\tilde{\mathbf{V}}). \tag{8}$$

Assuming BDF4 uniform time discretization, and defining

$$\mathbf{A} \equiv \frac{25}{12}\mathbf{I} - k\mathbf{L},$$

the time stepping follows the rule

$$\mathbf{A}\tilde{\mathbf{V}}^{n+4} = 4\tilde{\mathbf{V}}^{n+3} - 3\tilde{\mathbf{V}}^{n+2} + \frac{4}{3}\tilde{\mathbf{V}}^{n+1} - \frac{1}{4}\tilde{\mathbf{V}}^{n} + k\mathbf{b}^{n+4} + k\mathbf{q}(\tilde{\mathbf{V}}^{n+4}), \tag{9}$$

where k is time step size, $\mathbf{I}$ is the identity matrix of size $M \times M$, and the superscript n means the n-th time step. The system (9) is a nonlinear system due to the presence of the penalty term, and we solve it using the penalty iteration described in [10].

We note that matrix $\mathbf{A}$ in (9) is no longer an M-matrix, which breaks the assumption in [10] that ensures monotone convergence. However, on examination of the proof in [10] shows that this requirement can be relaxed to the condition that the matrix $\mathbf{A}$ is monotone. Our numerical experiments show that this weaker condition is sufficient for the rapid convergence of penalty iteration, as is also observed in [10].

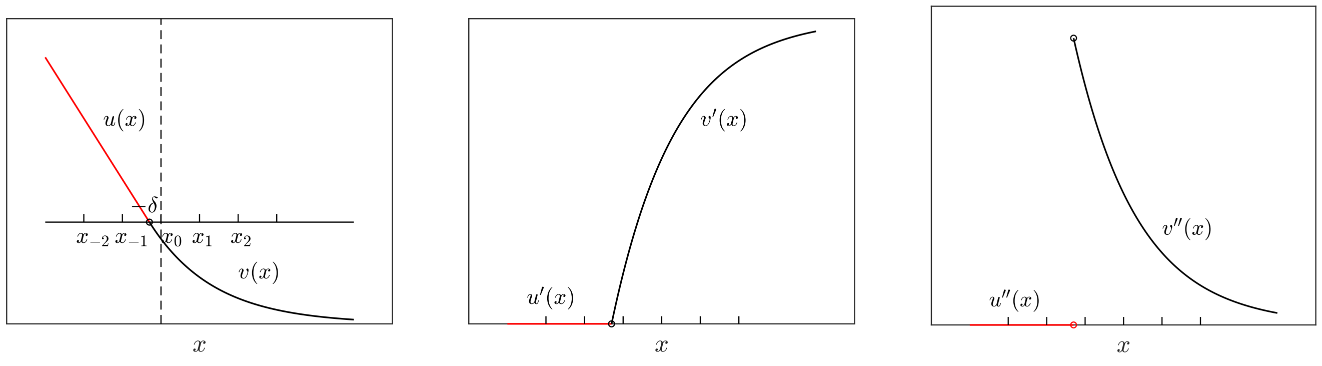
3.2 Finite difference approximation on a nonsmooth but piecewise smooth function

As has been shown in the previous section, the solution of the LCP has a discontinuous second derivative at the free boundary at all times. This is one of the major factors that causes the degeneracy of convergence rate when using the finite difference method on uniform grids. Since the finite difference approximation is based upon Taylor expansions, a certain level of smoothness has to be assumed in order to obtain the corresponding accuracy. When this smoothness requirement is not satisfied even at a single point, the truncation error will be contaminated by an additional error due to piecewise smoothness, and propagated to other points in the solution through the Green's function, as we will see later. The analysis of this section is similar to the analysis of Li [16], and Wiegmann and Bube [24], except that it is applied to our particular high-order finite difference operator.

To analyze the impact of piecewise smoothness on the finite difference approximation, consider a piecewise smooth function

$$f(x) = \begin{cases} v(x), & x + \delta > 0, \\ u(x), & x + \delta \le 0, \end{cases} \tag{10}$$

such that $u(-\delta) = v(-\delta)$ and $u'(-\delta) = v'(-\delta)$, where δ is a positive constant. In addition, suppose that $u(x)$ and $v(x)$ admit smooth extensions, i.e., $u(x)$ is well defined and can be smoothly extended to the domain $x > -\delta$, and similarly $v(x)$ can be smoothly extended to $x < -\delta$. Let $\{x_j\}$ be a grid with $x_i < x_j$ for $i < j$, and with $x_{-1} < -\delta < x_0 = 0$. An example graph of function $f(x)$ with grid points x_{-2} to x_2 is shown in Figure 1. We want to approximate the second derivative of $f(x)$ at grid points around the nonsmooth position $x = -\delta$.



(a) $f(x)$ (b) $f'(x)$ (c) $f''(x)$

Figure 1: An example graph of a nonsmooth function with a point of discontinuity of the second derivative at $x = -\delta$

3.2.1 *Second-order finite difference scheme*

For notational convenience, we give a detailed derivation only for the second-order method. Derivations for the fourth-order method follow similarly. We pick the grid and location of the nonsmooth point only for the ease of demonstration. The following derivation is generalizable to any other function of the same form, irrespectively of where the nonsmooth point is located. When using a second-order finite difference method to approximate the second derivative of $f(x)$ at point $x = x_0 = 0$, we are actually computing

$$D^2 f_0 = \frac{1}{\bar{h}_0}\left[\frac{1}{h_0}u_{-1} - \left(\frac{1}{h_0} + \frac{1}{h_1}\right)v_0 + \frac{1}{h_1}v_1\right], \tag{11}$$

where u_j, v_j denote $u(x_j), v(x_j)$ respectively, D^2 represents the standard centered three-point finite-difference operator, $h_j = x_j - x_{j-1}$, and $\bar{h}_j = (h_j + h_{j+1})/2$. Note that the value of $u(x_{-1})$ instead of $v(x_{-1})$ is used for the left-most stencil point in (11). This is because the finite difference operator is applied to $f(x)$, which is equal to $u(x_{-1})$ at point x_{-1}. However, the correct (in the sense that it is second-order accurate) approximation to the second derivative at the point x_0 should be

$$D^2 v_0 = \frac{1}{\bar{h}_0}\left[\frac{1}{h_0}v_{-1} - \left(\frac{1}{h_0} + \frac{1}{h_1}\right)v_0 + \frac{1}{h_1}v_1\right], \tag{12}$$

where we recall the assumption that $v(s)$ has smooth extension for $x < -\delta$. Note that u_{-1} in the formula $D^2 f_0$ is replaced by v_{-1} in the formula $D^2 v_0$. The other problematic point is at $x = x_{-1} = -h_0$, where we approximate the derivative by

$$D^2 f_{-1} = \frac{1}{\bar{h}_{-1}} \left[\frac{1}{h_{-1}} u_{-2} - \left(\frac{1}{h_{-1}} + \frac{1}{h_0} \right) u_{-1} + \frac{1}{h_0} v_0 \right],$$

rather than the second-order accurate finite difference

$$D^2 u_{-1} = \frac{1}{\bar{h}_{-1}} \left[\frac{1}{h_{-1}} u_{-2} - \left(\frac{1}{h_{-1}} + \frac{1}{h_0} \right) u_{-1} + \frac{1}{h_0} u_0 \right].$$

The points x_{-1} and x_0 are the only problematic points for a second-order method. The degeneracy of the finite difference approximation accuracy comes from the inconsistency of the formulas for $D^2 f_0$ and $D^2 v_0$, $D^2 f_{-1}$ and $D^2 v_{-1}$.

The following theorem describes the relationship between $D^2 f_0,\ D^2 f_{-1}$ and $D^2 v_0,\ D^2 u_{-1}$, respectively, in terms of the jumps of $u(x)$ and $v(x)$ at point $x = -\delta$, and quantifies the degeneration of accuracy.

Theorem 3.1. *Suppose $f(x)$ is given by (10), where $f(x) = v(x)$ for $x > -\delta$ and $f(x) = u(x)$ for $x \leq -\delta$, with $u(-\delta) = v(-\delta)$ and $u'(-\delta) = v'(-\delta)$, where $u(x)$ and $v(x)$ admit smooth extensions. Consider the functions on a grid $\{x_j\}$ with $x_i < x_j$ for $i < j$, and with $x_{-1} < -\delta < x_0 = 0$. Then, $D^2 f_0,\ D^2 f_{-1}$ and $D^2 v_0,\ D^2 u_{-1}$ satisfy the relations*

$$\begin{aligned} D^2 v_0 = D^2 f_0 - \frac{(h_0 - \delta)^2}{h_0(h_0 + h_1)}(u''_\delta - v''_\delta) + \frac{(h_0 - \delta)^3}{3h_0(h_0 + h_1)}(u'''_\delta - v'''_\delta) \\ - \frac{(h_0 - \delta)^4}{12h_0(h_0 + h_1)}(u''''_\delta - v''''_\delta) + \mathcal{O}(h^3), \end{aligned} \tag{13}$$

and

$$\begin{aligned} D^2 u_{-1} = D^2 f_{-1} + \frac{\delta^2}{h_0(h_{-1} + h_0)}(u''_\delta - v''_\delta) + \frac{\delta^3}{3h_0(h_{-1} + h_0)}(u'''_\delta - v'''_\delta) \\ + \frac{\delta^4}{12h_0(h_{-1} + h_0)}(u''''_\delta - v''''_\delta) + \mathcal{O}(h^3), \end{aligned} \tag{14}$$

where $h = \max\{h_0, h_1\}$, and the subscript δ denotes the quantities at the nonsmooth point $x = -\delta$, e.g. $u''_\delta = u''(-\delta)$.

Proof. Subtracting (11) from (12), we get

$$D^2 f_0 = D^2 v_0 + \frac{2}{h_0(h_0 + h_1)}(u_{-1} - v_{-1}). \tag{15}$$

Applying Taylor expansions for functions $u(x)$ and $v(x)$ around $x = -\delta$, we have

$$\begin{aligned} u_{-1} &= u_\delta - (h_0 - \delta)u'_\delta + \frac{(h_0 - \delta)^2}{2}u''_\delta - \frac{(h_0 - \delta)^3}{6}u'''_\delta + \frac{(h_0 - \delta)^4}{24}u''''_\delta + \mathcal{O}((h_0 - \delta)^5), \\ v_{-1} &= v_\delta - (h_0 - \delta)v'_\delta + \frac{(h_0 - \delta)^2}{2}v''_\delta - \frac{(h_0 - \delta)^3}{6}v'''_\delta + \frac{(h_0 - \delta)^4}{24}v''''_\delta + \mathcal{O}((h_0 - \delta)^5), \end{aligned}$$

which gives

$$u_{-1} - v_{-1} = \frac{(h_0-\delta)^2}{2}(u''_\delta - v''_\delta) - \frac{(h_0-\delta)^3}{6}(u'''_\delta - v'''_\delta) + \frac{(h_0-\delta)^4}{24}(u''''_\delta - v''''_\delta) + \mathcal{O}((h_0-\delta)^5), \tag{16}$$

using the assumptions that $u_\delta = v_\delta$ and $u'_\delta = v'_\delta$. Substituting (16) into (15), we get (13). Following a similar derivation, we get (14). □

3.2.2 *Fourth-order finite difference scheme*

In the previous section, we use the second-order approximation as a convenient way to demonstrate the essential relations that lead to our method. In this paper, we focus on high-order methods. Following exactly the same derivation procedure, we can arrive at similar formulas for fourth-order methods. The main difference between the second-order and fourth-order FDs is that in the fourth-order FDs there are four problematic points, namely x_{-2}, x_{-1}, x_0, x_1, instead of just two. Let the finite difference coefficients at the points $x_{j-2},\ x_{j-1},\ x_j,\ x_{j+1},\ x_{j+2}$ be denoted by $c_{-2},\ c_{-1},\ c_0,\ c_1,\ c_2$, respectively, for the finite difference approximation at x_j. We give the following theorem for fourth-order discretization.

Theorem 3.2. *Under the same assumptions as in Theorem 3.1, we have that $D_4^2u_{-2},\ D_4^2u_{-1},\ D_4^2v_0,\ D_4^2v_1$ satisfy the relations*

$$D_4^2u_{-2} = D_4^2f_{-2} + c_2\frac{\delta^2}{2}(u''_\delta - v''_\delta) + c_2\frac{\delta^3}{6}(u'''_\delta - v'''_\delta) + c_2\frac{\delta^4}{24}(u''''_\delta - v''''_\delta) + \mathcal{O}(h^3), \tag{17}$$

$$\begin{aligned} D_4^2u_{-1} = D_4^2f_{-1} &+ \left(c_1\frac{\delta^2}{2} + c_2\frac{(h_1+\delta)^2}{2}\right)(u''_\delta - v''_\delta) \\ &+ \left(c_1\frac{\delta^3}{6} + c_2\frac{(h_1+\delta)^3}{6}\right)(u'''_\delta - v'''_\delta) \\ &+ \left(c_1\frac{\delta^4}{24} + c_2\frac{(h_1+\delta)^4}{24}\right)(u''''_\delta - v''''_\delta) + \mathcal{O}(h^3), \end{aligned} \tag{18}$$

$$\begin{aligned} D_4^2v_0 = D_4^2f_0 &- \left(c_{-2}\frac{(h_{-1}+h_0-\delta)^2}{2} + c_{-1}\frac{(h_0-\delta)^2}{2}\right)(u''_\delta - v''_\delta) \\ &+ \left(c_{-2}\frac{(h_{-1}+h_0-\delta)^3}{6} + c_{-1}\frac{(h_0-\delta)^3}{6}\right)(u'''_\delta - v'''_\delta) \\ &- \left(c_{-2}\frac{(h_{-1}+h_0-\delta)^4}{24} + c_{-1}\frac{(h_0-\delta)^4}{24}\right)(u''''_\delta - v''''_\delta) + \mathcal{O}(h^3), \end{aligned} \tag{19}$$

$$\begin{aligned} D_4^2v_1 = D_4^2f_1 &- c_{-2}\frac{(h_0-\delta)^2}{2}(u''_\delta - v''_\delta) + c_{-2}\frac{(h_0-\delta)^3}{6}(u'''_\delta - v'''_\delta) \\ &- c_{-2}\frac{(h_0-\delta)^4}{24}(u''''_\delta - v''''_\delta) + \mathcal{O}(h^3), \end{aligned} \tag{20}$$

where $h = \max\{h_{-1}, h_0, h_1\}$.

Proof. From the approximation equations, we easily see that

$$\begin{aligned}
&D_4^2 f_{-2} = D_4^2 u_{-2} + c_2(v_0 - u_0),\\
&D_4^2 f_{-1} = D_4^2 u_{-1} + c_1(v_0 - u_0) + c_2(v_1 - u_1),\\
&D_4^2 f_0 = D_4^2 v_0 + c_{-2}(u_{-2} - v_{-2}) + c_{-1}(u_{-1} - v_{-1}),\\
&D_4^2 f_1 = D_4^2 v_1 + c_{-2}(u_{-1} - v_{-1}).
\end{aligned}$$

Then, expressing the quantities $u_{-2} - v_{-2},\ u_{-1} - v_{-1},\ v_0 - u_0,\ v_1 - u_1$, by applying Taylor expansions to $u_{-2}, v_{-2}, u_{-1}, v_{-1}, v_0, u_0, v_1, u_1$ about the point $x = -\delta$, exactly as in the proof of Theorem 3.1, we get the desired relations. □

From Theorems 3.1 and 3.2, we see that, as a result of the piecewise smoothness in the solution, the second derivative approximations have degenerated orders of accuracy, regardness of the order of discretization. Since $c_j = \mathcal{O}(1/h^2)$ and $\delta = \mathcal{O}(h)$, dominant $\mathcal{O}(1)$ terms appear in the truncation errors. In order to achieve the desired order of accuracy, we have to eliminate the remainder terms. This can be done by adding corrections. We call the right-hand side terms of Equations (17)–(20) that are added to $D_4^2 f_j$ the *correction terms* to the finite difference approximation of the second derivatives at x_{-2} to x_1. For example, we call $c_2 \frac{\delta^2}{2}(u''_\delta - v''_\delta) + c_2 \frac{\delta^3}{6}(u'''_\delta - v'''_\delta) + c_2 \frac{\delta^4}{24}(u''''_\delta - v''''_\delta)$ the correction terms to the FD $D_4^2 f_{-2}$ at x_{-2}.

3.2.3 *Modifying the finite differences with approximate corrections*

The order of the FDs at the problematic points x_{-2} to x_1 is determined by the dominant error term in the corrections. If we were able to apply the exact corrections using Equations (17)–(20), we would fully recover the fourth-order convergence of the FDs at the four problematic points x_{-2} to x_1. However, in this work, we assume that the exact free boundary location and the derivative jumps are unknown a priori. They are instead approximated using a previously computed $\mathcal{O}(h^\ell)$ solution, as we describe later. Therefore, we replace the exact free boundary and derivative jumps in the correction terms by the approximated ones. The accuracies of the corrected FDs depend on the accuracy of the approximate free boundary and derivative jumps.

To see how the order of accuracy of the free boundary and derivative jumps affect the correction, suppose that the free boundary is known exactly. Then, it is obvious that $\mathcal{O}(h^\ell)$ derivative jumps will give $\mathcal{O}(h^\ell)$ corrections. On the other hand, suppose that the derivative jumps are known exactly, but we are given an approximate free boundary equal to $\delta + \mathcal{O}(h^\ell)$ with $\ell \geq 1$. It is important to notice that the approximate free boundary introduces an extra source of error in the correction terms. To see this, we take one point, x_{-2}, for example. Defining $c_2' = c_2 h^2$, we have that $c_2' = \mathcal{O}(1)$. The finite difference scheme with approximate correction terms becomes

$$\begin{aligned}
&D_4^2 f_{-2} + c_2' \frac{(\delta + \mathcal{O}(h^\ell))^2}{2h^2}(u_\delta'' - v_\delta'') + c_2' \frac{(\delta + \mathcal{O}(h^\ell))^3}{6h^2}(u_\delta''' - v_\delta''') \\
&\qquad + c_2' \frac{(\delta + \mathcal{O}(h^\ell))^4}{24h^2}(u_\delta'''' - v_\delta'''') \\
=&D_4^2 f_{-2} + \left(c_2 \frac{\delta^2}{2} + \mathcal{O}(h^{\ell-1})\right)(u_\delta'' - v_\delta'') + \left(c_2 \frac{\delta^3}{6} + \mathcal{O}(h^\ell)\right)(u_\delta''' - v_\delta''') \\
&\qquad + \left(c_2 \frac{\delta^4}{24} + \mathcal{O}(h^{\ell+1})\right)(u_\delta'''' - v_\delta'''') \\
=&D_4^2 f_{-2} + c_2 \frac{\delta^2}{2}(u_\delta'' - v_\delta'') + c_2 \frac{\delta^3}{6}(u_\delta''' - v_\delta''') + c_2 \frac{\delta^4}{24}(u_\delta'''' - v_\delta'''') \\
&\qquad + \mathcal{O}(h^{\ell-1})(u_\delta'' - v_\delta'') + \mathcal{O}(h^\ell)(u_\delta''' - v_\delta''') + \mathcal{O}(h^{\ell+1})(u_\delta'''' - v_\delta'''') \\
=&D_4^2 u_{-2} + \mathcal{O}(h^{\ell-1})(u_\delta'' - v_\delta'') + \mathcal{O}(h^\ell)(u_\delta''' - v_\delta''') + \mathcal{O}(h^{\ell+1})(u_\delta'''' - v_\delta'''') + \mathcal{O}(h^3).
\end{aligned} \tag{21}$$

Equation (21) implies that, when applying corrections using an approximate free boundary, the correction terms produce additional errors that are one order lower than the accuracy of the approximate free boundary. In order to improve the order of accuracy of the finite difference scheme by adding back the correction terms, we see that ℓ has to satisfy $\ell \geq 2$, because if $\ell = 1$, the leading order term of the corrections on the right-and side of (21) is still of constant order $\mathcal{O}(h^{\ell-1}) = \mathcal{O}(1)$. Therefore, we require the approximate derivative jumps and the free boundary location to be of at least $\mathcal{O}(h)$ and $\mathcal{O}(h^2)$, respectively, in order to increase the order of accuracy of the corrected finite differences to first order, $\mathcal{O}(h^2)$ and $\mathcal{O}(h^3)$ to increase the order of accuracy to second-order, and so on.

3.3 Convergence of the fourth-order finite difference space discretization and its error propagation through the Green's function

3.3.1 Boundary value problems

In the previous section, we derived the correction terms for the finite difference approximations of derivatives of a piecewise smooth function. Unless the values of $u_\delta'' - v_\delta''$, $u_\delta''' - v_\delta'''$, $u_\delta'''' - v_\delta''''$ are known, we cannot make use of these formulas directly to obtain fourth-order convergence. To solve this problem, we use a deferred correction approach, and successively compute the approximate derivatives from the lower-order solutions that are already known, and make sure to match up the orders of solutions and orders of corrections. In order to decide how much accuracy is required for the derivative approximation to result in corrections of the required order, we need to first understand the error behaviour at each time step.

We consider boundary value problems that are time independent, i.e., we assume that $\partial_t \hat{V} = 0$ in the LCP given by (1) and solve

$$\begin{aligned}
&\mathcal{L}\hat{V} \geq 0, \\
&\hat{V} - V^* \geq 0, \\
&\mathcal{L}\hat{V} \cdot (\hat{V} - V^*) = 0,
\end{aligned} \tag{22}$$

so that we can leave the complexity of time evolution for later discussion. The free boundary S_f is defined to be the point such that

$$\begin{cases} \mathcal{L}\hat{V}(S) = 0,\ V(S) > V^*(S), & \text{when } S > S_f, \\ \mathcal{L}\hat{V}(S) \geq 0,\ V(S) = V^*(S), & \text{when } S \leq S_f. \end{cases}$$

We recall that the corresponding penalized equation becomes

$$\mathcal{L}V + \rho \max\{V^* - V, 0\} = 0. \tag{23}$$

To analyze the error behavior of the space discretization scheme, we consider the finite difference approximation of the PDE in (23),

$$\mathbf{L}\tilde{\mathbf{V}} + \mathbf{b} + \mathbf{q}(\tilde{\mathbf{V}}) = \mathbf{0}, \tag{24}$$

where $\mathbf{q}(\tilde{\mathbf{V}}) = \rho \boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}}(\mathbf{V}^* - \tilde{\mathbf{V}})$ is the discrete penalty term defined in (6).

The theorem below describes the error behaviour of the fourth-order finite difference scheme applied to (23), and how the nonsmoothness at the free boundary causes the convergence order of the fourth-order difference scheme to degenerate.

Proposition 3.1. *Consider the penalized PDE (23) and let $V(S)$ be its exact solution. In addition, let $\hat{V}(S)$ be the exact solution to the original LCP in (22). Suppose that the first $m+1$ points $\hat{V}(S_0)$, $\hat{V}(S_1)$, …, $\hat{V}(S_m)$ lie on the penalty region, i.e. $\hat{V}(S_j) = V^*(S_j) = V(S_j) + \epsilon$ for $0 \leq j \leq m$, with $0 < \epsilon \ll 1$ being approximately the size of the stopping tolerance set in the penalty iteration, and $\hat{V}(S_j) > V^*(S_j)$, $V(S_j) + \epsilon > V^*(S_j)$, for $m+1 \leq j \leq M+1$. Assume also that the approximate solution $\tilde{\mathbf{V}}$ of the penalty iteration exactly recovers $\boldsymbol{\mathcal{I}}_{\hat{\mathbf{V}}}$, i.e., $\boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}} = \boldsymbol{\mathcal{I}}_{\hat{\mathbf{V}}}$. Then, the error, $\mathbf{e} = [\hat{V}(S_1) - \tilde{V}_1,\ \hat{V}(S_2) - \tilde{V}_2,\ \ldots,\ \hat{V}(S_M) - \tilde{V}_M]^T$, of the fourth-order finite difference scheme in (24) for solving the penalized PDE (23) is the solution to*

$$(\mathbf{L} - \rho\boldsymbol{\mathcal{I}}_{\mathbf{V}})\mathbf{e} = \sum_{j=m-1}^{m+2} \mathcal{O}(1)\mathbf{1}_j - \sum_{j=1}^{m} \gamma_j \mathbf{1}_j + \sum_{j=1}^{M} \mathcal{O}(h^4)\mathbf{1}_j \equiv \mathbf{r}, \tag{25}$$

when the grid point S_m is not exactly on the free boundary, i.e. $S_m < S_f$, where $\mathbf{1}_j$ is the j-th column of an $M \times M$ identity matrix and $\gamma_j = \mathcal{L}\hat{V}(S_j)$, for $j = 1,\ \ldots,\ m$. When S_m is exactly on the free boundary, i.e., $S_m = S_f$, the sum in the first summation term is taken from $j = m-1$ to $m+1$.

Proof. Since $S_0, S_1, \ldots, S_m$ lie on the penalty region and we assume Dirichlet boundary conditions, $\boldsymbol{\mathcal{I}}_{\hat{\mathbf{V}}}$ is an $M \times M$ diagonal matrix with the diagonal elements $(\boldsymbol{\mathcal{I}}_{\hat{\mathbf{V}}})_{i,i} = 1$ for $i = 1, \ldots, m$ and $(\boldsymbol{\mathcal{I}}_{\hat{\mathbf{V}}})_{i,i} = 0$ for $i = m+1, \ldots, M$. Hence, from the assumption that $\boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}} = \boldsymbol{\mathcal{I}}_{\hat{\mathbf{V}}}$, we have

$$\begin{aligned} \mathbf{q}(\tilde{\mathbf{V}}) &= \rho[V^*(S_1) - \tilde{V}_1,\ \ldots,\ V^*(S_m) - \tilde{V}_m,\ 0,\ \ldots, 0]^T \\ &= \rho[\hat{V}(S_1) - \tilde{V}_1,\ \ldots,\ \hat{V}(S_m) - \tilde{V}_m,\ 0,\ \ldots, 0]^T \\ &= \rho[e_1,\ \ldots, e_m,\ 0,\ \ldots, 0]^T \\ &= \rho\boldsymbol{\mathcal{I}}_{\hat{\mathbf{V}}}\mathbf{e}, \end{aligned} \tag{26}$$

Moreover, from Theorem 3.2 for fourth-order discretization, we apply the discrete $\mathbf{L}$ operator to the true solution $\hat{\mathbf{V}} = \hat{V}(\mathbf{S})$ to get

$$\begin{aligned}\mathbf{L}\hat{\mathbf{V}} + \mathbf{b} &= \sum_{j=1}^{M} \left(\mathcal{L}\hat{V}(S_j) + \mathcal{O}(h^4)\right) \mathbf{1}_j + \sum_{j=m-1}^{m+2} \mathcal{O}(1)\mathbf{1}_j \equiv \mathcal{L}\hat{V}(\mathbf{S}) + \boldsymbol{\theta}_2,\\ \boldsymbol{\theta}_2 &\equiv \sum_{j=m-1}^{m+2} \mathcal{O}(1)\mathbf{1}_j + \sum_{j=1}^{M} \mathcal{O}(h^4)\mathbf{1}_j,\end{aligned} \tag{27}$$

when the grid point S_m is not exactly on the free boundary, i.e. $S_m < S_f$, where $\mathbf{1}_j$ is the j-th column of an $M \times M$ identity matrix, and

$$\begin{aligned}\mathbf{L}\hat{\mathbf{V}} + \mathbf{b} &= \sum_{j=1}^{M} \left(\mathcal{L}\hat{V}(S_j) + \mathcal{O}(h^4)\right) \mathbf{1}_j + \sum_{j=m-1}^{m+1} \mathcal{O}(1)\mathbf{1}_j \equiv \mathcal{L}\hat{V}(\mathbf{S}) + \boldsymbol{\theta}_1,\\ \boldsymbol{\theta}_1 &\equiv \sum_{j=m-1}^{m+1} \mathcal{O}(1)\mathbf{1}_j + \sum_{j=1}^{M} \mathcal{O}(h^4)\mathbf{1}_j,\end{aligned} \tag{28}$$

when S_m is exactly on the free boundary, i.e., $S_m = S_f$. Without loss of generality, we assume $S_m < S_f$ in the following. The proof for the case when $S_m = S_f$ is similar.

Since $\mathcal{L}\hat{V} > 0$ for $S_1 \le S \le S_m$, and $\mathcal{L}\hat{V} = 0$ for $S_{m+1} \le S \le S_M$, we have

$$\mathcal{L}\hat{V}(\mathbf{S}) = \begin{bmatrix} \mathcal{L}\hat{V}(S_1) \\ \mathcal{L}\hat{V}(S_2) \\ \vdots \\ \mathcal{L}\hat{V}(S_m) \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \mathcal{L}V^*(S_1) \\ \mathcal{L}V^*(S_2) \\ \vdots \\ \mathcal{L}V^*(S_m) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \equiv \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_m \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \boldsymbol{\gamma} \tag{29}$$

Subtracting (29) from (24) and applying (28), we get

$$\mathbf{L}(\hat{\mathbf{V}} - \tilde{\mathbf{V}}) - \mathbf{q}(\tilde{\mathbf{V}}) - \boldsymbol{\theta}_2 + \boldsymbol{\gamma} = (\mathbf{L} - \rho\boldsymbol{\mathcal{I}}_{\hat{\mathbf{V}}})\mathbf{e} - \boldsymbol{\theta}_2 + \boldsymbol{\gamma} = \mathbf{0},$$

where the first equality comes from (26). Therefore, the error satisfies

$$(\mathbf{L} - \rho\boldsymbol{\mathcal{I}}_{\hat{\mathbf{V}}})\mathbf{e} = \sum_{j=m-1}^{m+2} \mathcal{O}(1)\mathbf{1}_j - \sum_{j=1}^{m} \gamma_j \mathbf{1}_j + \sum_{j=1}^{M} \mathbf{O}(h^4)\mathbf{1}_j$$

The theorem is proved. □

Proposition 3.1 identifies the error equation $\mathbf{e} = (\mathbf{L} - \rho\boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}})^{-1}\mathbf{r}$. The following proposition tells us how the operator $(\mathbf{L} - \rho\boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}})^{-1}$ behaves.

Proposition 3.2. *Consider a matrix*

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{bmatrix} \tag{30}$$

where both $\mathbf{L}_{11}$ and $\mathbf{L}_{22}$ are nonsingular, and the submatrices $\mathbf{L}_{11},\ \mathbf{L}_{12},\ \mathbf{L}_{21},\ \mathbf{L}_{22}$ are of sizes $m \times m$, $m \times (M-m)$, $(M-m) \times m$ and $(M-m) \times (M-m)$ respectively. Suppose that ρ is a positive number such that $\rho \gg \max_{ij}\{|L_{i,j}|\}$, and $\boldsymbol{\mathcal{I}}$ is a diagonal matrix such that $(\boldsymbol{\mathcal{I}})_{i,i} = 1$ for $i = 1, \ldots, m$ and $(\boldsymbol{\mathcal{I}})_{i,i} = 0$ for $i = m+1, \ldots, M$. Then $(\mathbf{L} - \rho\boldsymbol{\mathcal{I}})^{-1}$ has the approximation

$$(\mathbf{L} - \rho\boldsymbol{\mathcal{I}})^{-1} \approx \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{22}^{-1} \end{bmatrix}. \tag{31}$$

Proof. Since $\mathbf{L}_{11},\ \mathbf{L}_{22}$ are nonsingular, the exact inverse matrix of $\mathbf{L} - \rho\boldsymbol{\mathcal{I}}$ is

$$(\mathbf{L} - \rho\boldsymbol{\mathcal{I}})^{-1} = \begin{bmatrix} \mathbf{B} & -\mathbf{B}\mathbf{L}_{12}\mathbf{L}_{22}^{-1} \\ -\mathbf{L}_{22}^{-1}\mathbf{L}_{21}\mathbf{B} & \mathbf{L}_{22}^{-1} + \mathbf{L}_{22}^{-1}\mathbf{L}_{21}\mathbf{B}\mathbf{L}_{12}\mathbf{L}_{22}^{-1} \end{bmatrix}$$

where $\mathbf{B} = (\mathbf{L}_{11} - \rho\mathbf{I} - \mathbf{L}_{12}\mathbf{L}_{22}^{-1}\mathbf{L}_{21})^{-1}$. Since $\rho \gg \max_{ij}\{|L_{i,j}|\}$, we have $\mathbf{B} \approx \frac{1}{\rho}\mathbf{I}$. Therefore,

$$(\mathbf{L} - \rho\boldsymbol{\mathcal{I}}^{n+1})^{-1} \approx \begin{bmatrix} -\frac{1}{\rho}\mathbf{I} & -\frac{1}{\rho}\mathbf{L}_{12}\mathbf{L}_{22}^{-1} \\ \frac{1}{\rho}\mathbf{L}_{22}^{-1}\mathbf{L}_{21} & \mathbf{L}_{22}^{-1} + \frac{1}{\rho}\mathbf{L}_{22}^{-1}\mathbf{L}_{21}\mathbf{L}_{12}\mathbf{L}_{22}^{-1} \end{bmatrix} \approx \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{22}^{-1} \end{bmatrix}.$$

□

Let the matrix $\mathbf{L}$ denote the discretization of $\mathcal{L}$ in (2) be decomposed as in (30), with $\mathbf{L}_{22}$ being the lower-right block. Let $\mathbf{r}$ be defined as in Proposition 3.1 and $\mathbf{r}_{m+1:M}$ denote the subvector of $\mathbf{r}$ starting from entry $m+1$ to M. We then have the following theorem.

Theorem 3.3. *Under the same assumptions as in Proposition 3.1, when using the fourth-order finite difference scheme in (24) to solve the penalized PDE (23), the error satisfies*

$$\mathbf{e} \approx \begin{bmatrix} \mathbf{O} \\ \mathbf{L}_{22}^{-1}\mathbf{r}_{m+1:M} \end{bmatrix}.$$

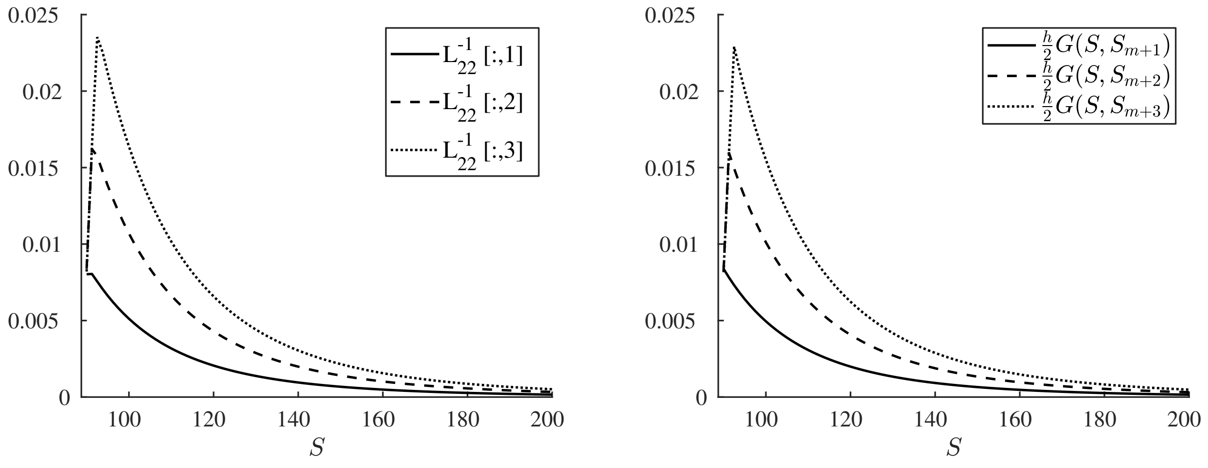
Proof. The theorem is easily obtained from Propositions 3.1 and 3.2. □

Theorem 3.3 shows that the penalty method obtains the exact solution within a pre-specified tolerance on the penalty region, while on the PDE region where the solution satisfies the PDE (5), the error is given by $\mathbf{L}_{22}^{-1}\mathbf{r}_{m+1:M}$. Note that $\mathbf{L}_{22}$ is just the fourth-order finite difference discretization of the operator $\mathcal{L}$ on the grid $S_{m+1},\ S_{m+2},\ \ldots,\ S_M$ with S_m and S_{M+1} as the boundary points. Note that $\mathbf{e}$ is implicitly given zero boundary conditions. On a uniform grid with step size h, $\mathbf{L}_{22}^{-1}$ can be thought of as the finite difference analogue of the continuous Green's function of $\mathcal{L}$ on the PDE region, scaled by the grid size, $\frac{h}{2}G(S, S_j)$, for $j = m+1, \ldots, M$. Therefore, we have

$$\mathbf{e} \approx \sum_{j=m+1}^{M} r_j \begin{bmatrix} \mathbf{0} \\ [\mathbf{L}_{22}^{-1}]_{:,j} \end{bmatrix} \approx \sum_{j=m+1}^{M} r_j \frac{h}{2} \begin{bmatrix} \mathbf{0} \\ G(\mathbf{S}_2, S_j) \end{bmatrix}, \tag{32}$$

where $G(\mathbf{S}_2, S_j)$ is a column vector of Green's function values $G(S, S_j)$ at points $\mathbf{S}_2 = \{S_{m+1}, \ldots, S_M\}$. Here, we use the fact that $\mathbf{L}_{22}^{-1}$ is a discrete analogue of the continuous Green's function.

To visualize the shape of the $\mathbf{L}_{22}^{-1}$ and its relation to the continuous Green's function, Figure 2 gives the first three columns of the $\mathbf{L}_{22}^{-1}$ and corresponding $\frac{h}{2}G(\mathbf{S}, S_{m+j})$, for $j = 1,\ 2$ and 3, for the operator $\mathcal{L}_{BS}$ given by Equation (2), on an example nonuniform grid where the free boundary is located at $S_m < S_f = 95.1 < S_{m+1}$. We can see that $[\mathbf{L}_{22}^{-1}]_{:,j}$ and $\frac{h}{2}G(S, S_{m+j})$ behave similarly.



(a) The first 3 columns of $\mathbf{L}_{22}^{-1}$ (b) The continuous Green's function $\frac{h}{2}G(S, S_j)$

Figure 2: (a) The first three columns of $\mathbf{L}_{22}^{-1}$ on an example uniform grid of size $h = 1.25$; (b) The scaled continuous Green's function $\frac{h}{2}G(S, S_j)$ for the operator $\mathcal{L}_{BS}$ at S_{m+1}, S_{m+2} and S_{m+3}. The free boundary location is $S_f = 89.748$. Note that the zero value on the left of the free boundary is not included.

In order to analyze the error behavior, we turn to understanding the properties of the Green's function $G(S, S_j)$, which is easier to investigate than its discrete analogue $\mathbf{L}_{22}^{-1}$. The following proposition gives the exact expression of the Green's function to a general operator.

Proposition 3.3. *Suppose that $Tu(x) = 0$ is a constant-coefficient, second-order homogeneous differential equation. Let $u(x) = c_1 e^{\xi_1 x} + c_2 e^{\xi_2 x}$ denote the general solution to this equation. Suppose further that ξ_1 and ξ_2 are real and $\xi_1 \neq \xi_2$. Then, the Green's function for the operator T defined on the domain $[a, b]$ is*

$$G(x, \bar{x}) = \begin{cases} \dfrac{e^{(\xi_2-\xi_1)b} - e^{(\xi_2-\xi_1)\bar{x}}}{(\xi_2-\xi_1)e^{\xi_2\bar{x}}\left(e^{(\xi_2-\xi_1)b} - e^{(\xi_2-\xi_1)a}\right)} \left(e^{\xi_2 x} - e^{(\xi_2-\xi_1)a+\xi_1 x}\right), & a \le x < \bar{x}, \\ \dfrac{e^{(\xi_2-\xi_1)a} - e^{(\xi_2-\xi_1)\bar{x}}}{(\xi_2-\xi_1)e^{\xi_2\bar{x}}\left(e^{(\xi_2-\xi_1)b} - e^{(\xi_2-\xi_1)a}\right)} \left(e^{\xi_2 x} - e^{(\xi_2-\xi_1)b+\xi_1 x}\right), & \bar{x} \le x \le b. \end{cases} \tag{33}$$

Moreover, we have

$$G(x,\bar{x}) = \mathcal{O}(\bar{x}-a), \quad \text{as } \bar{x} \to a.$$

Proof. The computation of the Green's function follows the standard procedure and we omit it here. Note that as $\bar{x} \to a$, we have

$$e^{(\xi_2-\xi_1)a} - e^{(\xi_2-\xi_1)\bar{x}} = e^{(\xi_2-\xi_1)a}\left(1 - e^{(\xi_2-\xi_1)(\bar{x}-a)}\right) \approx (\xi_1-\xi_2)(\bar{x}-a)e^{(\xi_2-\xi_1)a} = \mathcal{O}(\bar{x}-a).$$

Hence, we see that $G(x,\bar{x}) = \mathcal{O}(\bar{x}-a)$ as $\bar{x} \to a$. $\square$

From Equation (32), it is obvious that the global error is dominated by the $\mathcal{O}(1)$ entries r_{m+1} and r_{m+2} (if S_m is not on the free boundary) which is propagated to every other grid point through the discrete Green's function. Hence, at first glance, it appears that solving (24) using fourth-order finite differences will only give us first-order $\mathcal{O}(h)$ convergence. However, from the conclusion of Proposition 3.3, we observe that second-order $\mathcal{O}(h^2)$ convergence is obtained for points far enough away from the free boundary. This is because, in Equation (32), the Green's function $G(S,S_{m+1}) = \mathcal{O}(S_{m+1}-S_f) = \mathcal{O}(h)$, and $G(S,S_{m+2}) = \mathcal{O}(S_{m+2}-S_f) = \mathcal{O}(h)$ as $h \to 0$. For a visual demonstration of property, Figure 3 gives an illustration of Green's function for a hypothetical second-order differential equation on two successive grid refinements.

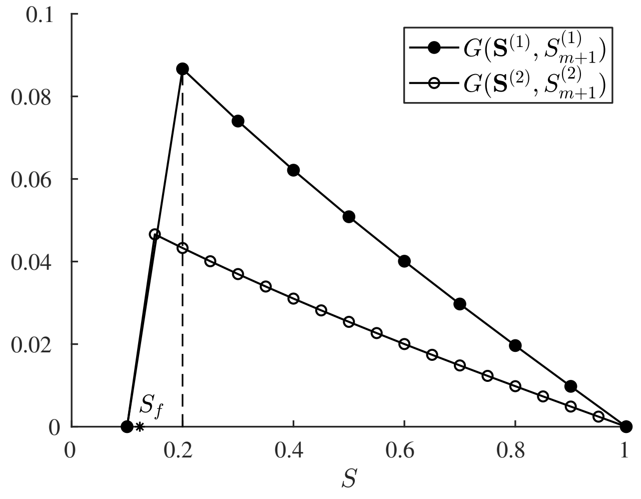


Figure 3: Illustration of Green's functions on two successive grid refinements $\mathbf{S}^{(1)}, \mathbf{S}^{(2)}$, for a hypothetical second-order differential equation.

3.3.2 Initial value problems

When the time variable is included, the analysis becomes more involved. However, the conclusions are similar to the ones for boundary value problems. Consider the original LCP given by (1), and the corresponding penalty equation (4), which we repeat here for convenience:

$$\partial_t V = \mathcal{L}V + \rho \mathcal{I}_V (V^* - V). \tag{34}$$

From the proof in Proposition 3.1, we can see that

$$\partial_t \hat{V} = \mathcal{L}\hat{V} + \gamma, \quad \text{with } \gamma(t,S) \equiv (\partial_t - \mathcal{L})\hat{V}(t,S)\mathbb{1}_{S<S_f(t)} = (\partial_t - \mathcal{L})V^*(t,S)\mathbb{1}_{S<S_f(t)}, \tag{35}$$

and

$$\partial_t \hat{\mathbf{V}} = \mathbf{L}\hat{\mathbf{V}} + \mathbf{b} + \boldsymbol{\theta} + \boldsymbol{\gamma}, \quad \text{with } [\boldsymbol{\gamma}(t)]_j \equiv (\partial_t - \mathcal{L})V^*(t,S_j)\mathbb{1}_{1\le j\le m(t)}, \tag{36}$$

where $m(t)$ is the node index at time t such that $S_{m(t)} \le S_f(t) < S_{m(t)+1}$, and $\mathbb{1}_{1\le j\le m(t)}$ is the indicator function, which is one when $1 \le j \le m(t)$ and zero otherwise, and $\boldsymbol{\theta} = \boldsymbol{\theta}_1$ or $\boldsymbol{\theta}_2$ as defined (27) and (28).

Since our approximate solution is computed from a fully discretized system, we are interested in studying the discretization in t. Consider the BDF4 discretization applied to Equations (36) starting at the fourth time step. We have, for the exact LCP solution,

$$\frac{\frac{25}{12}\hat{\mathbf{V}}^{n+4} - 4\hat{\mathbf{V}}^{n+3} + 3\hat{\mathbf{V}}^{n+2} - \frac{4}{3}\hat{\mathbf{V}}^{n+1} + \frac{1}{4}\hat{\mathbf{V}}^{n}}{k} = \mathbf{L}\hat{\mathbf{V}}^{n+4} + \mathbf{b}^{n+4} + \boldsymbol{\theta} + \boldsymbol{\gamma} + \boldsymbol{\beta}, \tag{37}$$

where $\boldsymbol{\beta}$ is the truncation error of the BDF4 time stepping scheme of (36). Note that the fully discrete system that we are actually solving satisfies

$$\frac{\frac{25}{12}\tilde{\mathbf{V}}^{n+4} - 4\tilde{\mathbf{V}}^{n+3} + 3\tilde{\mathbf{V}}^{n+2} - \frac{4}{3}\tilde{\mathbf{V}}^{n+1} + \frac{1}{4}\tilde{\mathbf{V}}^{n}}{k} = \mathbf{L}\tilde{\mathbf{V}}^{n+4} + \mathbf{b}^{n+4} + \rho\mathcal{I}_{\tilde{\mathbf{V}}}(\mathbf{V}^* - \tilde{\mathbf{V}}). \tag{38}$$

The following proposition gives the relationship between the solution $\hat{V}$ of the exact LCP and the solution $\tilde{\mathbf{V}}$ of the fully discrete system (38).

Proposition 3.4. *Let $\hat{V}$ be the solution to the exact LCP (1), and V be the solution to the continuous penalty equation (34). Assume that the penalty terms in the fully discrete equations reflect the correct behavior, i.e., $\mathcal{I}_{\tilde{\mathbf{V}}} = \mathcal{I}_{\hat{\mathbf{V}}}$. Then, we have*

$$\left(\frac{25}{12k}\mathbf{I} - \mathbf{L} + \rho\mathcal{I}_{\hat{\mathbf{V}}}\right)\mathbf{e}^{n+4} = \frac{1}{k}\left(4\mathbf{e}^{n+3} - 3\mathbf{e}^{n+2} + \frac{4}{3}\mathbf{e}^{n+1} - \frac{1}{4}\mathbf{e}^{n}\right) + (\boldsymbol{\gamma} + \boldsymbol{\theta} + \boldsymbol{\beta}), \tag{39}$$

where $\mathbf{e} = \hat{\mathbf{V}} - \tilde{\mathbf{V}}$ is error in the solution.

Proof. First, notice that $\rho\mathcal{I}_V(V^* - V) = \rho\mathcal{I}_{\hat{V}}(\hat{V} - V)$, since

$$\rho\mathcal{I}_V(V^* - V) = \rho\mathcal{I}_V(V^* - \hat{V} + \hat{V} - V) = \rho\mathcal{I}_V(V^* - \hat{V}) + \rho\mathcal{I}_V(\hat{V} - V) = \rho\mathcal{I}_V(\hat{V} - V) = \rho\mathcal{I}_{\hat{V}}(\hat{V} - V).$$

Subtracting Equation (38) from (37), we have

$$\frac{\frac{25}{12}\mathbf{e}^{n+4} - 4\mathbf{e}^{n+3} + 3\mathbf{e}^{n+2} - \frac{4}{3}\mathbf{e}^{n+1} + \frac{1}{4}\mathbf{e}^{n}}{k} = (\mathbf{L} - \rho\mathcal{I}_{\tilde{\mathbf{V}}})\mathbf{e}^{n+4} + \boldsymbol{\gamma} + \boldsymbol{\theta} + \boldsymbol{\beta},$$

By rearranging, we get Equation (39). □

Corresponding to Proposition 3.1 in the previous section which describes the error equation for time-independent free boundary problems, Proposition 3.4 gives an expression of the error evolution for solving time-dependent free boundary problems using a fourth-order finite difference scheme and BDF4. For convenience of discussion, define

$$\mathbf{L}_k = \frac{25}{12k}\mathbf{I} - \mathbf{L},$$

and

$$\mathbf{r}_k^{n+3} = \frac{1}{k}\left(4\mathbf{e}^{n+3} - 3\mathbf{e}^{n+2} + \frac{4}{3}\mathbf{e}^{n+1} - \frac{1}{4}\mathbf{e}^{n}\right) + (\boldsymbol{\gamma} + \boldsymbol{\theta} + \boldsymbol{\beta}).$$

In addition, assume that, at the $(n+4)$-th time step, the free boundary is located in between S_m and S_{m+1}, i.e., $S_m \le S_f(t_{n+4}) < S_{m+1}$, on the space grid $\{S_0, S_1, \ldots, S_m, \ldots S_M\}$. Similar to the discussion for boundary value problems, we divide the matrix $\mathbf{L}_k$ into four block submatrices

$$\mathbf{L}_k = \begin{bmatrix} ([\mathbf{L}_k])_{11} & ([\mathbf{L}_k])_{12} \\ ([\mathbf{L}_k])_{21} & ([\mathbf{L}_k])_{22} \end{bmatrix}$$

where the block matrices $([\mathbf{L}_k])_{11},\ ([\mathbf{L}_k])_{12},\ ([\mathbf{L}_k])_{21},\ ([\mathbf{L}_k])_{22}$ are of sizes $m \times m$, $m \times (M-m)$, $(M-m)\times m$ and $(M-m)\times(M-m)$, respectively. Corresponding to Theorem 3.3 for time-independent problems, the error decomposition for time-dependent problems is given by Theorem 3.4.

Theorem 3.4. *When using BDF4 time stepping and the fourth-order finite difference scheme to solve the penalized PDE (34), the solution error at the $(n+4)$-th time step satisfies*

$$\mathbf{e}^{n+4} \approx \begin{bmatrix} \mathbf{O} \\ ([\mathbf{L}_k]_{22})^{-1}\,[\mathbf{r}_k^{n+3}]_{m+1:M} \end{bmatrix}.$$

Proof. From the results in Proposition 3.4, we know that the error is the solution to

$$(\mathbf{L}_k + \rho\mathcal{I}_{\tilde{\mathbf{V}}})\mathbf{e}^{n+4} = \mathbf{r}_k^{n+3}.$$

Therefore, we get

$$\mathbf{e}^{n+4} = (\mathbf{L}_k + \rho\mathcal{I}_{\tilde{\mathbf{V}}})^{-1}\mathbf{r}_k^{n+3}.$$

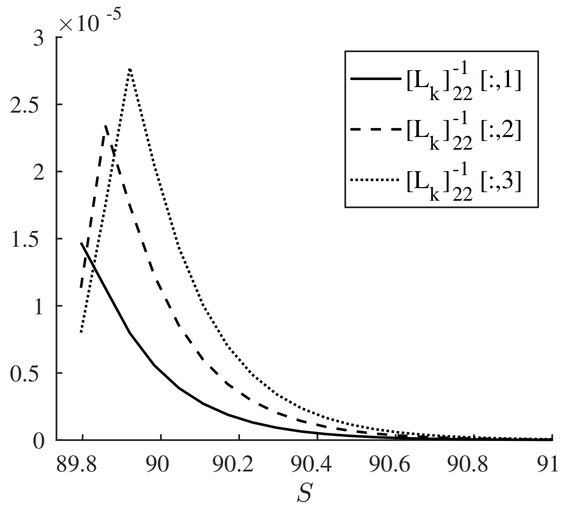
Then by making use of the conclusion in Proposition 3.2, the theorem is proved. □

Theorem 3.4 shows that the errors in the approximate solutions of moving boundary problems behave in a similar way to the solution of free boundary problems. The solution on the penalty region is computed exactly within a tolerance, while on the PDE region, in addition to the truncation errors, the solution errors from previous time steps also contribute to the solution error at the current time step. The error propagation is governed by $([\mathbf{L}_k]_{22})^{-1}$, which depends on both the time stepping and space discretization schemes. Similar to the discussion of boundary value problems, it can be treated as the discrete analogue of the Green's function to the continuous operator on the PDE region.

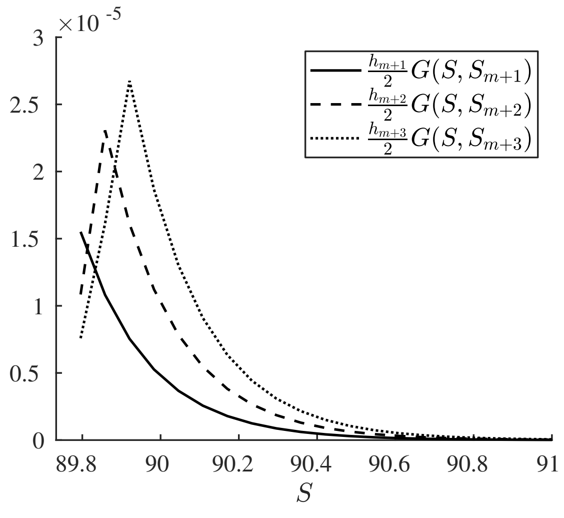
In the following, we consider a concrete example of the Black-Scholes operator, i.e., let $\mathcal{L} = \mathcal{L}_{BS}$. Instead of studying $\mathbf{L}_k$, which is hard to analyze, we investigate the Green's function for the continuous operator $\mathcal{L}_k = \frac{25}{12k} - \mathcal{L}$ on the PDE region for a fixed time step size k. In Figure 4, we show the comparison of the graphs of $G(\mathbf{S}_2, S_j)$ on the PDE region and $[\mathbf{L}_k]_{22}^{-1}$ for the first three Green's functions on an example grid, where the free boundary is located at $S_f = 93.11$. Again, we see that they have the same shape with similar magnitudes. By performing the usual variable transformation $S = Ke^x$ to $\mathcal{L}_k$, we can get a transformed operator $\mathcal{L}_{k,x} = -\partial_{xx} - (\kappa - 1)\partial_x + (\kappa + \frac{25}{6\kappa\sigma^2})$, whose Green's function is given by Equation (33) in Proposition 3.3, with $\xi_1 = \frac{-(\kappa-1)+\sqrt{(\kappa+1)^2+4\lambda}}{2}$, $\xi_2 = \frac{-(\kappa-1)-\sqrt{(\kappa+1)^2+4\lambda}}{2}$, where $\lambda = \frac{25}{6\kappa\sigma^2}$, and $\kappa = \frac{2r}{\sigma^2}$. Let $x_{m+1} = \log(S_{m+1}/K)$ and $x_f = \log(S_f/K)$. When $x_{m+1} - x_f$ is small enough, we have that $G(x, x_{m+1}) = \mathcal{O}(x_{m+1} - x_f)$ by Proposition 3.3. Since

$$x_{m+1} - x_f = \log(S_{m+1}/K) - \log(S_f/K) = \log\left(1 + \frac{S_{m+1} - S_f}{S_f}\right) \approx \frac{S_{m+1} - S_f}{S_f} = \mathcal{O}(h),$$

we see that when $S_{m+1} \to S_f > 0$, the Green's function $G(S, S_{m+1}) = \mathcal{O}(h)$. Therefore, using the same argument as in Section 3.3.1, we see that the error in the solution of Equation (38) is of order $\mathcal{O}(h^2)$.



(a) The first 3 columns of $[\mathbf{L}_k]_{22}^{-1}$



(b) The continuous Green's function $\frac{h_j}{2}G(S, S_j)$

Figure 4: (a) The first three columns of $[\mathbf{L}_k]_{22}^{-1}$ on an example nonuniform grid; (b) The scaled continuous Green's function $\frac{h_j}{2}G(S, S_j)$ for the operator $\mathcal{L}_{BS}$ at S_{m+1}, S_{m+2} and S_{m+3}. The free boundary location is $S_f = 93.11$. Note that the zero value on the left of the free boundary is not included.

3.4 Grid crossing

In Section 3.3, the error analysis is based on the assumption that the time derivative is computed exactly. However, this assumption is generally not true, due to the existence of the moving free boundary. To see this, we can look at the example grid shown in Figure 5. With BDF4 time stepping, the time derivative of the solution at point p_1 is computed by a linear combination of the solutions at the four points directly prior to p_1. However, since p_1 and the previous points lie on different sides of the free boundary, the BDF4 scheme has degenerated accuracy, similar to the situation for the nonsmooth space discretization discussed in Section 3.3, where the fourth-order finite difference scheme degenerates to only first-order convergence due to piecewise smoothness in the solution. These black hollow points are problematic points for the BDF4 time stepping scheme, and we would like to avoid them. On the other hand, for the black solid points in Figure 5, such as point p_2, the time derivative of the solution at p_2 with the BDF4 scheme uses only the grid points above the free boundary. Hence, the BDF4 scheme at the black solid points is fourth-order accurate. In general, we can see that the number of problematic points in a problem depends on how quickly the free boundary moves relative to the grid spacing. A slow-moving free boundary will have a smaller number of black hollow points on a fixed grid.

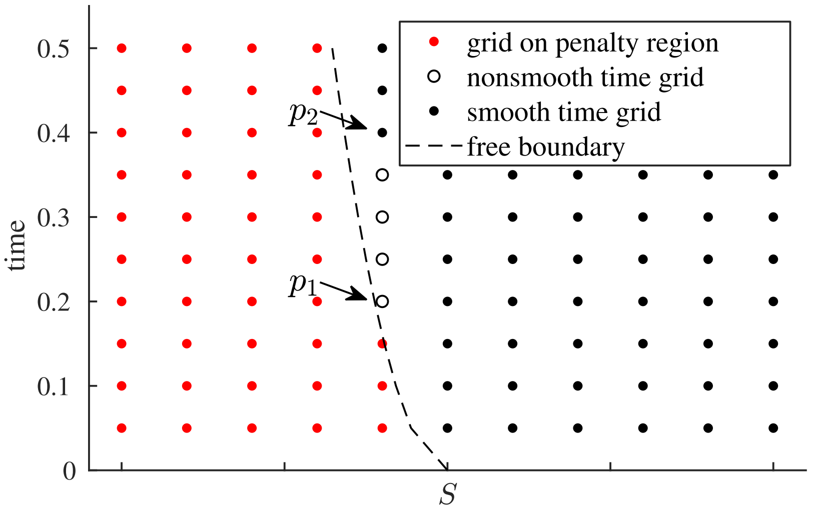


Figure 5: **An example layout of grid points in the time and space domain.** The dashed line is the free boundary. The red points to the left of the free boundary are on the penalty region, and the points to the right of the free boundary are on the PDE region. Unlike at the solid black points, BDF4 has degenerated accuracy at the hollow black points because it involves the solutions that lie on different sides of the free boundary.

At minimum, we require the time discretization to be accurate enough so as not to affect the convergence order in space. Despite the complicated behavior of the time derivative discretization, we do not explicitly deal with the loss of accuracy in the time derivative in our algorithm. Instead, we apply a time variable transformation to change the shape of the free boundary and try to reduce the number of problematic points in the time stepping. This turns out to be good enough to maintain high-order accuracy in space discretization with appropriate time and space stretching.

3.5 Extrapolating the numerical solution

In this section, we discuss how to approximate the free boundary location and the derivative jumps using a given numerical solution, and analyze some related technical details.

3.5.1 Approximation of the solution derivatives

From the conclusions in Theorem 3.3, we know that the numerical solution of the PDE in (5) from a standard fourth-order solve is exact within a tolerance $\mathcal{O}(1/\rho)$ at points $S_j \leq S_m$ and second-order accurate at the points $S_j > S_m$. More precisely, the error is of the form

$$\begin{aligned} \mathbf{e} &\approx \sum_{j=m+1}^{M} r_j \frac{h}{2} \begin{bmatrix} \mathbf{0} \\ G(\mathbf{S}_2, S_j) \end{bmatrix} \\ &= \mathcal{O}(h) \begin{bmatrix} \mathbf{0} \\ G(\mathbf{S}_2, S_{m+1}) \end{bmatrix} + \mathcal{O}(h) \begin{bmatrix} \mathbf{0} \\ G(\mathbf{S}_2, S_{m+2}) \end{bmatrix} + \sum_{j=m+3}^{M} \mathcal{O}(h^5) \begin{bmatrix} \mathbf{0} \\ G(\mathbf{S}_2, S_j) \end{bmatrix}. \end{aligned} \tag{40}$$

Suppose that we are given the numerical solution of $\mathcal{O}(h^2)$ accuracy with error components given by Equation (40). Suppose further that the free boundary is located in between grid points S_m and S_{m+1}, as shown on an example uniform grid in Figure 6. Since the Green's functions $G(S, S_j)$ are piecewise smooth with first-derivative jumps at points S_j, and the $\mathcal{O}(h^5)$ terms in (40) are negligible compared to the $\mathcal{O}(h)$ terms, the solution error $\mathbf{e}$ is of order $\mathcal{O}(h)$,

$$e_j \approx \mathcal{O}(h)G(S_j, S_{m+1}) + \mathcal{O}(h)G(S_j, S_{m+2}),$$

and it is smooth for $S_j \geq S_{m+2}$, which means the numerical solution is also smooth. This is the key to the success of our method to approximate derivatives without of loss of accuracy. In order to maintain the same order of accuracy as the numerical solutions, when applying finite differences to approximate the derivatives, we should only use points on the right of S_{m+2} (inclusive). Therefore, to approximate the first derivative of the solution at the free boundary $S_f < S_{m+1}$ to $\mathcal{O}(h^2)$ accuracy, we first obtain a second-order approximation of the first derivative at the points $S_{m+2}, S_{m+3}, S_{m+4}$, denoted by $\tilde{V}'_{m+2}, \tilde{V}'_{m+3}, \tilde{V}'_{m+4}$, respectively. Then, we extrapolate the solution derivative at S_f using the computed $\tilde{V}'_{m+2}, \tilde{V}'_{m+3}, \tilde{V}'_{m+4}$ by

$$\tilde{V}'(S) = \sum_{i=2}^{4} \left(\tilde{V}'_{m+i} \prod_{j=2, j\neq i}^{4} \frac{S - S_{m+j}}{S_{m+i} - S_{m+j}} \right),$$

i.e., by using a quadratic Lagrange polynomial. The obtained approximate derivative at S_f is of $\mathcal{O}(h^2)$ accuracy. Higher-order derivatives are computed in the same way. Hence, from the analysis in Section 3.2.3, we see that the correction terms computed using the approximate derivative jumps will be of $\mathcal{O}(h^2)$ accurate, which is more than the required $\mathcal{O}(h)$ accuracy to increase the order of the corrected finite differences. Similarly, we use $\tilde{V}_{m+2}, \tilde{V}_{m+3}, \tilde{V}_{m+4}, \tilde{V}_{m+5}$ and a cubic polynomial for the third-order approximation, and $\tilde{V}_{m+2}, \tilde{V}_{m+3}, \tilde{V}_{m+4}, \tilde{V}_{m+5}, \tilde{V}_{m+6}$ and a quartic polynomial for the fourth-order approximation of the solution derivatives at the free boundary.

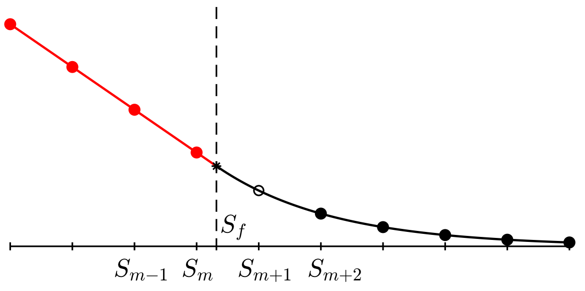


Figure 6: An example grid on which a free boundary problem is defined. Point S_f is the free boundary location.

3.5.2 Approximation of the free boundary location

To approximate the free boundary location, we apply the smooth pasting condition of the derivative at the free boundary in Equation (3), which we repeat here for convenience:

$$\frac{\partial V}{\partial S}(t, S_f(t)) = \frac{\partial V^*}{\partial S}(S_f(t)). \tag{41}$$

Assume we have already calculated the approximate derivatives $\tilde{V}'_{m+2}$, $\tilde{V}'_{m+3}$, $\tilde{V}'_{m+4}$, $\tilde{V}'_{m+5}$ and $\tilde{V}'_{m+6}$ at S_{m+2}, S_{m+3}, S_{m+4}, S_{m+5}, S_{m+6} as in Section 3.5.1. We then apply the quartic Lagrange polynomial to fit the derivative by

$$\tilde{V}'(S) = \sum_{i=2}^{6} \left(\tilde{V}'_{m+i} \prod_{j=2, j\neq i}^{6} \frac{S - S_{m+j}}{S_{m+i} - S_{m+j}} \right).$$

Then the approximate free boundary is obtained by Newton's root finding algorithm such that

$$\tilde{V}'(S) - \frac{\partial V^*}{\partial S}(S) \approx 0.$$

The approximate free boundary obtained in this way is expected to be of the same order as the numerical solution. Hence, from the analysis in Section 3.2.3, we see that the correction terms computed using the approximate free boundary will be $\mathcal{O}(h)$ accurate for the first correction, as is required to increase the order of the corrected finite differences. Note that we decided to use the smoothing pasting condition (41) to locate the free boundary instead of the value matching condition $V(t, S_f(t)) = V^*(S_f(t))$. The reason for this is that the value matching equation has a zero derivative at the root. Therefore, Newton's root-finding method is slow if value matching is used.

4 Algorithm

4.1 A fourth-order deferred correction algorithm for solving free boundary problems

We are now ready to present a fourth-order deferred correction finite difference algorithm for solving free and moving boundary problems. To start, we first present the algorithm for solving free boundary problems where no time variable is involved. The penalized equation for solving free boundary problems is given by Equation (5). We discretize Equation (5) using fourth-order finite difference and get a nonlinear system

$$\mathbf{L}\tilde{\mathbf{V}} + \mathbf{b} + \rho\boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}}(\mathbf{V}^* - \tilde{\mathbf{V}}) = 0, \tag{42}$$

with the notation defined in Section 3.1. This system can be solved efficiently by a generalized Newton's iteration as described in [10], which is also referred to as discrete penalty iteration.

The main idea of our algorithm is to use a deferred correction technique to eliminate the lower-order errors in the finite difference approximation introduced by piecewise smoothness in the solution. We illustrate our correction scheme by considering only the leading order terms on the right-hand side of Equations (17)–(20) which are associated with the jump in the second derivative. The other terms are corrected in the same manner and we omit the discussion.

Recall that the fourth-order FD discretization of $\frac{\partial^2 V}{\partial S^2}$ and $\frac{\partial V}{\partial S}$ is $\bar{\mathbf{L}}_2\tilde{\mathbf{V}}_{\text{aug}}$ and $\bar{\mathbf{L}}_1\tilde{\mathbf{V}}_{\text{aug}}$, respectively, at the interior nodes of a grid $S_0 < S_1 < \ldots < S_{M+1}$, where $\bar{\mathbf{L}}_2$ and $\bar{\mathbf{L}}_1$ are both $M \times (M+2)$ FD coefficient matrices, and $\tilde{\mathbf{V}}_{\text{aug}} = [\tilde{V}_0, \tilde{V}_1, \ldots, \tilde{V}_{M+1}]^T$, as defined in Section 3.1. Suppose that, at some time step t_n, the free boundary location $S_f(t_n)$ is in between S_m and S_{m+1}, i.e., $S_m < S_f < S_{m+1}$ as shown in Figure 6. The second derivative jump at the free boundary is pre-computed to be $\Delta V_f''$ (either approximate or exact). From Theorem 3.2, making $\delta = S_{m+1} - S_f$, and picking appropriate FD coefficients, we see that the correction terms for the FD approximation of the second derivatives at nodes $S_{m-1},\ S_m,\ S_{m+1}$ and S_{m+2}, are computed by

$$\begin{aligned}
C_1'' &= \frac{(S_{m+1} - S_f)^2}{2}\bar{\mathbf{L}}_2(m-1, m+2)\Delta V_f'',\\
C_2'' &= \left(\frac{(S_{m+1} - S_f)^2}{2}\bar{\mathbf{L}}_2(m, m+2) + \frac{(S_{m+2} - S_f)^2}{2}\bar{\mathbf{L}}_2(m, m+3)\right)\Delta V_f'',\\
C_3'' &= -\left(\frac{(S_f - S_{m-1})^2}{2}\bar{\mathbf{L}}_2(m+1, m) + \frac{(S_f - S_m)^2}{2}\bar{\mathbf{L}}_2(m+1, m+1)\right)\Delta V_f'',\\
C_4'' &= -\frac{(S_f - S_m)^2}{2}\bar{\mathbf{L}}_2(m+2, m+1)\Delta V_f'',
\end{aligned}$$

where $\bar{\mathbf{L}}_2(i,j)$ denotes the (i,j) entry of the coefficient matrix $\bar{\mathbf{L}}_2$. Corrections to the remaining higher-order derivatives can be computed similarly. By replacing the $\bar{\mathbf{L}}_2$ entries with the corresponding $\bar{\mathbf{L}}_1$ entries, we can similarly calculate the correction terms C_j' to the first derivative approximations, for $j = 1, 2, 3, 4$, at nodes $S_{m-1},\ S_m,\ S_{m+1}$ and S_{m+2}.

With these correction entries in hand, instead of solving Equation (24), we solve a modified system

$$\mathbf{L}\tilde{\mathbf{V}} + \mathbf{b} + \rho\mathcal{I}_{\tilde{\mathbf{V}}}(\mathbf{V}^* - \tilde{\mathbf{V}}) + \mathbf{a}_2 + \mathbf{a}_1 = \mathbf{0}, \tag{43}$$

with correction terms $\mathbf{a}_2$ and $\mathbf{a}_1$, where

$$\mathbf{a}_2 = [0,\ \dots,\ 0,\ p(S_{m-1})C_1'',\ p(S_m)C_2'',\ p(S_{m+1})C_3'',\ p(S_{m+2})C_4'',\ 0,\ \dots,\ 0]^T,$$
$$\mathbf{a}_1 = [0,\ \dots,\ 0,\ w(S_{m-1})C_1',\ w(S_m)C_2',\ w(S_{m+1})C_3',\ w(S_{m+2})C_4',\ 0,\ \dots,\ 0]^T.$$

Ideally, we would know the exact derivative jumps and apply them to correct the FDs when discretizing the PDE. However, these jumps are not known a priori in the setting of this work. Therefore, we make use of the approximate solution derivatives and free boundary location that we get, as described in Section 3.5, to compute the approximate correction terms. Applying these correction terms increase the order of the finite difference approximation. As a result, it also increase the order of the new approximate solution when we solve the discrete system again with the corrected FDs. We give the details in Algorithm 1.

Algorithm 1: A fourth-order FD algorithm for solving free boundary problems

```
1: Solve (42) to obtain an approximate solution $\tilde{\mathbf{V}}^{(0)}$ of order $\mathcal{O}(h^2)$ using penalty iteration.
   Find $S_m$ and $S_{m+1}$ as in Proposition 3.1.
   Compute the approximate free boundary $\tilde{S}_f^{(0)}$ to $\mathcal{O}(h^2)$ accuracy, using Newton's method with initial guess $(S_m + S_{m+1})/2$, as in Section 3.5.2.
   Approximate $V''(S_f)$ at $\tilde{S}_f^{(0)}$ to obtain $\tilde{V}_{S_f}''^{(0)} = V''(S_f) + O(h)$, as in Section 3.5.1.
2: Compute the FD corrections $\mathbf{a}_1$, $\mathbf{a}_2$ using $\tilde{S}_f^{(0)}$ and $V_{S_f}^{*}{}'' - \tilde{V}_{S_f}''^{(0)}$.
   Solve (43) to obtain an approximate solution $\tilde{\mathbf{V}}^{(1)}$ of $\mathcal{O}(h^3)$ using penalty iteration with initial guess $\tilde{\mathbf{V}}^{(0)}$.
   Compute the approximate free boundary $\tilde{S}_f^{(1)}$ to $\mathcal{O}(h^3)$ accuracy, using Newton's method with initial guess $\tilde{S}_f^{(0)}$.
   Approximate $V''(S_f),\ V'''(S_f)$ at $\tilde{S}_f^{(1)}$ to obtain $\tilde{V}_{S_f}''^{(1)} = V''(S_f) + \mathcal{O}(h^2)$, $\tilde{V}_{S_f}'''^{(1)} = V'''(S_f) + \mathcal{O}(h)$.
3: Compute the FD corrections $\mathbf{a}_1$, $\mathbf{a}_2$ using $\tilde{S}_f^{(1)}$ and $V_{S_f}^{*}{}'' - \tilde{V}_{S_f}''^{(1)}$, $V_{S_f}^{*}{}''' - \tilde{V}_{S_f}'''^{(1)}$.
   Solve (43) to obtain an approximate solution $\tilde{\mathbf{V}}^{(2)}$ of $\mathcal{O}(h^4)$ using penalty iteration with initial guess $\tilde{\mathbf{V}}^{(1)}$.
   Compute the approximate free boundary $\tilde{S}_f^{(2)}$ to $\mathcal{O}(h^4)$ accuracy, using Newton's method with initial guess $\tilde{S}_f^{(1)}$.
   Approximate $V''(S_f),\ V'''(S_f),\ V''''(S_f)$ at $\tilde{S}_f^{(2)}$ to obtain $\tilde{V}_{S_f}''^{(2)} = V''(S_f) + \mathcal{O}(h^3)$, $\tilde{V}_{S_f}'''^{(2)} = V'''(S_f) + \mathcal{O}(h^2)$, $\tilde{V}_{S_f}''''^{(2)} = V''''(S_f) + \mathcal{O}(h)$.
4: Compute the FD corrections $\mathbf{a}_1$, $\mathbf{a}_2$ using $\tilde{S}_f^{(2)}$ and $V_{S_f}^{*}{}'' - \tilde{V}_{S_f}''^{(2)}$, $V_{S_f}^{*}{}''' - \tilde{V}_{S_f}'''^{(2)}$, $V_{S_f}^{*}{}'''' - \tilde{V}_{S_f}''''^{(2)}$.
   Solve (43) to obtain an approximate solution $\tilde{\mathbf{V}}^{(3)}$ using penalty iteration with initial guess $\tilde{\mathbf{V}}^{(2)}$.
   Compute the approximate free boundary $\tilde{S}_f^{(3)}$ accuracy, using Newton's method with initial guess $\tilde{S}_f^{(2)}$.
```

Remark 4.1. *In fact, the algorithm already reaches fourth-order accuracy with two corrections. In practice, however, it turns out the third correction improves the error noticeably. Therefore, we present the algorithm with three corrections.*

4.2 A fourth-order deferred correction algorithm for solving moving boundary problems

When solving moving boundary problems, we also need to consider time discretization. We assume that Equation (4) has been discretized in time by BDF4, except for the first three time steps, and in space by standard fourth-order FDs, resulting in the nonlinear system (38), which we rearrange to get

$$\mathbf{A}\tilde{\mathbf{V}} = \tilde{\mathbf{y}} + k\rho\boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}}(\mathbf{V}^* - \tilde{\mathbf{V}}), \tag{44}$$

where

$$\mathbf{A} = \left(\frac{25}{12}\mathbf{I} - k\mathbf{L}\right), \quad \tilde{\mathbf{y}} = k\tilde{\mathbf{b}}^{n+4} + 4\tilde{\mathbf{V}}^{n+3} - 3\tilde{\mathbf{V}}^{n+2} + \frac{4}{3}\tilde{\mathbf{V}}^{n+1} - \frac{1}{4}\tilde{\mathbf{V}}^{n}.$$

Note that we have omitted the time indices for simplicity in (44). Similarly, instead of solving Equation (38), we solve a modified system

$$\frac{\frac{25}{12}\tilde{\mathbf{V}}^{n+4} - 4\tilde{\mathbf{V}}^{n+3} + 3\tilde{\mathbf{V}}^{n+2} - \frac{4}{3}\tilde{\mathbf{V}}^{n+1} + \frac{1}{4}\tilde{\mathbf{V}}^{n}}{k} = \mathbf{L}\tilde{\mathbf{V}}^{n+4} + \mathbf{b}^{n+4} + \rho\mathcal{I}_{\tilde{\mathbf{V}}}(\mathbf{V}^* - \tilde{\mathbf{V}}) + \mathbf{a}_1 + \mathbf{a}_2, \tag{45}$$

with correction terms $\mathbf{a}_1$ and $\mathbf{a}_2$ computed in the same way as for free boundary problems discussed in the previous section. The corrected system, after rearranging, becomes

$$\mathbf{A}\tilde{\mathbf{V}} = \tilde{\mathbf{y}} + k\rho\boldsymbol{\mathcal{I}}_{\tilde{\mathbf{V}}}(\mathbf{V}^* - \tilde{\mathbf{V}}) + k(\mathbf{a}_1 + \mathbf{a}_2). \tag{46}$$

Only slight modifications to Algorithm 1 are required to include BDF4 time stepping and solve moving boundary problems. Our high-order finite difference method for solving moving boundary problems is given in Algorithm 2.

Algorithm 2: A high-order FD algorithm for solving moving boundary problems

```
1: for each time step $t_n$ do
       Solve (44) to obtain an approximate solution $\tilde{\mathbf{V}}^{(0)}$ of order $\mathcal{O}(h^2)$ using penalty iteration
       with initial guess $\tilde{\mathbf{V}}^{(0)}$ at $t_{n-1}$.
       Find $S_m, S_{m+1}$ at $t_n$ as in Proposition 3.1.
       Compute the approximate free boundary $\tilde{S}_f^{(0)}$ at $t_n$ to $\mathcal{O}(h^2)$ accuracy, using Newton's
       method with initial guess $(S_m + S_{m+1})/2$ as in Section 3.5.2.
       Approximate $V''(t_n, S_f)$ at $\tilde{S}_f^{(0)}$ to obtain $\tilde{V}_{S_f}''^{(0)} = V''(t_n, S_f) + O(h)$, as in Section 3.5.1.
  2:   Compute the FD corrections $\mathbf{a}_1$, $\mathbf{a}_2$ using $\tilde{S}_f^{(0)}$ and $V_{S_f}^*{}'' - \tilde{V}_{S_f}''^{(0)}$.
       Solve (46) to obtain an approximate solution $\tilde{\mathbf{V}}^{(1)}$ of $\mathcal{O}(h^3)$ using penalty iteration with
       initial guess $\tilde{\mathbf{V}}^{(0)}$.
       Compute the approximate free boundary $\tilde{S}_f^{(1)}$ at $t_n$ to $\mathcal{O}(h^3)$ accuracy, using Newton's
       method with initial guess $\tilde{S}_f^{(0)}$.
       Approximate $V''(t_n, S_f),\ V'''(t_n, S_f)$ at $\tilde{S}_f^{(1)}$ to obtain
       $\tilde{V}_{S_f}''^{(1)} = V''(t_n, S_f) + \mathcal{O}(h^2),\ \tilde{V}_{S_f}'''^{(1)} = V'''(t_n, S_f) + \mathcal{O}(h)$.
  3:   Compute the FD corrections $\mathbf{a}_1$, $\mathbf{a}_2$ using $\tilde{S}_f^{(1)}$ and $V_{S_f}^*{}'' - \tilde{V}_{S_f}''^{(1)},\ V_{S_f}^*{}''' - \tilde{V}_{S_f}'''^{(1)}$.
       Solve (46) to obtain an approximate solution $\tilde{\mathbf{V}}^{(2)}$ of $\mathcal{O}(h^4)$ using penalty iteration with
       initial guess $\tilde{\mathbf{V}}^{(1)}$.
       Compute the approximate free boundary $\tilde{S}_f^{(2)}$ at $t_n$ to $\mathcal{O}(h^4)$ accuracy, using Newton's
       method with initial guess $\tilde{S}_f^{(1)}$.
       Approximate $V''(t_n, S_f), V'''(t_n, S_f), V''''(t_n, S_f)$ at $\tilde{S}_f^{(2)}$ to obtain
       $\tilde{V}_{S_f}''^{(2)} = V''(t_n, S_f) + \mathcal{O}(h^3), \tilde{V}_{S_f}'''^{(2)} = V'''(t_n, S_f) + \mathcal{O}(h^2), \tilde{V}_{S_f}''''^{(2)} = V''''(t_n, S_f) + \mathcal{O}(h)$.
  4:   Compute the FD corrections $\mathbf{a}_1$, $\mathbf{a}_2$ using $\tilde{S}_f^{(2)}$ and $V_{S_f}^*{}'' - \tilde{V}_{S_f}''^{(2)}, V_{S_f}^*{}''' - \tilde{V}_{S_f}'''^{(2)}$,
       $V_{S_f}^*{}'''' - \tilde{V}_{S_f}''''^{(2)}$.
       Solve (46) to obtain an approximate solution $\tilde{\mathbf{V}}^{(3)}$ using penalty iteration with initial guess
       $\tilde{\mathbf{V}}^{(2)}$.
       Compute the approximate free boundary $\tilde{S}_f^{(3)}$ at $t_n$, using Newton's method with initial
       guess $\tilde{S}_f^{(2)}$.
   end
```

5 Numerical results

In this section, we demonstrate the effectiveness of our high-order deferred correction algorithm for solving free boundary problems with several examples. We begin with a simple one-dimensional elliptic obstacle problem in Section 5.1. We then consider a time-dependent problem in Section 5.2. In this problem, the free boundary position $x_f(t)$ is singular at $t = 0$, but the solution itself is smooth everywhere. For both problems, we show that the convergence rate at each solve phase in Algorithm 1 and 2, respectively, is exactly as predicted. Finally, in Section 5.3, we apply our Algorithm 2 with some additional considerations to the

American option pricing problem. For this problem, both the free boundary and the solution itself is singular at $t = 0$. The numerical results show the expected convergence rate at each solve phase. Furthermore, our method appears to be more efficient than the state of the art for pricing American options to high accuracy.

5.1 Solving an elliptic obstacle problem

Consider the boundary value problem defined in the LCP form

$$\begin{aligned} -f'' + f + 1 &\geq 0, \\ f - f^* &\geq 0, \\ (-f'' + f + 1 = 0) \vee (f - f^* &= 0), \end{aligned} \tag{47}$$

on the domain $x \in [-1, 1]$, where $f^*(x) = x$, with boundary conditions

$$f(-1) = -1, \ f(1) = e - 1.$$

The exact solution to this problem is simply the piecewise smooth function

$$f(x) = \begin{cases} e^x - 1, & 0 < x \leq 1, \\ x, & -1 \leq x \leq 0. \end{cases}$$

It is obvious that the solution is smooth on both $[-1, 0]$ and $[0, 1]$, separately. At the point $x = 0$, the solution satisfies the value matching and smooth pasting conditions, that is,

$$\lim_{x \to 0^-} f(x) = \lim_{x \to 0^+} f(x) = f(0) = 0, \text{ and } \lim_{x \to 0^-} f'(x) = \lim_{x \to 0^+} f'(x) = f'(0) = 1.$$

However, the solution $f(x)$ has a discontinuous second derivative at $x = 0$, which means $f(x) \in C^1 \backslash C^2$. To apply Algorithm 4, we write (47) in the penalty form

$$-f'' + f + 1 - \rho \max(f^* - f, 0) = 0, \tag{48}$$

with Dirichlet boundary conditions at $x = -1$ and 1, where ρ is a penalty constant, taken to be $\rho = 1 \times 10^{12}$ in the numerical experiment.

We use this example to demonstrate that the four solve phases with deferred corrections in Algorithm 1 improve the convergence rates as expected. In Table 1, we can see that, away from the free boundary $x = 0$, the convergence orders at the first, second, third and fourth solves are 2, 3, 4 and 5, respectively. The free boundary approximation also follows the same successive increase of convergence order, as shown in Table 2. To demonstrate the computational efficiency of our algorithm, we have plotted the log-log graph of the solution errors versus the computational complexity, represented by grid size in space multiplied by the total number of penalty iterations, as shown in Figure 7. Note that in the first solve phase with no corrections, we only use a rough initial guess of a constant function $f = 1$, even though a better initial guess could be chosen. As a result, the first solve phase requires several penalty iterations to converge and takes up the major computational cost of the algorithm. In the second to fourth solve phases, only a single iteration is required for each solve phase, because the solution of the previous solve phase provides a good initial guess for Newton's method.

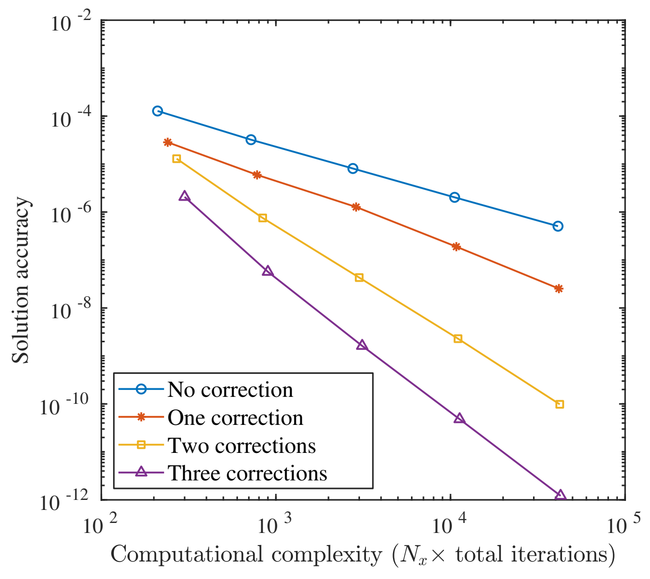
Remark 5.1. *Note that in both Tables 1 and 2, we see that fifth-order convergence is obtained after applying three corrections. The reason for this is that, even though we are using a fourth-order method, the error from the nonsmoothness of the solution at the free boundary has been reduced to $\mathcal{O}(h^5)$. Since this error is large compared to the $\mathcal{O}(h^4)$ error arising from the fourth-order finite difference scheme, the convergence appears to be fifth-order.*

N	$x = 0.2$							
	1st solve (no correction)				2nd solve (one correction)			
	niters	value	error	conv	niters	value	error	conv
30	7	0.221530668	1.28e-04	-	8	0.221431233	2.85e-05	-
60	12	0.221434987	3.22e-05	1.99	13	0.221396803	5.96e-06	2.26
120	23	0.221410846	8.09e-06	1.99	24	0.221401484	1.27e-06	2.22
240	44	0.221404784	2.03e-06	2.00	45	0.221402568	1.90e-07	2.75
480	86	0.221403265	5.07e-07	2.00	87	0.221402733	2.53e-08	2.91
N	3rd solve (two corrections)				4th solve (three corrections)			
	niters	value	error	conv	niters	value	error	conv
30	9	0.221415708	1.29e-05	-	10	0.221400689	2.07e-06	-
60	14	0.221403511	7.53e-07	4.10	15	0.221402701	5.68e-08	5.19
120	25	0.221402801	4.30e-08	4.13	26	0.221402757	1.62e-09	5.13
240	46	0.221402760	2.29e-09	4.23	47	0.221402758	4.77e-11	5.09
480	88	0.221402758	9.83e-11	4.54	89	0.221402758	1.06e-12	5.50

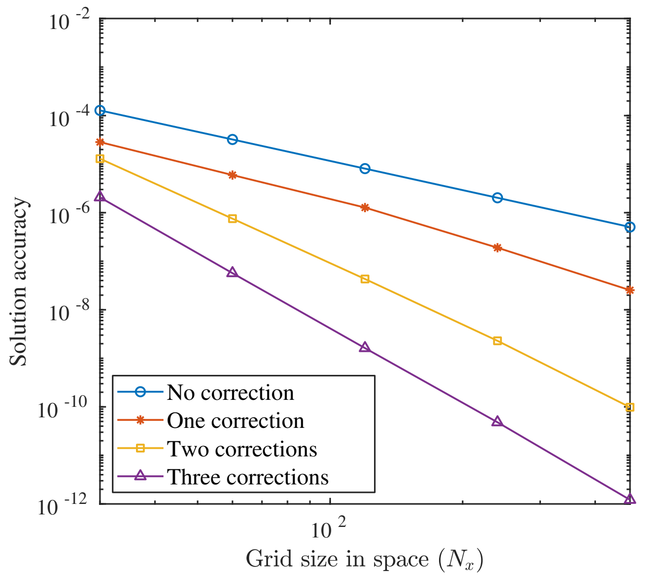
Table 1: The convergence results of solutions at point $x = 0.2$, of each solve phase in Algorithm 1 for solving the penalty equation (48) of a one-dimensional free boundary obstacle problem with free boundary at $x = 0$. Uniform grid spacing is used. Note that "niters" for the second to fourth solve includes the total number of iterations from all previous solve phases.

N	1st solve		2nd solve		3rd solve		4th solve	
	error	conv	error	conv	error	conv	error	conv
30	1.82e-02	-	2.20e-03	-	2.20e-04	-	1.87e-05	-
60	4.69e-03	1.96	2.51e-04	3.13	1.15e-05	4.26	4.12e-07	5.50
120	1.19e-03	1.98	2.84e-05	3.15	5.85e-07	4.30	6.55e-09	5.97
240	2.97e-04	2.00	2.93e-06	3.28	2.58e-08	4.50	8.38e-11	6.29
480	7.33e-05	2.02	2.17e-07	3.75	1.02e-09	4.67	1.03e-11	3.02

Table 2: The convergence results of the free boundary approximation of each solve phase in Algorithm 1 for solving the penalty equation (48) of a one-dimensional free boundary obstacle problem with the exact free boundary at $x = 0$. Uniform grid spacing is used.



(a)



(b)

Figure 7: The log-log plot of solution error at point $x = 0.2$ versus computational complexity (a) and grid size in space (b) using results of Table 1 for solving the penalty equation (48) of a one-dimensional free boundary obstacle problem. The computational complexity is represented by the grid size times the total number of penalty iterations.

5.2 A simple test free boundary problem

In this second example, we introduce the time variable and consider a time dependent free boundary problem. Consider the LCP

$$\begin{aligned} f_t - \frac{1}{2\sqrt{t}} f'' + \frac{1}{2\sqrt{t}} &\geq 0, \\ f - f^* &\geq 0, \\ \left(f_t - \frac{1}{2\sqrt{t}} f'' + \frac{1}{2\sqrt{t}} = 0 \right) &\vee (f - f^* = 0), \end{aligned} \tag{49}$$

on the domain $(t, x) \in [0, 0.5] \times [-2, 2]$, where $f^*(t, x) = x$. The solution satisfies the Dirichlet boundary conditions

$$f(t, -2) = -2, \;\; f(t, 2) = e^{2+\sqrt{t}} - \sqrt{t} - 1,$$

and the initial condition

$$f(0, x) = \begin{cases} e^x - 1, & 0 \leq x \leq 2, \\ x, & -2 \leq x < 0. \end{cases}$$

The exact solution to (49) is

$$f(t,x) = \begin{cases} e^{x+\sqrt{t}} - \sqrt{t} - 1, & x_f(t) \leq x \leq 2, \\ x, & -2 \leq x < x_f(t), \end{cases}$$

where $x_f(t)$ is the moving free boundary

$$x_f(t) = -\sqrt{t}.$$

The value matching and smooth pasting conditions at the free boundary $x = x_f(t)$ follow naturally. Again, we see that $f(\cdot, x) \in C^1 \backslash C^2$ on $[-2, 2]$, but it is smooth on $[-2, x_f(t)]$ and $[x_f(t), 2]$, separately. To apply Algorithm 2, we write (49) in penalty form

$$f_t - \frac{1}{2\sqrt{t}} f'' + \frac{1}{2\sqrt{t}} + \rho \max(f^* - f, 0) = 0, \tag{50}$$

where ρ is a penalty constant, taken to be $\rho = 1 \times 10^8$ in the numerical experiment.

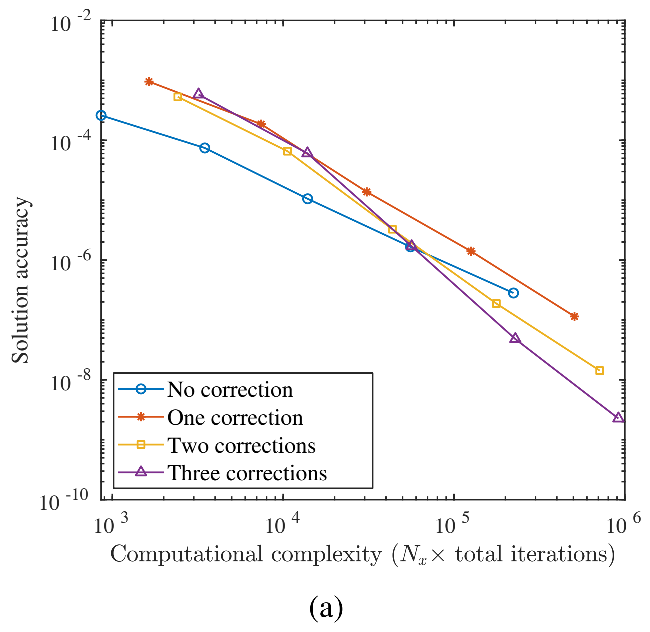
Since the free boundary $x_f(t) = -\sqrt{t}$, its location changes rapidly near time $t = 0$. This will cause a problem in the BDF4 time-stepping scheme because many grid points will cross the free boundary in the initial time steps (see Section 3.4). Hence, BDF4 degenerates to only first-order convergence due to piecewise smoothness in the solution across the free boundary. To avoid this situation, we perform a time-variable transformation $t = \tau^2$ so that the free boundary changes more slowly, and fewer points will cross the free boundary in the initial time steps. Although this does not completely solve the problem, it is accurate enough for the algorithm to achieve high-order convergence, as shown in the numerical results. To start BDF4, we use the exact solutions for the first three time steps. For this problem, we simply use a uniform grid in space.

In Table 3, we record the convergence results at $x = -0.37$ and at $x = 0$. The point $x = -0.37$ is slightly to the right of the first grid point right of the final-time free boundary location on the coarsest grid $N_x = 20$. The point $x = 0$ is the initial free boundary location. We see that the solutions at both points gain the expected order of convergence after each correction. To demonstrate the computational efficiency of our algorithm, we plot the log-log graph of the solution errors versus the computational complexity represented by the grid size in space multiplied by the total number of penalty iterations, as shown in Figure 8.

Remark 5.2. *Note that the solutions after solving with corrections have larger errors on the coarsest grid $N_x = 20$. This is due to large extrapolation errors of free boundary and derivatives approximations when the grid size near the free boundary is large, which occurs on a uniform coarse grid. This can be avoided by applying grid stretching around the free boundary.*

(N_x, N_t)	$x=-0.37$							
	1st solve (no correction)				2nd solve (one correction)			
	niters	value	error	conv	niters	value	error	conv
(20,40)	43	-0.307227	1.04e-03	-	82	-0.304930	1.26e-03	-
(40,80)	87	-0.306241	8.16e-06	7.00	186	-0.306075	1.47e-04	3.10
(80,160)	174	-0.306227	9.13e-06	-0.16	386	-0.306205	1.33e-05	3.46
(160,320)	348	-0.306220	1.94e-06	2.24	786	-0.306216	1.67e-06	3.00
(320,640)	697	-0.306218	3.19e-07	2.60	1586	-0.306218	1.44e-07	3.53
(N_x, N_t)	3rd solve (two corrections)				4th solve (three corrections)			
	niters	value	error	conv	niters	value	error	conv
(20,40)	121	-0.306334	1.61e-04	-	160	-0.305883	3.03e-04	-
(40,80)	265	-0.306147	7.44e-05	1.11	347	-0.306163	5.90e-05	2.36
(80,160)	545	-0.306215	3.35e-06	4.47	707	-0.306216	1.77e-06	5.06
(160,320)	1106	-0.306218	2.20e-07	3.93	1429	-0.306218	5.78e-08	4.94
(320,640)	2227	-0.306218	1.12e-08	4.29	2866	-0.306218	2.79e-09	4.37
(N_x, N_t)	$x=0$							
	1st solve (no correction)				2nd solve (one correction)			
	niters	value	error	conv	niters	value	error	conv
(20,40)	43	0.320748	2.60e-04	-	82	0.321960	9.52e-04	-
(40,80)	87	0.320934	7.43e-05	1.81	186	0.321194	1.85e-04	2.36
(80,160)	174	0.320998	1.05e-05	2.82	386	0.321022	1.37e-05	3.76
(160,320)	348	0.321007	1.66e-06	2.66	786	0.321010	1.40e-06	3.29
(320,640)	697	0.321008	2.82e-07	2.56	1586	0.321008	1.14e-07	3.62
(N_x, N_t)	3rd solve (two corrections)				4th solve (three corrections)			
	niters	value	error	conv	niters	value	error	conv
(20,40)	121	0.321537	5.29e-04	-	160	0.321588	5.80e-04	-
(40,80)	265	0.321074	6.56e-05	3.01	347	0.321069	6.04e-05	3.26
(80,160)	545	0.321011	3.27e-06	4.33	707	0.321010	1.70e-06	5.15
(160,320)	1106	0.321008	1.88e-07	4.12	1429	0.321008	4.80e-08	5.15
(320,640)	2227	0.321008	1.43e-08	3.72	2866	0.321008	2.27e-09	4.41

Table 3: The convergence results of solutions at points $x = -0.37$ and $x = 0$ of each solve phase in Algorithm 2 for solving the penalty equation (50) of a moving boundary problem with the exact moving boundary $x_f(t) = -\sqrt{t}$. Note that "niters" for the second to fourth solve includes the total number of iterations from all previous solve phases.



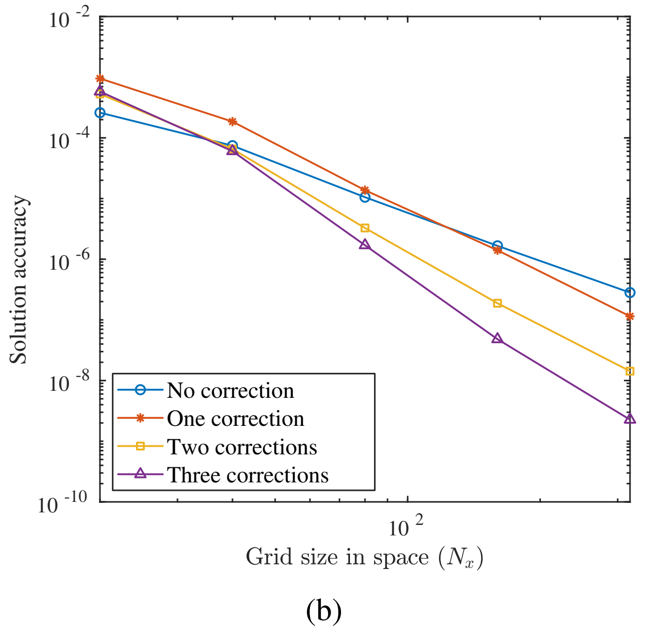


Figure 8: The log-log plot of solution errors at point $x = 0$ versus computational complexity (a) and grid size in space (b) using results of Table 3 for solving the penalty equation (50) of a moving boundary problem with the exact moving boundary $x_f(t) = -\sqrt{t}$. The computational complexity is represented by the grid size times the total number of penalty iterations.

5.3 American option pricing

Finally, we use our algorithm to solve the American put option pricing problem. We repeat the penalty equation

$$\partial_t V = \mathcal{L}_{BS}V + \rho \max\{V^* - V, 0\},$$

for convenience of discussion, where $V^*(S) = \max\{K - S, 0\}$ is the payoff of the American put option struck at K. The initial condition is

$$V(0, S) = V^*(S).$$

We truncate the right end of the domain at $S = S_{\max}$ and use Dirichlet boundary conditions $V(t, 0) = K$ and $V(t, S_{\max}) = 0$. To avoid complications due to the payoff singularity at the strike price of the American put options, we compute the difference between an American option and a European option, as in [27], where this is referred to as the singularity-separating method. A European put option with the same volatility σ, bank interest r, dividend q, and strike price K satisfies the Black-Scholes equation

$$\partial_t V^E = \mathcal{L}_{BS}V^E, \quad \text{where } \mathcal{L}_{BS} = \frac{\sigma^2 S^2}{2}\frac{\partial^2}{\partial S^2} + (r - d)S\frac{\partial}{\partial S} - r,$$

and the initial condition $V^E(0,S) = \max\{K-S,0\}$. The difference of their solutions $V^{\text{diff}} = V - V^E$ has a zero initial condition. The analytical solution of the European put option price is

$$V^E(t,S) = Ke^{-rt}\Phi(-d_-) - Se^{-dt}\Phi(-d_+),$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function. Therefore, instead of solving for the original American option price, we solve for V^{diff}, which satisfies the equation

$$\partial_t V^{\text{diff}} = \mathcal{L}_{BS} V^{\text{diff}} + \rho \max\{(V^* - V^E) - V^{\text{diff}}, 0\},$$

with a zero initial condition, and then add the European option price back to obtain the final American option price. The penalty constant ρ is chosen to be $\rho = 1 \times 10^8$ in the numerical experiment.

In this problem, we do not have the exact solution. Since BDF4 requires solutions from the previous four time steps to proceed, we need to be careful when starting BDF4. An inapproprate starting mechanism will likely cause the high order convergence of BDF4 to degenerate. For this problem, we compute the first time step using the fourth-order Runge-Kutta method [6]. For the second time step, we use a three-level fourth-order method

$$\left(\mathbf{I} - \frac{k}{3}\mathbf{L}\right)\tilde{\mathbf{V}}^{n+2} = \tilde{\mathbf{V}}^n + \frac{k}{3}\mathbf{L}\left(4\tilde{\mathbf{V}}^{n+1} + \tilde{\mathbf{V}}^n\right) + \frac{k}{3}(\mathbf{b}^{n+2} + 4\mathbf{b}^{n+1} + \mathbf{b}^n) + 2k\mathbf{q}(\tilde{\mathbf{V}}^{n+2}),$$

see [15], and for the third time step, we simply appy BDF3. We observe that this starting scheme is sufficient for fourth-order convergence.

We test the algorithm on two example problems with different volatilities, $\sigma = 0.2$ and $\sigma = 0.8$. Both examples have the other parameters the same: zero dividend payment, interest rate $r = 0.1$ and strike price $K = 100$. We truncate the infinite domain at $S_{\max} = 10K = 1000$ for the problem with smaller volatility $\sigma = 0.2$, and at $S_{\max} = 13K = 1300$ for the larger volatility $\sigma = 0.8$. As it turns out, a larger $S_{\max}$ is not only necessary for the accuracy of solution with a larger volatility, it is also important for observing the convergence results of our algorithm. To implement the algorithm, we use the time variable transformation $t = \tau^3$ for $\sigma = 0.2$ and $t = \tau^2$ for $\sigma = 0.8$. We propose the stretching function

$$\xi(S) = \left(S - \frac{\sqrt{\pi}}{2}\frac{1-\beta}{\beta}\alpha \operatorname{erfc}\left(\frac{S-K}{\alpha}\right)\right)C_1 + C_2,$$

to stretch the space grid, where

$$C_1 = 1/\left[(S_{\max} - S_{\min}) - \frac{\sqrt{\pi}}{2}\frac{1-\beta}{\beta}\alpha\left(\text{erfc}\left(\frac{S_{\max}-K}{\alpha}\right) - \text{erfc}\left(\frac{S_{\min}-K}{\alpha}\right)\right)\right],$$
$$C_2 = \left[\frac{\sqrt{\pi}}{2}\frac{1-\beta}{\beta}\alpha \operatorname{erfc}\left(\frac{S_{\min}-K}{\alpha}\right) - S_{\min}\right]C_1.$$

where α and β are parameters controlling the density of the stretching, and chosen to be $\alpha = 20.83$ and $\beta = 1/20$ for $\sigma = 0.2$, and $\alpha = 49.83$, $\beta = 1/13$ for $\sigma = 0.8$, in the numerical experiments. This function adds additional grid points with density $1/\beta$, on a region of width 6α around the point K, while maintaining the density of 1 elsewhere.

In addition, since the solution of the American option price is singular at the strike at expiry, meaning that the solution is not smooth, we do not apply the correction scheme in Algorithm 2 for the first several time steps. In the numerical experiments, the number of time steps skipped is chosen to be $t_{skip} = 20$ for $\sigma = 0.2$ and $t_{skip} = 10$ for $\sigma = 0.8$. Since we apply time and space stretching near the strike at expiry, the errors from the skipped corrections in the first few time steps are sufficiently small as to not affect the high-order convergence.

In Table 4, we can see clear convergence rate improvements and error reduction with corrections at each solve phase, for the smaller volatility $\sigma = 0.2$. The second solve phase, corresponding to one correction, with first correction only slightly reduces the error. The third- and fourth-solve phases, corresponding to two and three corrections, respectively, reduce the error significantly. The final results after the fourth solve phase exhibit a reduction of error by nearly 100 times. Moreover, the convergence rates are consistently above fourth-order.

To demonstrate the computational efficiency of our algorithm, we have shown in Figure 9 the solution accuracy versus the computational complexity represented by the grid size in space multiplied with the total number of penalty iterations. We can see that the three-correction algorithm is slightly more expensive if high accuracy is not the goal. However, when a high accuracy solution is desired, the three-correction algorithm is more efficient.

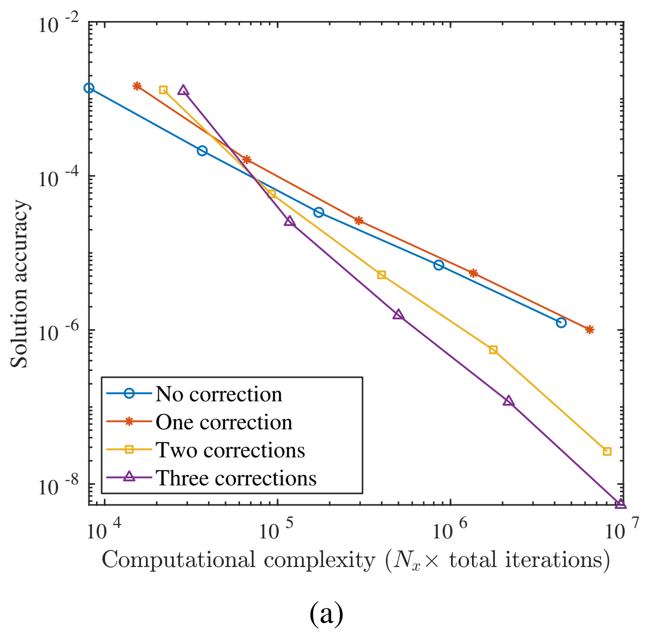
Remark 5.3. *For this example, we see that the second solve does not improve the convergence much. This is because we have applied enough stretching to reduce the leading-order error term due to the second-derivative jump in the first solve, even without correction. Moreover, we observe that our algorithm for $\sigma = 0.2$ performs better than for $\sigma = 0.8$. For a larger volatility $\sigma = 0.8$, the optimal exercise boundary moves more quickly and ranges over a larger part of the domain within the same time span. Since we only apply stretching in space around the initial free boundary, when the free boundary moves out of the stretched area, the extrapolation error when approximating the free boundary and derivatives becomes more dominant due to larger grid spacing. This explains the increase in error from applying the first correction. One way to avoid large extrapolation errors is to implement a time-dependent grid stretching that follows the free boundary movement, see e.g. [18] where a predictor-corrector scheme is applied. We leave this to future work.*

6 Conclusions

In this paper, we presented a high-order deferred correction algorithm using fourth-order finite difference method and the BDF4 time stepping scheme for solving free and moving boundary problems. Our algorithm utilizes the penalty method and assumes no prior knowledge of the exact free boundary location and derivative jumps at the free boundary. Our method does not modify the finite difference stencils and the arising matrix, but applies the corrections to the right-hand side. The penalty iteration converges in a few iterations. From the analysis of the error behaviors when solving free boundary problems, we showed that our deferred correction algorithm can successively increase the solution order of convergence from $\mathcal{O}(h^2)$ to $\mathcal{O}(h^3)$, and from $\mathcal{O}(h^3)$ to $\mathcal{O}(h^4)$ after applying each successive correction. Our numerical results

(N_x, N_t)	$\sigma = 0.2,\ T = 0.25$							
	1st solve (no correction)				2nd solve (one correction)			
	niters	value	error	conv	niters	value	error	conv
(53,30)	37	3.068467970	-	-	72	3.068445580	-	-
(104,60)	78	3.069852898	1.38e-03	-	148	3.069911010	1.47e-03	-
(206,120)	178	3.070064665	2.12e-04	2.71	322	3.070073785	1.63e-04	3.17
(410,240)	422	3.070098279	3.36e-05	2.66	720	3.070100037	2.63e-05	2.63
(818,480)	1046	3.070105234	6.96e-06	2.27	1658	3.070105494	5.46e-06	2.27
1635,960)	2675	3.070106475	1.24e-06	2.49	3909	3.070106503	1.01e-06	2.43
(N_x, N_t)	3rd solve (two corrections)				4th solve (three corrections)			
	niters	value	error	conv	niters	value	error	conv
(53,30)	106	3.068734387	-	-	140	3.068818541	-	-
(104,60)	210	3.070042641	1.31e-03	-	274	3.070079883	1.26e-03	-
(206,120)	448	3.070100955	5.83e-05	4.49	571	3.070105056	2.52e-05	5.65
(410,240)	974	3.070106149	5.19e-06	3.49	1221	3.070106603	1.55e-06	4.02
(818,480)	2162	3.070106702	5.53e-07	3.23	2655	3.070106720	1.17e-07	3.72
(1635,960)	4930	3.070106729	2.65e-08	4.38	5907	3.070106725	5.35e-09	4.45
(N_x, N_t)	$\sigma = 0.8,\ T = 0.25$							
	1st solve (no correction)				2nd solve (one correction)			
	niters	value	error	conv	niters	value	error	conv
(50,30)	34	14.676127404	-	-	64	14.675281018	-	-
(98,60)	75	14.678320134	2.19e-03	-	135	14.677946769	2.67e-03	-
(195,120)	158	14.678780547	4.60e-04	2.25	284	14.678664938	7.18e-04	1.89
(388,240)	366	14.678861193	8.06e-05	2.51	633	14.678833363	1.68e-04	2.09
(775,480)	902	14.678875150	1.40e-05	2.53	1471	14.678870242	3.69e-05	2.19
(1548,960)	2258	14.678877820	2.67e-06	2.39	3452	14.678877049	6.81e-06	2.44
(N_x, N_t)	3rd solve (two corrections)				4th solve (three corrections)			
	niters	value	error	conv	niters	value	error	conv
(50,30)	94	14.676826997	-	-	126	14.677702973	-	-
(98,60)	202	14.678464702	1.64e-03	-	266	14.678691419	9.88e-04	-
(195,120)	428	14.678822500	3.58e-04	2.19	553	14.678868177	1.77e-04	2.48
(388,240)	919	14.678870119	4.76e-05	2.91	1163	14.678877514	9.34e-06	4.24
(775,480)	2033	14.678877395	7.28e-06	2.71	2520	14.678878299	7.85e-07	3.57
(1548,960)	4534	14.678878259	8.64e-07	3.07	5502	14.678878361	6.25e-08	3.65

Table 4: The convergence results of an American put option at $S = 100$, $T = 0.25$ with $K = 100,\ r = 0.1,\ q = 0$ for $\sigma = 0.2$ and $\sigma = 0.8$. Note that "niters" for the second to fourth solve includes the total number of iterations from all previous solve phases.



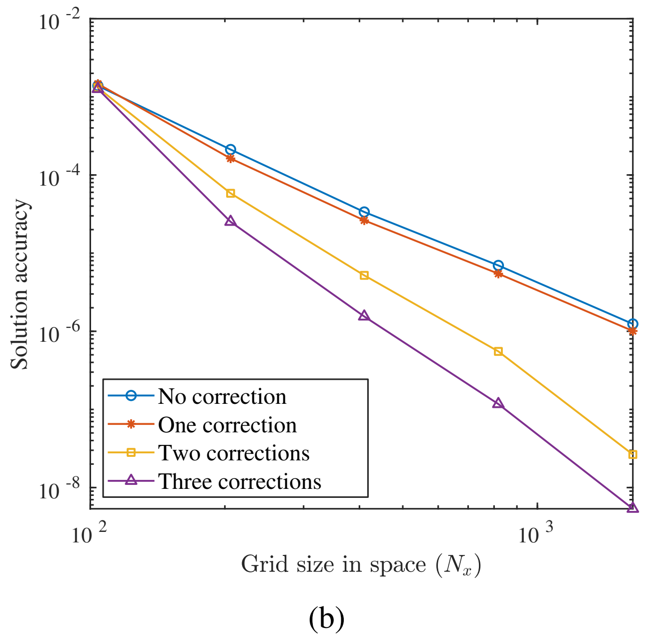


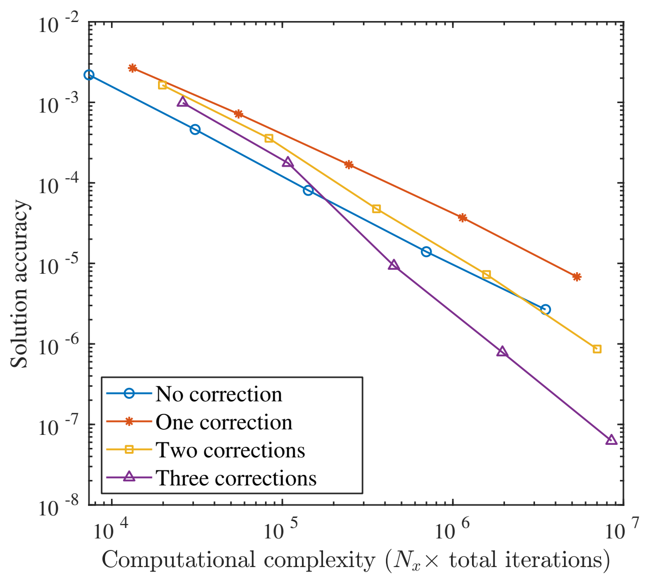
Figure 9: The log-log plot of solution changes at the strike point K versus computational complexity (a) and grid size in space (b) at the final time using results of solving American option prices in Table 4 for $\sigma = 0.2$. The computational complexity is represented by the grid size in space times the total number of penalty iterations.

validate the theoretical analysis. On simple test problems when the solution is not singular, results show that the behavior of our algorithm matches exactly with our theory. When solving the more challenging American put option problem, our algorithm also performs well.

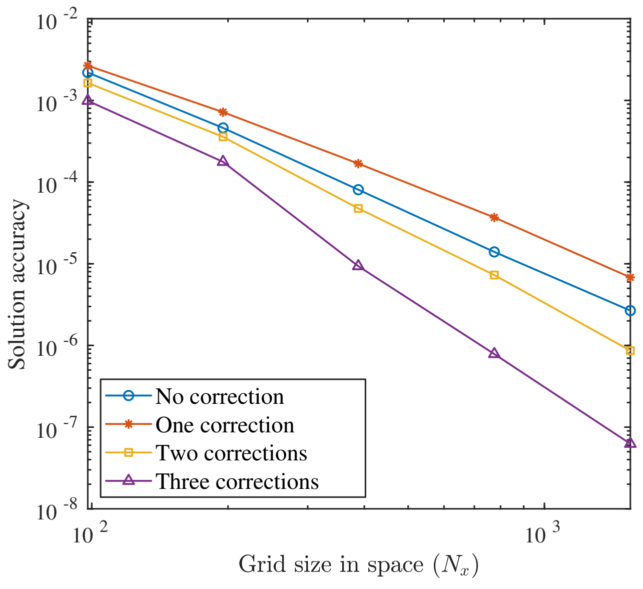
6.1 Generalizations and future work

We only considered problems with one space dimension in this work, however, the deferred correction idea can be generalized to two space dimensions. One possible extension is to the elliptic obstacle problem in two dimensions, which is still an active area of research. The major difference in the algorithm when applied to two-dimensional problems is how the extrapolation scheme is designed. Two dimensional extrapolation has already been extensively studied in the literature (see, for example, [12, 17, 24]). To reduce the extrapolation error on a uniform grid, we can also refine the grid around the free boundary, the location of which can be approximated on a coarser grid.

Since grid stretching in space is useful for reducing the extrapolation errors, another possible extension of our work is to apply a time-dependent grid stretching scheme when solving moving boundary problems. As noted in Section 5, our algorithm applies grid stretching only around the initial free boundary, which can cause more extrapolation errors in later time steps when the free boundary moves out of the stretched area. Hence, it does not achieve ideal performance, as seen in the application to the American option pricing problem when $\sigma = 0.8$. By adapting the grid stretching so that it follows the moving boundary, the extrapolation error can be reduced at all time steps. One possible way of doing this is to use the predictor-corrector idea, as discussed in [18]. This scheme precomputes an approximate solution, which



(a)



(b)

Figure 10: The log-log plot of computational complexity and grid size in space versus solution changes at the strike point K at the final time using results of solving American option prices in Table 4 for $\sigma = 0.8$. The computational complexity is represented by the grid size in space times the total number of penalty iterations.

then gives an approximate moving boundary, on a coarse grid. When we solve the problem on a finer grid at each time step, we can apply grid stretching around the precomputed approximate free boundary. By doing this, we expect our algorithm to achieve even better results.

References

[1] D. M. Anderson, G. B. McFadden, and A. A. Wheeler, *Diffuse-interface methods in fluid mechanics*, Annual Review of Fluid Mechanics, 30 (1998), pp. 139–165.

[2] N. Clarke and K. Parrott, *Multigrid for American option pricing with stochastic volatility*, Applied Mathematical Finance, 6 (1999), pp. 177–195.

[3] J. Crank and J. Crank, *Free and moving boundary problems*, Oxford University Press, USA, 1984.

[4] J. N. Dewynne, S. D. Howison, I. Rupf, and P. Wilmott, *Some mathematical results in the pricing of American options*, European Journal of Applied Mathematics, 4 (1993), pp. 381–398.

[5] M. J. Dilloo and D. Y. Tangman, *A high-order finite difference method for option valuation*, Computers & Mathematics with Applications, 74 (2017), pp. 652–670.

[6] J. R. Dormand and P. J. Prince, *A family of embedded Runge-Kutta formulae*, Journal of Computational and Applied Mathematics, 6 (1980), pp. 19–26.

[7] R. P. FEDKIW, T. ASLAM, B. MERRIMAN, AND S. OSHER, *A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)*, Journal of Computational Physics, 152 (1999), pp. 457–492.

[8] B. FORNBERG, *Classroom note: Calculation of weights in finite difference formulas*, SIAM Review, 40 (1998), pp. 685–691.

[9] B. FORNBERG AND R. MEYER-SPASCHE, *A finite difference procedure for a class of free boundary problems*, Journal of Computational Physics, 102 (1992), pp. 72–77.

[10] P. A. FORSYTH AND K. R. VETZAL, *Quadratic convergence for valuing American options using a penalty method*, SIAM Journal on Scientific Computing, 23 (2002), pp. 2095–2122.

[11] A. FRIEDMAN, *Variational principles and free-boundary problems*, A Wiley-Interscience Publication, John Wiley & Sons, Inc., New York, 1982.

[12] F. GIBOU AND R. FEDKIW, *A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem*, Journal of Computational Physics, 202 (2005), pp. 577–601.

[13] F. GIBOU, R. P. FEDKIW, L.-T. CHENG, AND M. KANG, *A second-order-accurate symmetric discretization of the poisson equation on irregular domains*, Journal of Computational Physics, 176 (2002), pp. 205–227.

[14] R. J. LEVEQUE AND Z. LI, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM Journal on Numerical Analysis, 31 (1994), pp. 1019–1044.

[15] M. LI AND T. TANG, *A compact fourth-order finite difference scheme for unsteady viscous incompressible flows*, Journal of scientific computing, 16 (2001), pp. 29–45.

[16] Z. LI, *A fast iterative algorithm for elliptic interface problems*, SIAM Journal on Numerical Analysis, 35 (1998), pp. 230–254.

[17] M. N. LINNICK AND H. F. FASEL, *A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains*, Journal of Computational Physics, 204 (2005), pp. 157–192.

[18] C. W. OOSTERLEE, C. C. LEENTVAAR, AND X. HUANG, *Accurate American option pricing by grid stretching and high order finite differences*, Delft University of Technology, The Netherlands, Technical Report, (2005).

[19] C. S. PESKIN, *Flow patterns around heart valves: a numerical method*, Journal of Computational Physics, 10 (1972), pp. 252–271.

[20] J.-F. RODRIGUES, *Obstacle problems in mathematical physics*, Elsevier, 1987.

[21] J. A. SETHIAN AND P. SMEREKA, *Level set methods for fluid interfaces*, Annual review of fluid mechanics, 35 (2003), pp. 341–372.

[22] A. SHARMA AND R. RANGARAJAN, *A shape optimization approach for simulating contact of elastic membranes with rigid obstacles*, International Journal for Numerical Methods in Engineering, 117 (2019), pp. 371–404.

[23] D. TAVELLA AND C. RANDALL, *Pricing financial instruments: The finite difference method*, vol. 13, John Wiley & Sons, 2000.

[24] A. WIEGMANN AND K. P. BUBE, *The explicit-jump immersed interface method: finite difference methods for PDEs with piecewise smooth solutions*, SIAM Journal on Numerical Analysis, 37 (2000), pp. 827–862.

[25] P. WILMOTT, S. HOWSON, S. HOWISON, J. DEWYNNE, ET AL., *The mathematics of financial derivatives: a student introduction*, Cambridge university press, 1995.

[26] L. WU AND Y.-K. KWOK, *A front-fixing finite difference method for the valuation of American options*, Journal of Financial Engineering, 6 (1997), pp. 83–97.

[27] Y. ZHU, X. WU, I.-L. CHERN, AND Z.-Z. SUN, *Derivative securities and difference methods*, Springer, 2004.