High Performance American Option Pricing

Leif Andersen*, Mark Lake and Dimitri Offengenden

Bank of America

First Version: June, 2013. This Version[1]: January, 2020.

Abstract

We develop a new high-performance spectral collocation method for the computation of American put and call option prices. The proposed algorithm involves a carefully posed Jacobi-Newton iteration for the optimal exercise boundary, aided by Gauss-Legendre quadrature and Chebyshev polynomial interpolation on a certain transformation of the boundary. The resulting scheme is straightforward to implement and converges at a speed several orders of magnitude faster than existing approaches. Computational effort depends on required accuracy; at precision levels similar to, say, those computed by finite difference grid with several hundred steps, the computational throughput of the algorithm in the Black-Scholes model is typically close to 100,000 option prices per second per CPU. For benchmarking purposes, Black-Scholes American option prices can generally be computed to at 10 or 11 significant digits in less than one-tenth of a second.

1 Introduction

American put and call options trade on a large number of exchanges worldwide, and cover many different asset classes, such as FX, commodities, and equities. The work undertaken in this paper was originally motivated by the practical problem of computing real-time risk for large portfolios of such options, especially the popular type of contract where the exercise value references a futures contract[2]. Such applications often require

*One Bryant Park, New York, NY 10036, USA. E-mail: leif.andersen@bofa.com. The authors are grateful for comments received from their colleagues and from participants at the January 2014 Aarhus University Quant Week, the January 2014 Bloomberg Quant Seminar, the March 2014 Python Quants Conference, and the July 2014 Risk Magazine Quant Congress USA.

[1]Since the original publication of this paper in the *Journal of Computational Finance*, we have received comments from many people that implemented our numerical scheme in real life. In this 2020 version, we have incorporated some of these comments, including those recently offered to us by Mike Staunton and Jesper Andreasen, both of whom we thank. Changes in the 2020 update of our paper include the correction of a few minor typos; improvement of notation and flow in our exposition of Chebyshev interpolation; and some additional guidance on proper usage of the square-root time transformation for the final option pricing integral. We point the reader to a forthcoming publication [67] by Mike Staunton which contains a complete VBA implementation of the algorithm in this paper.

[2]We are here primarily interested in the "true" American futures options traded in the US (CME, CBOT) and in Asia (SGC, TSE). The "American" options traded in Europe (LIFFE, EUREX, NLX) are equipped with a margining mechanism that effectively simplifies them to European options.

thousands, if not tens of thousands, of option value computations per "tick" of the clock, so at their core necessarily lies a very fast method for the computation of American option premia.

Although more sophisticated models have been proposed over the years, the current market practices for the pricing and relative-value analysis of listed American options[3] still revolve almost exclusively around the standard Black-Scholes model. Robust and well-tested, this model has proven its mettle over decades, and still offers sufficient flexibility for market participants to quote, position themselves, and risk-manage effectively. Indeed, as is the case for European options, quotation standards for exchange-traded American commonly revolve around the so-called Black-Scholes *implied volatility*.

To calculate American option prices (or implied volatility, for a known option price) in the Black-Scholes model, it is, unfortunately, necessary to rely on numerical methods, as no true closed-form American option price exists. For speed reasons, it is not uncommon to apply semi-analytical approximation methods for American options, such as those in the works of [12], [22], [50], and many[4] more. However, while typically fast, these methods by their very nature involve inaccuracies that, even for the best algorithms, can be highly significant. Moreover, there is generally no possibility of using these approximations to make an intelligent, and application-specific, trade-off between speed and accuracy. That is, there is no way one can systematically increase precision of a computed price by spending more computational resources on the problem.

There are, of course, many well-known methods that *can* produce American option prices to very high precision. A partial list includes binomial trees (see [35]), "accelerated" binomial trees (e.g., [57] and [47]), finite difference methods (see, e.g., [17] and [38]), the method of lines ([24]), least-squares Monte Carlo methods (e.g., [61] and [5]), and numerical integral equation methods ([68], and many others). Of those, Monte Carlo methods are no doubt the most flexible but also the slowest and noisiest; they are therefor of little relevance to our specific application where speed and accuracy, rather than flexibility, is at a premium. Of the remaining methods, it is fair to say that the integral equation methods have not fared particularly well in head-to-head comparisons (see, e.g., [2]) and are reputed to be slow[5] – but this is nevertheless the approach that we shall pursue here. We are motivated by excellent results in a fixed income setting (see [7] and [4]) as well as by recent papers (e.g., [53] and [33]) that show promising speed-accuracy performance for the integral equation approach.

In this paper, we refine earlier work in a number of ways, to achieve levels of speed and accuracy far better than previous methods. Our approach is based on a spectral collocation method on a carefully transformed integral equation for the optimal exer-

[3] Many OTC American put and call options are, in fact, also priced on the Black-Scholes models, often with special-purpose grids for the "American" implied volatility. This, for instance, is common practice in equity markets, even for relatively long-dated puts and calls.

[4] A more complete list of papers on American option approximation methods can be found in Section 4.5.

[5] For instance, in [27] the integral equation method advocated by the authors takes in the order of minutes to solve on a Sun Sparc server.

cise boundary. The boundary is found by a modified Jacobi-Newton function iteration, starting from an approximate guess. When run on a single run-of-the-mill 2GHz CPU, the numerical scheme in this paper will, without much optimization, produce accurate pricing precision at the level of a finite difference grid solver with a few hundred steps (or a binomial tree with thousands of steps) at the rate of around 100,000 option prices per second, per CPU. If our goal is extreme precision, rather than speed, one hundredth of second of computation time with our approach can produce prices that are beyond the practical reach of a finite difference grid, even when run with 100,000's (or even 1,000,000's) of steps.

The rest of this paper is organized as follows. In Section 2, we state process and payout assumptions, and outline a series of basic results for American put options and their optimal exercise boundary. In particular, we list a series of equivalent integral equations for the optimal exercise boundary. Section 3 surveys known numerical methods for the boundary computations and provides a general analysis of iterative methods for the exercise boundary. In Section 4 this analysis is supplemented by a series of relevant theoretical results for the shape and asymptotics of the exercise boundary. Section 4 also discusses certain boundary approximation schemes, some of which are new. Armed with the material of Sections 3 and 4, we proceed to develop in detail our new numerical scheme in Section 5. Numerical tests of the scheme are provided in Section 6, and Section 7 outlines a variety of extensions of the method to different option payouts and to different stochastic processes for the underlying asset. Finally, Section 8 concludes the paper.

2 Model Setup

2.1 Process Definition

Consider a financial asset exhibiting constant log-normal volatility at an annualized rate of σ. Let r be a constant risk-free interest rate, and let $\beta(t) = \exp(rt)$ be the rolling money market numeraire. In the risk-neutral measure $\mathbb{Q}$ induced by β, let the asset process be given by a GBM SDE of the form

$$dS(t)/S(t) = \mu\, dt + \sigma\, dW(t), \tag{1}$$

where $W(t)$ is a $\mathbb{Q}$-Brownian motion. The constant drift μ in (1) is asset specific and given by arbitrage considerations. For instance, if S is a stock paying dividends at a continuous rate of q, we have $\mu = r - q$. Alternatively, if S represents a futures (or forward) price, we have $\mu = 0$. The case $\mu = 0$ is of particular practical importance and originally motivated most of the work in this paper, but for generality we work with the case $\mu = r - q$ going forward. Extensions to time-dependent parameters are possible, and are discussed in Section 7.

2.2 The American Put and its Price Formula

Within the setting of Section 2.1, we shall focus on a K-strike *American put option*, paying $(K - S(\nu))^+$ if exercised at time $\nu \in [0, T]$, with T being the terminal maturity of the put. Note that American call option prices can always be inferred from put prices through put-call symmetry (e.g., [62]). If not previously exercised, the time t no-arbitrage value of the put is given by

$$p(t) = \sup_{\nu \in [t,T]} \mathrm{E}_t \left(e^{-r(\nu - t)} \left(K - S(\nu) \right)^+ \right) \tag{2}$$

where $\mathrm{E}_t(\cdot)$ denotes time t expectation in $\mathbb{Q}$, and where the supremum is taken over all stopping times on $[t, T]$. For the T-maturity American put on an asset following (1), it is known that the decision to exercise is characterized by a deterministic *exercise boundary* $S_T^*(t)$, in the sense that the optimal exercise policy ν^* (as seen at time 0) may be written

$$\nu^* = \inf \left(t \in [0, T] : S(t) \leq S_T^*(t) \right). \tag{3}$$

We emphasize that the exercise boundary $S_T^*(t)$ for the American put is indexed by option maturity T, as the decision to exercise at time t obviously depends on how much time is left before the option matures. For the constant-parameter process (1), however, it is easily seen that $S_T^*(t) = B(T-t)$ for some (time-reversed) boundary function $B : \mathbb{R} \to \mathbb{R}$ satisfying $B(0) = K$ and (see Section 4.1) $B(0+) = K \min(1, r/q)$.

By well-known arguments (see [52], [44], and [45], among several others), the exercise boundary, if known, may be used to compute the American put price through an integral equation. Specifically, if we let $V(T - t, S)$ be the time t price[6] of the T-maturity American put when $S(t) = S$, then, for $\tau = T - t$,

$$V(\tau, S) = v(\tau, S) + \int_0^\tau rKe^{-r(\tau - u)} \Phi \left(-d_- \left(\tau - u, S/B(u) \right) \right) du \\ - \int_0^\tau qSe^{-q(\tau - u)} \Phi \left(-d_+ \left(\tau - u, S/B(u) \right) \right) du \tag{4}$$

where $v(\tau, S)$ is the European put option price, $\Phi(\cdot)$ is the cumulative Gaussian distribution function, and

$$d_\pm(\tau, z) \triangleq \frac{\ln z + (r - q)\tau \pm \frac{1}{2}\sigma^2 \tau}{\sigma \sqrt{\tau}}.$$

In (4), by the standard Black-Scholes result, we have

$$v(\tau, S) = e^{-r\tau} K \Phi \left(-d_-(\tau, S/K) \right) - Se^{-qt} \Phi \left(-d_+(\tau, S/K) \right). \tag{5}$$

Let us briefly discuss the intuition behind (4). First, transforming the integration variable back to calendar time we notice that (with $\tau - u = s - t$, or $u = T - s$)

$$V(\tau, S) = v(\tau, S) + \int_t^T \mathrm{E} \left[e^{-r(s-t)} \left(rK - qS(s) \right) \cdot 1_{S(s) < S_T^*(s)} \, ds | S(t) = S \right]. \tag{6}$$

[6] Such that $p(t) = V(T - t, S(t))$.

The term $(rK - qS(s)) \cdot 1_{S(s) \le S_T^*(s)}\, ds$ (which is always non-negative) is the "carry" associated with the early exercise right of the American option, and represents the cash flow on the time interval $[s, s + ds]$ that the American option holder would require to give up on his early exercise rights. Specifically, we recognize $rK\, ds$ as the interest rate payment on the strike (long), and $-qS(s)\, ds$ as the dividend payment on the stock (short). Integrating the present value of this stream of cash flows yields the American exercise premium. We note that (6) is more general than (4) and continues to hold for more complex Markovian dynamics for S than (1); we shall revisit this in Section 7.

2.3 Integral Equations for the Boundary B

In order to apply (4), a methodology to construct the function $B(\tau)$ is needed. For this purpose, first notice that, for $S > B(\tau)$, $V(\tau, S)$ will satisfy the (time-reversed) Black-Scholes PDE

$$V_\tau - (r - q)SV_S - \frac{1}{2}\sigma^2 S^2 V_{SS} + rV = 0, \quad V(0, S) = (K - S)^+, \tag{7}$$

subject to the *value match condition*

$$V(\tau, B(\tau)) = K - B(\tau) \tag{8}$$

and the *smooth pasting condition*

$$V_S(\tau, B(\tau)) = -1. \tag{9}$$

It is worth noticing that the fundamental conditions (8) and (9) may be combined with each other and with (7) to reveal new relations for the behavior of the American put at the exercise boundary. For instance, differentiating (8) with respect to τ and then using (9) yields

$$\frac{d}{d\tau}V(\tau, B(\tau)) = -\frac{d}{d\tau}B(\tau) = V_S(\tau, B(\tau))\frac{d}{d\tau}B(\tau).$$

On the other hand, by the chain rule,

$$\frac{d}{d\tau}V(\tau, B(\tau)) = V_\tau(\tau, B(\tau)) + V_S(\tau, B(\tau))\frac{d}{d\tau}B(\tau),$$

and it follows that

$$V_\tau(\tau, B(\tau)) = 0, \tag{10}$$

a result we believe was first presented in [22]. Inserting (8), (9), and (10) in (7) then yields, in the limit $S \downarrow B(\tau)$,

$$V_{SS}(\tau, B(\tau)) = \frac{2\,(rK - qB(\tau))}{B(\tau)^2\sigma^2}. \tag{11}$$

Combining any of the four[7] relations (8), (9), (10), (11) with the basic expression (4) will yield a different integral equation for $B(\tau)$. The most common[8] representation in the financial literature uses (8) to write

$$K - B(\tau) = v\left(\tau, B(\tau)\right) + \int_0^\tau rKe^{-r(\tau-u)}\Phi\left(-d_-\left(\tau-u, B(\tau)/B(u)\right)\right)\, du - \int_0^\tau qB(\tau)e^{-q(\tau-u)}\Phi\left(-d_+\left(\tau-u, B(\tau)/B(u)\right)\right)\, du, \tag{12}$$

an integral equation for $B(\tau)$ that resembles[9], but is more complicated than, a non-linear Volterra equation. Using the fact that $\Phi(-x) = 1 - \Phi(x)$ and

$$x\int_0^\tau e^{-x(\tau-u)}\, du = 1 - e^{-x\tau}, \tag{13}$$

we can rewrite (12) in a form that is better suited for our purposes

$$B(\tau)e^{-q\tau}\left\{\Phi\left(d_+(\tau, B(\tau)/K)\right) + q\int_0^\tau e^{qu}\Phi\left(d_+\left(\tau-u, B(\tau)/B(u)\right)\right)\, du\right\} = Ke^{-r\tau}\left\{\Phi\left(d_-(\tau, B(\tau)/K)\right) + r\int_0^\tau e^{ru}\Phi\left(d_-\left(\tau-u, B(\tau)/B(u)\right)\right)\, du\right\}. \tag{14}$$

Differentiating (4) with respect to S and inserting into (9) yields, after multiplication with $-B(\tau)$ and use of (13), the alternative equation

$$B(\tau)e^{-q\tau}\Phi\left(d_+\left(\tau, B(\tau)/K\right)\right) + B(\tau)e^{-q\tau}q\int_0^\tau e^{qu}\left(\Phi\left(d_+\left(\tau-u, B(\tau)/B(u)\right)\right) + \frac{\phi\left(d_+\left(\tau-u, B(\tau)/B(u)\right)\right)}{\sigma\sqrt{\tau-u}}\right) du = rK\int_0^\tau e^{ru}\frac{\phi\left(d_-\left(\tau-u, B(\tau)/B(u)\right)\right)}{\sigma\sqrt{\tau-u}}\, du, \tag{15}$$

where $\phi(x) = (2\pi)^{-1/2}\, e^{-x^2/2}$ is the Gaussian density. Unlike (14) this equation lacks symmetry between integral and non-integral terms, but we may restore this by using

$$\frac{Ke^{-r\tau}}{\sigma\sqrt{\tau}}\phi\left(d_-(\tau, B(\tau)/K)\right) = \frac{B(\tau)e^{-q\tau}}{\sigma\sqrt{\tau}}\phi\left(d_+(\tau, B(\tau)/K)\right),$$

[7] By forming additional derivatives (including cross-derivatives such as $\partial^2 V/\partial S\partial\tau$) many other integral equations are obviously possible. We may also "blend" different integral equations with a set of mixing weights.

[8] Exceptions include [53] and [60] who appear to use (9) and (11), respectively, although the technique used in both papers to develop the integral equation is less straightforward than the one used here. The condition (10) is used by [26] both in technical work and to create an approximate ODE-based iteration for the boundary.

[9] In a regular non-linear Volterra equation, the integration kernel on the right-hand side of the equation does not depend on $B(\tau)$.

which leads to

$$B(\tau)e^{-q\tau}\left\{\Phi\left(d_+(\tau,B(\tau)/K)\right)+\frac{\phi\left(d_+(\tau,B(\tau)/K)\right)}{\sigma\sqrt{\tau}}\right.$$
$$\left.+\,q\int_0^\tau e^{qu}\left(\Phi\left(d_+\left(\tau-u,B(\tau)/B(u)\right)\right)+\frac{\phi\left(d_+\left(\tau-u,B(\tau)/B(u)\right)\right)}{\sigma\sqrt{\tau-u}}\right)du\right\}$$
$$=Ke^{-r\tau}\left\{\frac{\phi\left(d_-(\tau,B(\tau)/K)\right)}{\sigma\sqrt{\tau}}+r\int_0^\tau e^{ru}\frac{\phi\left(d_-\left(\tau-u,B(\tau)/B(u)\right)\right)}{\sigma\sqrt{\tau-u}}\,du\right\}. \tag{16}$$

3 Numerical Schemes for the Exercise Boundary

3.1 First Approach to Numerical Solution

The various integral equations in Section 2.3 above have no simple analytical solution, and generally need to be solved by numerical methods. The most straightforward technique (see [3], [51], [52], [54], [68], to name a few) attacks (14) with a slight adaptation of *direct quadrature methods* for Volterra equations (as presented, for instance, in [66], Chapter 18.2). This approach uses fixed numerical quadrature weights (e.g., trapezoid) on a discrete grid $\{\tau_i\}_{i=0}^n$, and constructs $B(\tau_i)$ sequentially starting from $B(\tau_0) = K\min(1, r/q)$ at $\tau_0 = 0+$. Given $B(\tau_0), B(\tau_1), \ldots, B(\tau_{i-1})$, satisfying (14) at $\tau = \tau_i$ will involve a non-linear root search for $B(\tau_i)$, easily handled by the one-dimensional Newton's method, say. Once the exercise boundary function has been constructed on the grid $\{\tau_i\}_{i=0}^n$, the entire exercise boundary function might be approximated through interpolation (e.g., a spline, as in [33]).

Due to the the presence of τ and $B(\tau)$ in the integration kernels, when computing the integrals needed to establish $B(\tau_i)$ at some point τ_i, the numerical quadrature scheme must be applied to the entire interval $[0, \tau_i]$, rather than just $[\tau_{i-1}, \tau_i]$. As a consequence, the effort of direct quadrature is typically $O(mn^2)$, where n is the number of discretization points and m is the average number of root-search iterations required to establish $B(\tau_i)$. The convergence of such methods is algebraic[10] and rather modest compared to modern methods for integral equations.

3.2 Second Approach to Numerical Solution

Equations of the type (14)-(16) may all be rearranged to the form

$$B(\tau) = Ke^{-(r-q)\tau}\frac{N(\tau,B)}{D(\tau,B)} \tag{17}$$

[10] A numerical method has *algebraic convergence* if its error decreases as $1/n^p$ with n being the number of discretization points and p some constant (say, 2 for a second-order method). A method with *spectral convergence*, on the other hand, has an error that decreases as $1/n^n$, i.e., exponentially.

where N, D are functionals depending on $B(u)$, $u \leq \tau$. For instance, for the case (16) we can introduce operators[11]

$$\mathcal{K}_1(\tau) = \int_0^\tau e^{qu}\Phi\left(d_+\left(\tau - u, B(\tau)/B(u)\right)\right)\, du, \tag{18}$$

$$\mathcal{K}_2(\tau) = \int_0^\tau \frac{e^{qu}}{\sigma\sqrt{\tau - u}}\phi\left(d_+\left(\tau - u, B(\tau)/B(u)\right)\right)\, du, \tag{19}$$

$$\mathcal{K}_3(\tau) = \int_0^\tau \frac{e^{ru}}{\sigma\sqrt{\tau - u}}\phi\left(d_-\left(\tau - u, B(\tau)/B(u)\right)\right)\, du, \tag{20}$$

such that (say)

$$N(\tau, B) = \frac{\phi\left(d_-(\tau, B(\tau)/K)\right)}{\sigma\sqrt{\tau}} + r\mathcal{K}_3(\tau), \tag{21}$$

$$D(\tau, B) = \frac{\phi\left(d_+(\tau, B(\tau)/K)\right)}{\sigma\sqrt{\tau}} + \Phi\left(d_+(\tau, B(\tau)/K)\right) + q\left(\mathcal{K}_1(\tau) + \mathcal{K}_2(\tau)\right). \tag{22}$$

For ease of reference, we denote (17) with N, D set as in (21)-(22) as *fixed point system A*, abbreviated as FP-A.

On the other hand, for (14), say, we have

$$N(\tau, B) = \Phi\left(d_-(\tau, B(\tau)/K)\right) + r\int_0^\tau e^{ru}\Phi\left(d_-\left(\tau - u, B(\tau)/B(u)\right)\right)\, du, \tag{23}$$

$$D(\tau, B) = \Phi\left(d_+(\tau, B(\tau)/K)\right) + q\int_0^\tau e^{qu}\Phi\left(d_+\left(\tau - u, B(\tau)/B(u)\right)\right)\, du. \tag{24}$$

Equation (17) with N, D set as in (23)-(24) is denoted *fixed point system B*, or FP-B for short.

Fixed point system A was used in the recent paper [53] for the case[12] $q = 0$ to devise a fixed point iteration on an equidistant grid $\{\tau_i\}_{i=1}^n$ where, in the jth iteration,

$$B^{(j)}(\tau_i) = Ke^{-(r-q)\tau_i}\frac{N\left(\tau_i, B^{(j-1)}\right)}{D\left(\tau_i, B^{(j-1)}\right)}, \quad i = 1, \ldots, n. \tag{25}$$

[53] initialize (25) at a flat initial guess of $B^{(0)}(\tau) = K\min(1, r/q)$ for all τ, with about $6 - 10$ fixed point iterations needed for convergence. Using polynomial interpolation and adaptive Gauss-Kronrod quadrature to evaluate the integrals $\mathcal{K}_1, \mathcal{K}_1, \mathcal{K}_3$ in N and D, they record reasonable numerical efficiency, roughly 50-100 and 5-10 times faster than the binomial tree and finite difference methods, respectively. If l is the (average)

[11] Operators $\mathcal{K}_1$ and $\mathcal{K}_2$ might be combined. Splitting them serves to highlight the fact that $\mathcal{K}_2$ has a singular kernel, due to the factor $(\tau - u)^{-1/2}$.

[12] [53] list, but never test, expressions for the case $q > 0$. These expressions contain a number of typos which we have corrected here.

number of Gauss-Kronrod quadrature points used, the computational complexity of the algorithm is $O(lmn)$, excluding the polynomial interpolation.

We notice that the n equations in (25) are independent of each other, which allows for straightforward parallelization of the algorithm[13] across multiple processing units. This observation is used in [33] where a variant of the method in [53] has been implemented, with predictably good performance, on a multi-core computer. Besides emphasizing the parallelization aspect of fixed point iterations, [33] modify the algorithm in [53] in a number of ways, ultimately recommending that (25) be initialized at an approximation in [12] and, rather surprisingly, that all integrals be evaluated by trapezoid integration on an equidistant maturity grid. Perhaps most significantly, [33] conclude that the integrals in (21)-(22) are less well-behaved, and result in slower performance[14], than those in (23)-(24).

3.3 Analysis of Fixed Point Iteration Schemes

Iteration schemes based on (17) are obviously not unique: not only do we have multiple equivalent expressions for the boundary, we also have multiple ways to arrange each of these expressions into the form (17). For instance, if we let h denote some function or functional, it is clear that (17) remains valid (subject to some regularity on h) if we add $e^{(r-q)\tau}B(\tau)/K \cdot h$ to N and h to D, i.e. we write

$$B(\tau) = Ke^{-(r-q)\tau}\frac{N(\tau,B) + e^{(r-q)\tau}B(\tau)/K \cdot h(\tau,B)}{D(\tau,B) + h(\tau,B)}. \tag{26}$$

Alternatively, we may use a more traditional relaxation-type formulation, where we write

$$B(\tau) = Ke^{-(r-q)\tau}\frac{N(\tau,B)}{D(\tau,B)}(1 - h(\tau,B)) + h(\tau,B)B(\tau) \tag{27}$$

for some function(al) h. Despite their formal equivalence, it should be obvious that not all expressions for $B(\tau)$ are equally suited for embedding in a fixed point iteration; in fact, some seemingly reasonable formulations might fail to define a proper contraction mapping and will not converge.

Establishing whether a particular equation for the boundary is a candidate for fixed point iteration is essentially a question of how sensitive the right-hand side of (17) is to perturbations in the exercise boundary: the lower, the better. With this in mind, cursory examination of the various expressions in Section 2.3 shows that the two most promising

[13] Low-level parallelization of option pricing algorithms – especially relatively fast ones – is often of limited practical value, inasmuch as most banks do not price a single option at a time. Rather, they price entire trading books, often involving 1000s of options, at many input parameter settings. Should one have multiple cores available for computation, a more straightforward parallelization strategy would be to have each core assigned to a different option and/or a different input configuration, rather than having multiple cores dealing with a single option.

[14] As demonstrated in Section 6, we have not been able to detect this. We speculate that [33] may have run into difficulties with the $(\tau - u)^{-1/2}$ singularities in (21)-(22). See Section 5 for how to handle this.

candidates[15] for (17) are the two (fixed point systems A and B) already considered in Section 3.2.

Characterizing perturbation sensitivity is most easily done through the Gateaux derivative formalism, listed below for our two candidate equations. The proof relies only on simple algebraic manipulations and is omitted.

Lemma 1 *Set* $f \triangleq Ke^{-(r-q)\tau}N/D$ *in (17), and consider a perturbation of the exercise boundary around its optimal location, of the proportional form*

$$\ln B(\tau) \longrightarrow \ln B(\tau) + \omega g(\tau), \quad \omega \in \mathbb{R},$$

where $g : \mathbb{R} \longrightarrow \mathbb{R}$ *is a given scalar function. Then, for both fixed point system A (in equations (21)-(22)) and fixed point system B (in equations (23)-(24)),*

$$\frac{\partial f}{\partial \omega}|_{\omega=0} = \frac{Ke^{-(r-q)\tau}}{D(\tau, B)}\epsilon, \tag{28}$$

where

$$\epsilon = \int_0^\tau e^{ru}\left(r - q\frac{B(u)}{K}\right)\psi\left(\tau - u, B(\tau)/B(u)\right)\frac{\phi\left(d_-\left(\tau - u, B(\tau)/B(u)\right)\right)}{\sigma\sqrt{\tau - u}}\left(g(\tau) - g(u)\right)du.$$

For system B, we have $\psi = -1$*, and for system A*

$$\psi\left(\tau - u, B(\tau)/B(u)\right) = \frac{d_-\left(\tau - u, B(\tau)/B(u)\right)}{\sigma\sqrt{\tau - u}}. \tag{29}$$

Corollary 1 *For flat proportional shifts of the exercise boundary away from its optimal location, i.e., when* $g(\tau)$ *is a constant, then, for both fixed point systems A and B,*

$$\frac{\partial f}{\partial \omega}|_{\omega=0} = 0.$$

While Corollary 1 is limited to flat proportional shifts, the insensitivity to such shifts nevertheless suggests that a fixed point iteration on both fixed point systems A and B should generally be effective. The computational efforts involved in evaluating N/D are more or less identical for the two systems, and Corollary 1 does not settle which of the two representations is more favorable in numerical work. If we focus on robustness towards non-parallel proportional shifts, (28) suggests that a relevant metric is

$$\max_{u\le\tau}\left\|\frac{\psi\left(\tau - u, B(\tau)/B(u)\right)}{D(\tau, B)}\right\|,$$

in the sense that the method with the smaller value might be less affected by non-parallel proportional perturbations.

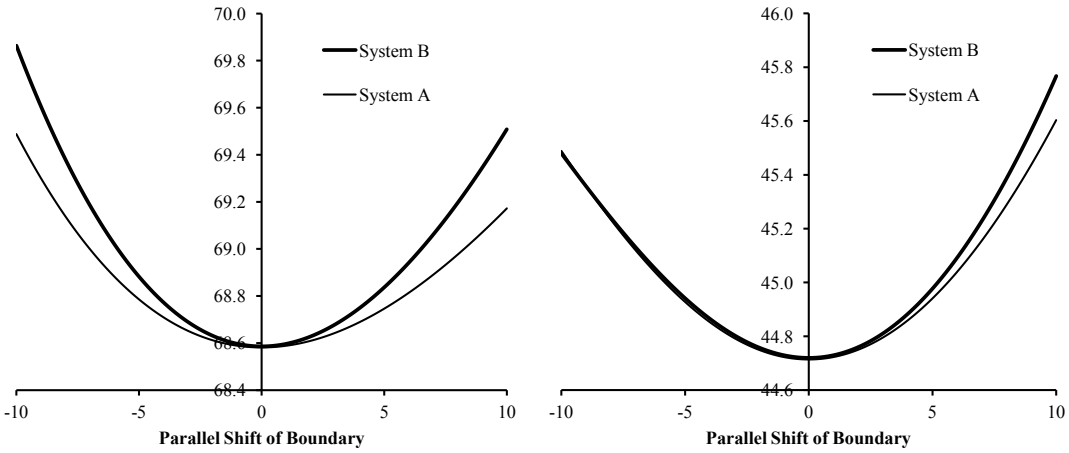
[15] As pointed out in Section 2.3, there are a large number of candidate equations available to us, and we do not claim to have performed an exhaustive search.

Another metric of interest is the convexity of the function f, e.g., as measured by $\partial^2 f/\partial\omega^2$; for fast convergence, we want the absolute value of this derivative to be as small as possible[16]. A few calculations show that, for parallel shifts in log-space (i.e., $g(x)$ a constant),

$$\frac{\partial^2 f}{\partial \omega^2}|_{\omega=0} = -\frac{Ke^{-(r-q)\tau}}{B(\tau)^2}\frac{\phi\left(d_-(\tau, B(\tau)/K)\right)}{\sigma\sqrt{\tau}}\frac{\psi\left(\tau, B(\tau)/K\right)}{D(\tau, B)}$$

where ψ is given in Lemma 1. Comparing with (29), it follows that the quantity ψ/D determines both the robustness and convergence speed of the fixed point system. Empirically, one finds that fixed point system A normally produces smaller values of $|\psi/D|$ than system B, except for the case where $r \gg q$ and σ is small. Figures 1 and 2 show a few illustrative graphs; notice that the convexity of the graphs are generally lower for system A, except for the left panel in Figure 2 where r is large relative to q.

Figure 1: RHS of Fixed Point System vs Perturbations of Boundary. Case $r \leq q$.



Notes: The functional $f = Ke^{-(r-q)\tau}N/D$ in (17), graphed against parallel proportional perturbations to the exercise boundary from its optimal level. Left graph: $r = q = 5\%$, $\sigma = 25\%$, $K = 130$, $\tau = 5$ years. Right graph: $r = 2.5\%$, $q = 5\%$, $\sigma = 25\%$, $K = 130$, $\tau = 5$ years.

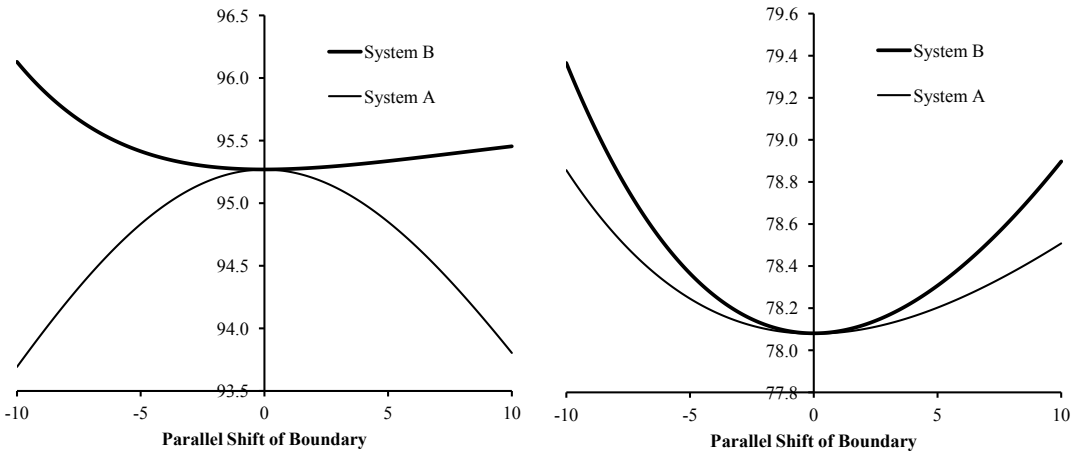
In summary, our investigations so far suggest that system A, despite the conclusions of [33], overall seems more promising than system B, except possibly for the case $r \gg q$ where system A may, if iterated on directly, be less stable than system B. We examine these predictions empirically later, in Section 6.

3.4 Roadmap for the Design of an Efficient Numerical Scheme

Excluding interpolation cost, the computational effort of existing methods designed to attack (17) is, as discussed, $O(lmn)$, where l is the number of nodes used in the evaluation

[16]Recall that a (scalar) fixed point iteration $x = f(x)$ has convergence order q if $f'(x_p) = f''(x_p) = \ldots = f^{(q-1)}(x_p) = 0$, where x_p is the fixed point.

Figure 2: RHS of Fixed Point System vs Perturbations of Boundary. Case $r > q$.



Notes: The functional $f = Ke^{-(r-q)\tau}N/D$ in (17), graphed against parallel proportional perturbations to the exercise boundary from its optimal level. Left graph: $r = 10\%$, $q = 0\%$, $\sigma = 10\%$, $K = 100$, $\tau = 3$ years. Right graph: $r = 5\%$, $q = 2.5\%$, $\sigma = 25\%$, $K = 130$, $\tau = 5$ years.

of integrals, m is the number of iterations used in the fixed point iteration, and n is the number of points on the τ-grid at which the fixed point iteration is solved. Our goal in this paper is to design an efficient spectral collocation method for the numerical solution of (17), where we aim to keep each of the constants l, m, n as small as possible. The strategy for this involves multiple steps, but roughly speaking our roadmap is as follows:

1. Formulate the fixed point iteration system (17) to be rapidly converging and stable, through relaxation (e.g., (27)) and a Jacobi-Newton iteration strategy.

2. Start the fixed point iteration at a tight, and computationally efficient, first guess for B.

3. Establish a strategy to set up a sparse fixed point grid in τ-space.

4. Choose a smooth function space and a suitable variable transformation to interpolate B on the the τ-grid.

5. Choose an efficient variable transformation and quadrature scheme to numerically compute the integrals in (17).

Our proposed scheme is discussed in detail in Section 5, but hinges on additional results for the asymptotics of the exercise boundary for small τ. We cover the required results in Section 4 below.

4 Boundary Properties and Approximations

4.1 Short-Expiration Asymptotic Behavior of B

To better understand the shape of the the function $B(\tau)$, it is useful to first consider known asymptotic results for $B(\tau)$ for large and small values of τ. Starting with the small-τ limit, relevant results have been developed by numerous authors, including [10], [27], [37], and, most recently, [77]. First, we notice that while always $B(0) = K$, a small carry argument (similar to the one that lead to (6)) demonstrates that $\lim_{\tau \downarrow 0} B(\tau) = X$, where

$$X = \begin{cases} K, & r \geq q, \\ K\left(\frac{r}{q}\right), & r < q. \end{cases} \tag{30}$$

When $r < q$, the boundary is therefore discontinuous at $\tau = 0$. The limit-behavior of $B(\tau)$ as τ approaches zero can be further characterized through asymptotic expansions, see e.g. [77]. For our purposes, leading-order terms suffice:

$$\frac{B(\tau)}{X} \sim \frac{B_s(\tau)}{X} = \begin{cases} \exp\left(-\sqrt{-k_1 \tau \ln(k_2 \tau)}\right), & r = q, \\ \exp\left(-\sqrt{-\frac{1}{2} k_1 \tau \ln(k_3 \tau)}\right), & r > q, \\ \exp\left(-k_4 \sqrt{\tau}\right), & r < q, \end{cases} \tag{31}$$

where

$$k_1 = 2\sigma^2, \quad k_2 = 4\sqrt{\pi} r, \quad k_3 = 8\pi \left(\frac{r-q}{\sigma}\right)^2, \quad k_4 \approx \sigma\sqrt{2} \cdot 0.451723.$$

We notice that the square root short-expiration boundary $B_s(\tau)$ for $r < q$ is quite different from those of $r \geq q$ which involve logarithms. We examine this in more detail later.

4.2 Long-Expiration Asymptotic Behavior of B

For $\tau \to \infty$, the exercise boundary straightens out (from above) to a flat strike-dependent level B_{inf}, which may be solved for easily by standard barrier pricing methods, as in [64]. The result is:

$$B_{\text{inf}} = K \frac{\theta_-}{\theta_- - 1}, \tag{32}$$

where

$$\theta_\pm = \alpha \pm \sqrt{\beta}, \quad \alpha = \frac{1}{2} - \frac{r-q}{\sigma^2}, \quad \beta = \alpha^2 + 2\sigma^{-2} r.$$

The decay towards B_{inf} for large τ is not trivial, and only fairly recently has there been progress in characterizing the long-term asymptotics. [32] and [1] propose similar asymptotic results, which recently were extended and sharpened by [29]. The [29] result is

$$\ln B(\tau) \sim \ln B_{\text{inf}} + \gamma \left(\sigma^2 \tau / 2\right)^{-3/2} e^{-\beta \tau}, \quad \tau \to \infty, \tag{33}$$

where γ is an unknown constant. The presence of an unknown constant γ is shared by the expressions in [32] and [1], and limits the practical usefulness of the results in many respects.

4.3 Boundary Shape

The following result was shown in [28] (see also [16]).

Theorem 1 *The American put exercise boundary $B(\tau)$ is infinitely differentiable (C^∞) on $\tau \in (0, \infty)$. When $r \geq q$ or $q \gg r$, the boundary $B(\tau)$ is convex for all $\tau > 0$. However, when $q > r$, and $q - r \ll 1$ then $B(\tau)$ is* not *uniformly convex in τ. In particular, if $\varepsilon = \ln(q/r)$ is positive and sufficiently small, then there exists a $\hat{\tau}$ for which $d^2B(\hat{\tau})/d\tau^2 < 0$, where*

$$0 < \hat{\tau} \leq \frac{\varepsilon}{3\sigma^2 \left|\ln(\varepsilon)\right|}.$$

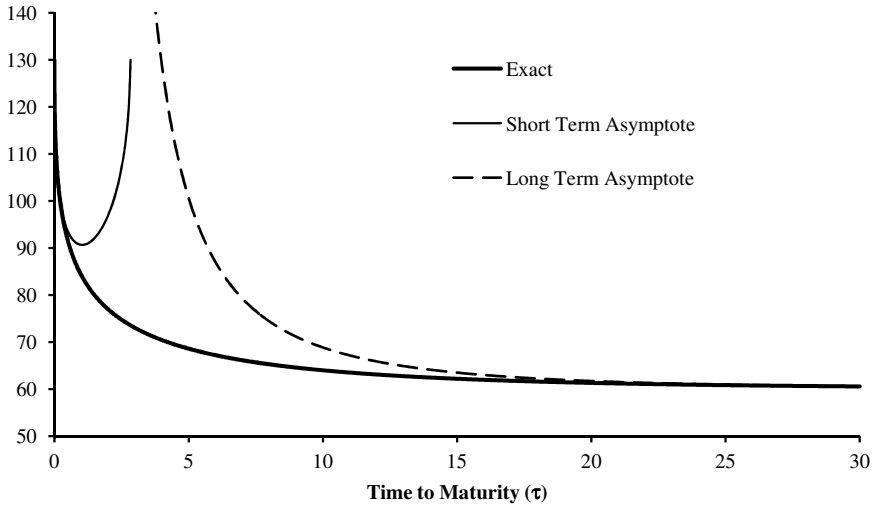
We may take two things away from this theorem. First, the exercise boundary is smooth, except at the origin $\tau = 0$. Second, the boundary is generally convex except for the case where q is slightly smaller than r. For this case, there will be a small region of non-convexity close to $\tau = 0$. As we show in a later example (see Figure 4), the violation of convexity can negatively impact the precision of short-time expansions.

4.4 Numerical Examples

To get a feel for the various asymptotic expansions above, consider first the case where $r = q = 5\%$, $K = 130$, and $\sigma = 25\%$. Figure 3 below shows the behavior of the short- and long-term approximation for B in (31) and (33), respectively, along with a high-accuracy estimate of the true exercise boundary. In estimating the constant γ in (33), we followed [32]) and match (33) to the true asymptote at $\sigma^2\tau/2 = 1$. It is quite obvious that both asymptotic expressions have limited ranges of applicability: the long-term expansion has very poor precision for maturities less than several decades, and the short-term expansion starts to show marked inaccuracies after less than a year (and eventually outright ceases to exist after $\tau = 2.82$ years). Of particular notice in Figure 3 is the behavior around the origin, where the exercise boundary undergoes rapid change and, in the limit of $\tau \downarrow 0$, exhibits unbounded derivatives of all orders.

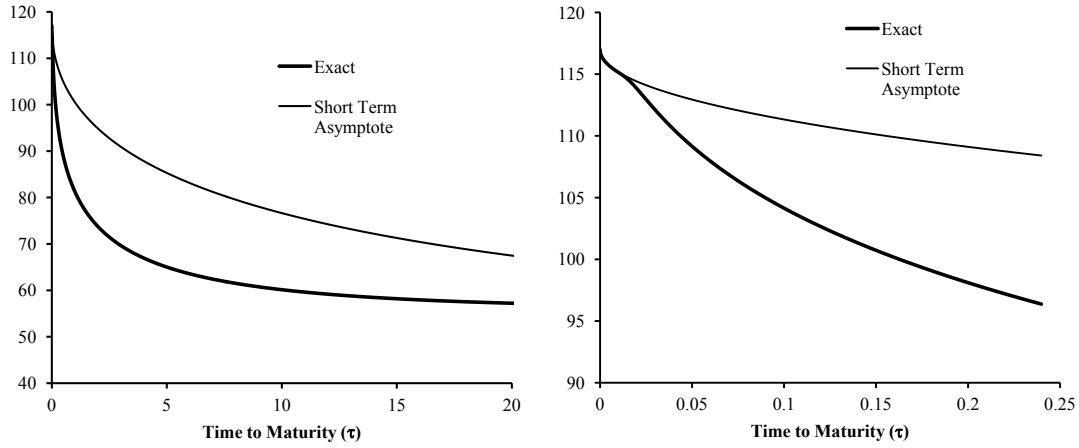
The case $r > q$ generally behaves similar to the case $r = q$, with the limit $r \downarrow q$ moving smoothly to the case $r = q$. For the case $r < q$, however, the short-term asymptotic behavior changes abruptly as r goes below q, with the $\sqrt{\tau \ln \tau}$ behavior being replaced by a simpler $\sqrt{\tau}$ asymptote. While the $\sqrt{\tau}$ asymptote is "classical" (see [75]) and often useful for $r \ll q$, it is not robust for r close to q, due to the break-down of convexity in a region close to the origin; see Theorem 1. Figure 4 illustrates this phenomenon, especially the right panel where the behavior around the origin has been emphasized. It is notable that virtually all papers on short-term expansions do not appear to recognize the deficiency apparent in the figure.

Figure 3: Exercise Boundary Asymptotes



Notes: Various asymptotes of $B(\tau)$ for the case $r = q = 5\%$, $\sigma = 25\%$, $K = 130$. The "Short Term Asymptote" and "Long Term Asymptote" graphs are computed from from (31) and (33), respectively. The "Exact" boundary is computed numerically by the method outlined in Section 5.

Figure 4: Exercise Boundary Asymptote



Notes: Short term asymptote of $B(\tau)$ for the case $r = 4.5\%$, $q = 5\%$, $\sigma = 25\%$, $K = 130$. The two panels above are identical, except for the range on the x-axis. The "Short Term Asymptote" graph is computed from from (31) and the "Exact" boundary is computed numerically by the method outlined in Section 5.

4.5 Approximations to the Exercise Boundary

Dating back to the 1980s, there is a remarkable volume of literature on methods to approximate the early exercise boundary. For a sample of results, see, among others, [11], [12], [14], [18], [22], [23], [26], [32], [39], [43], [46], [49], [50], [56], [58], [59], [63], [78], and [79]. We should note that not all of these support general values for μ in (1), as many older methods rely on the (classical) zero-dividend assumption, $\mu = r$.

Based on the recent comparisons in [59] and [39], the most precise of the currently published methods appears to be the QD^+ algorithm of [59], a method built on ideas in [50]. Like the majority of published methods, QD^+ involves implicit definitions of the boundary that must, at each point τ, be resolved by iterations and a root-search algorithm. The QD^+ algorithm will be used in many of our numerical tests as a first guess for the boundary, so for ease of reference we outline the algorithm in Appendix A.

4.6 Approximation based on Asymptotics

For later purposes, we briefly want to develop an approximation where the boundary is entirely analytical, i.e. does not involve a root search for each τ at which the boundary is needed. We do this by specifying the boundary as an analytical function, meant to reasonably interpolate the long- and short-time asymptotic behavior listed in Section 2 above. For concreteness, we first focus on the case $r \geq q$.

Judging from Figure 3, it appears reasonable to first focus on the challenging small-τ regime. To get around the fact that (31) ceases to existing for large τ, let us consider replacing the logarithmic term with a different function, e.g.,

$$x^*(\tau) \sim -\sqrt{\tau\alpha(\tau)}$$

for some function $\alpha : \mathbb{R}^+ \to \mathbb{R}^+$. To mimic the behavior of $x^*(\tau)$ around the origin, we require that $\alpha(\tau)$ explode in the limit $\tau \downarrow 0$, at a rate strictly slower than τ^{-1}. Additionally, we wish for the derivative of $\tau\alpha(\tau)$ to become unbounded for $\tau \downarrow 0$.

Combining the requirements above with a nod towards simplicity and ease of computing, one possible *ansatz* is to set

$$\alpha(\tau) = a\tau^b, \quad a > 0, b \in (-1, 0), \tag{34}$$

for constants a, b to be determined. To also ensure reasonable long-term asymptotic behavior, more precisely we here assume that

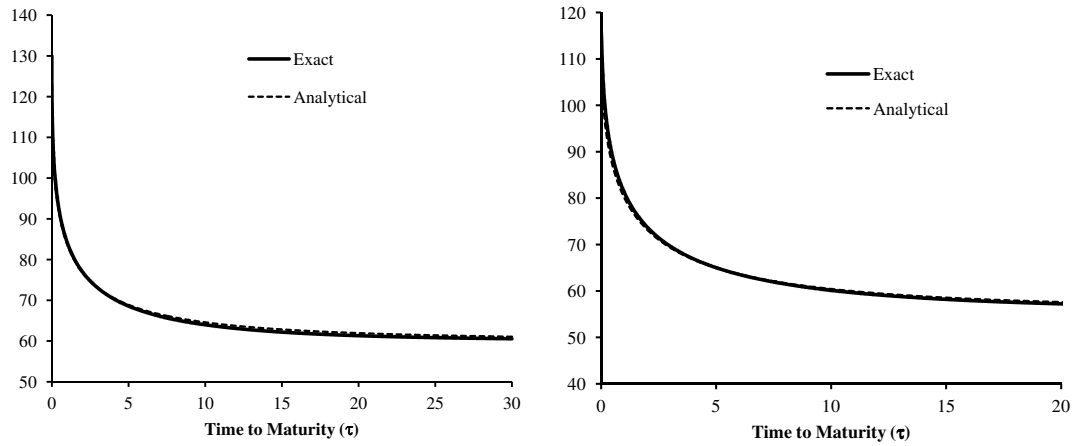
$$B(\tau) \approx \tilde{B}(\tau) \triangleq B_{\text{inf}} + (K - B_{\text{inf}})e^{-\sqrt{a\tau^{b+1}}}, \tag{35}$$

where B_{inf} is given in (32).

There are numerous ways of determining the "best" values of the constants a and b, but what matters here is that a and b in practice can always be set such that (35) is remarkably close to the true exercise boundary for all values of τ. In fact, this is true

for all configurations of r and q, not just $r \geq q$; see Figure 5 below for some examples. In virtually all cases, the best-fitting value of b is slightly negative, even[17] for the case $r < q$.

Figure 5: Exercise Boundary Approximation



Notes: Approximation of $B(\tau)$ using (35). Left panel: $r = q = 5\%$, $\sigma = 25\%$, $K = 130$. Right panel: $r = 4.5\%$, $q = 5\%$, $\sigma = 25\%$, $K = 130$. The "Analytical" graphs use (35) with $a = 1.3$ and $b = -0.15$ (left panel), and $a = 0.85$ and $b = -0.1$ (right panel). The "Exact" boundary is computed numerically by the method outlined in Section 5.

5 A New Numerical Scheme

Having now laid sufficient foundation, we return to the return to the roadmap in Section 3.4 and begin the concrete development of an efficient numerical scheme for the boundary equation (17). For reasons already discussed, we focus on fixed point system A (equations (21)-(22)), but our algorithm is easily extended to fixed point system B (equations (23)-(24)).

Fundamentally, we wish to use a *collocation scheme*, i.e., we solve (17) on a discrete set of collocation nodes $\{\tau_i\}_{i=1}^n$ only, and construct the full function $B(\tau)$ by interpolation. While there are many possibilities for this interpolation (e.g., splines), we here rely on polynomial approximation[18]; the resulting scheme can therefore also be cast as a projection method on the space of n-dimensional polynomials in τ. Background material

[17] While (31) suggest that $b = 0$ for the case $r < q$, we have already seen in Figure 4 that this can be misleading.

[18] There are numerous "myths" about the supposed instability of polynomial approximation, especially when the polynomial is of high order. As discussed in [72], such instabilities are associated with poor choices of interpolation grids (e.g., equidistant grids) and computational methods, rather with the fundamental precision of polynomial approximation.

on collocation-type projection methods for integral equations – a very active research area in numerical mathematics – can be found in [8], [19], [36], and [69].

In the practical computation of the boundary, specific algorithms are required for the computation of integral operators and for the numerical solution of (17). In addition, we need to make efficient choices for the parameter space in which we operate, especially in regards to integration variable transformations and for boundary interpolations. We outline, and justify, our choices in the following sections.

5.1 Fixed Point Iteration Scheme

Let us first investigate how we can use the relaxation formulation (27) to construct a (dampened) Jacobi-Newton iteration for the fixed point system (17). Dropping some arguments, we write

$$B(\tau) = (1 - h(\tau))f(\tau, B) + h(\tau)B(\tau), \tag{36}$$

where

$$f(\tau, B) = K^*(\tau)\frac{N(\tau, B)}{D(\tau, B)} \tag{37}$$

with $K^*(\tau) \triangleq Ke^{-(r-q)\tau}$. For a diagonal Jacobi iteration scheme, we are interested in measuring the sensitivity of the right-hand side of (27) with respect to moves in[19] $B(\tau)$. Treating f as a functional on an arbitrary function Q, we get

$$f'(\tau, Q) \triangleq \frac{\partial f}{\partial Q(\tau)} = K^*(\tau)\left(\frac{N'(\tau, Q)}{D(\tau, Q)} - \frac{D'(\tau, Q)N(\tau, Q)}{D(\tau, Q)^2}\right). \tag{38}$$

where N' and D' indicate partial derivatives with respect to $B(\tau)$. For instance, for fixed point system A (with N and D given in (21)-(22)), we have

$$\begin{aligned} N'(\tau, Q) = &-d_-(\tau, Q(\tau)/K)\frac{\phi\left(d_-(\tau, Q(\tau)/K)\right)}{Q(\tau)\sigma^2\tau} \\ &- r\int_0^\tau \frac{e^{ru}d_-(\tau, Q(\tau)/Q(u))}{Q(\tau)\sigma^2(\tau - u)}\phi\left(d_-\left(\tau - u, Q(\tau)/Q(u)\right)\right)du, \end{aligned} \tag{39}$$

and

$$\begin{aligned} D'(\tau, Q) = &-\frac{K^*(\tau)}{Q(\tau)}d_-(\tau, Q(\tau)/K)\frac{\phi\left(d_-(\tau, Q(\tau)/K)\right)}{Q(\tau)\sigma^2\tau} \\ &- q\frac{K^*(\tau)}{Q(\tau)}\int_0^\tau \frac{Q(u)}{K}\frac{e^{ru}d_-\left(\tau - u, Q(\tau)/Q(u)\right)}{\sigma^2(\tau - u)}\phi\left(d_-\left(\tau - u, Q(\tau)/Q(u)\right)\right)du. \end{aligned} \tag{40}$$

In an iterative scheme for B, at iteration j we now wish to locally cancel out first-order sensitivity by setting h such that the derivative $(1 - h)f'$ equals $-h$. This suggests an iteration where

$$B^{(j)}(\tau) = \left(1 - h^{(j-1)}(\tau)\right)f\left(\tau, B^{(j-1)}\right) + h^{(j-1)}(\tau)B^{(j-1)}(\tau)$$

[19] Notice that we do not consider the sensitivity to $B(u)$, $u \neq \tau$, which defines our scheme to be of the Jacobi type.

with

$$h^{(j-1)}(\tau) = \frac{f'\left(\tau, B^{(j-1)}\right)}{f'\left(\tau, B^{(j-1)}\right) - 1}.$$

Rearranging, we arrive at the Jacobi-Newton scheme we will rely on:

$$B^{(j)}(\tau) = B^{(j-1)}(\tau) + \eta \frac{B^{(j-1)}(\tau) - f\left(\tau, B^{(j-1)}\right)}{f'\left(\tau, B^{(j-1)}\right) - 1}. \tag{41}$$

We notice that a naive (Richardson) fixed point scheme is recovered if we set $\eta = 1$ and $f'(\tau, B^{(j-1)}) = 0$.

(41) relies on the computation of two new integrals, in order to evaluate $f'(\tau, B^{(j-1)})$. While this can often be done rapidly (several terms in the integrands are shared with other integrals), to the extent that we are mainly concerned with near-proportional shifts (see Lemma 1), it may be sufficient to ignore the integrals terms in (39) and (40). Doing so generally results in a performant scheme, and is the one that we shall use in most of our numerical tests.

5.2 Time Variable Transformation

Our fundamental iteration algorithm (41) will be applied on a discrete set of n collocation nodes $\{\tau_i\}_{i=1}^n$, to be established later. In the evaluation of the necessary integrals, the presence of $(\tau - u)^{-1/2}$ in the integrals $\mathcal{K}_2(\tau)$ and $\mathcal{K}_3(\tau)$ (see (19)-(20)) of fixed point system A first needs to be considered. While there are numerous methods in the literature for handling weakly singular kernels (e.g., [20] and [21]), we here deal with the singularity analytically through the variable transformation

$$z = \sqrt{\tau - u} \tag{42}$$

which, as $dz = -\frac{1}{2}(\tau - u)^{-1/2}\, du$, removes the kernel singularity of $\mathcal{K}_2(\tau)$ and $\mathcal{K}_3(\tau)$. Using the z variable, the integration region is $[0, \sqrt{\tau}]$ (rather than $[0, \tau]$). As we later want to apply high-precision quadrature to approximate the integrals, it is convenient to additionally introduce a transformation to normalize the integrals to the standard quadrature interval $[-1, 1]$, by writing

$$y = -1 + 2\frac{z}{\sqrt{\tau}} = -1 + 2\frac{\sqrt{\tau - u}}{\sqrt{\tau}}. \tag{43}$$

Applying this transformation to all three integrals $\mathcal{K}_1(\tau), \ldots, \mathcal{K}_3(\tau)$ we get

$$\mathcal{K}_1(\tau) = \frac{e^{q\tau}}{2}\tau \int_{-1}^{1} e^{-\frac{q}{4}\tau(1+y)^2}(y+1)\Phi\left(d_+\left(\tau\left(1+y\right)^2/4, \frac{B(\tau)}{B(\tau - \tau\left(1+y\right)^2/4)}\right)\right) dy, \tag{44}$$

$$\mathcal{K}_2(\tau) = e^{q\tau}\sqrt{\tau} \int_{-1}^{1} \frac{e^{-\frac{q}{4}\tau(1+y)^2}}{\sigma}\phi\left(d_+\left(\tau\left(1+y\right)^2/4, \frac{B(\tau)}{B(\tau - \tau\left(1+y\right)^2/4)}\right)\right) dy, \tag{45}$$

$$\mathcal{K}_3(\tau) = e^{r\tau}\sqrt{\tau} \int_{-1}^{1} \frac{e^{-\frac{r}{4}\tau(1+y)^2}}{\sigma}\phi\left(d_-\left(\tau\left(1+y\right)^2/4, \frac{B(\tau)}{B(\tau - \tau\left(1+y\right)^2/4)}\right)\right) dy. \tag{46}$$

We notice that the transformation (42) is not strictly necessary for $\mathcal{K}_1(\tau)$, but it is generally easier to apply it to all integrals. For the integrals in fixed point system B, the transformation is not needed [20], but does little harm if applied.

5.3 Numerical Integration Scheme

Assume for a moment that we can evaluate the function $B(\tau - \tau\,(1+y)^2\,/4)$ for all y on $(-1, 1)$. To compute (44)-(46), a numerical integration scheme is required to approximate the integrals. As our integrand is smooth (see Theorem 1), a number of quadrature rules are possible here, such as Gaussian and tanh-sinh quadrature rules. Common for such methods is that one may write, for each τ-indexed integrand $c(y;\tau)$ in (44)-(46), an l-point scheme

$$\int_{-1}^{1} c(y;\tau)\,dy \approx \sum_{k=1}^{l} w_k c(y_k;\tau), \tag{47}$$

where the weights w_k and the nodes y_k are specific to the chosen quadrature rule.

5.4 Interpolation and Collocation Time Grid

As discussed earlier, we intend to apply nth order polynomial interpolation to uncover B between collocation points. While we would like to keep the value of n as low as possible, it is clear from numerical results in Section 4.5 that the function $B(\tau)$ is not necessarily well-characterized by a low-dimensional polynomial in τ, especially close to the origin where derivatives diverge. As mentioned earlier, empirical examination of the power b in (34) shows that b is virtually always small (and non-positive), so one would expect the function $\ln(B(\tau))^2$ to be quite close[21] to a straight line for small τ. Writing our interpolation scheme on $\ln(B(\tau))^2$ in τ-space works well, as does using the transformation

$$G(\sqrt{\tau}) = \ln\left(B(\tau)/X\right), \quad X = K\min(1, r/q). \tag{48}$$

As confirmed in Figure 6, the function G is generally much better behaved than B itself, and a good candidate for polynomial interpolation. As it turns out, the combination of the two transformations, i.e.,

$$H(\sqrt{\tau}) = G(\sqrt{\tau})^2 = \ln\left(B(\tau)/X\right)^2, \tag{49}$$

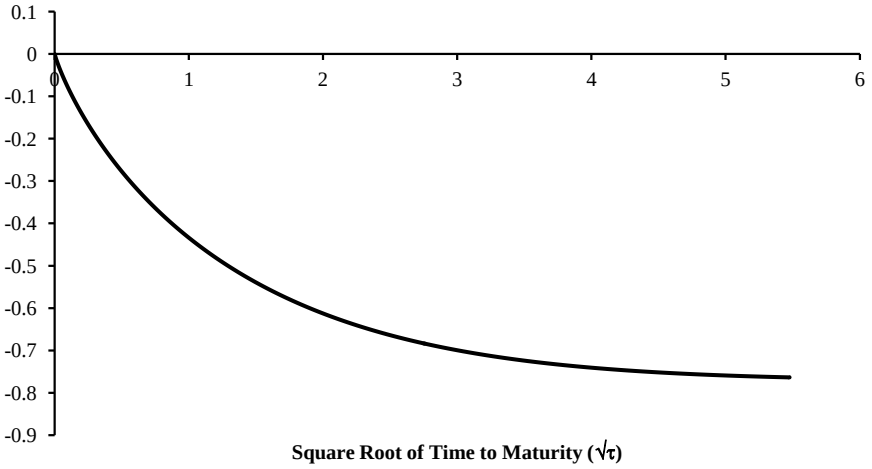
is even better suited for our purposes (and still easy to compute and invert), so we shall use this for our later numerical experiments.

To establish an interpolation grid $\{x_i\}_{i=0}^n$ for $H = H(x)$, an equidistant grid in x (or in x^2, for that matter) should *not* be used. Not only is such a grid prone to instabilities – the well-known *Runge phenomenon* – it rarely produces a competitive polynomial

[20] The simpler transformation $z = \tau - u$ is still convenient.

[21] Of course $\ln(B(\tau))^{2/(1-b)}$ would be even closer to a straight line, but working with non-integer powers in interpolation is typically not worth the additional computational cost.

Figure 6: Function G



Notes: The function G in (49), as a function of $\sqrt{\tau}$. Model settings are $r = q = 5\%$, $\sigma = 25\%$, $K = 130$.

approximation to the underlying function. Instead, we use *Chebyshev nodes*,

$$x_i = \frac{\sqrt{\tau_{\max}}}{2}(1 + z_i), \quad z_i = -\cos\left(\frac{i\pi}{n}\right), \quad i = 0, \ldots, n, \tag{50}$$

where $\tau_{\max}$ is the longest maturity for which we shall need to recover the exercise boundary. The z_i are just the extrema of the Chebyshev polynomial $T_n(z) = \cos(n \cos^{-1} z)$. It is known (see, e.g., [13]) that the Chebyshev node placement eliminates the Runge phenomenon. In addition, it is frequently near-optimal, in the sense that the resulting polynomial interpolant is close to the *minimax polynomial*, i.e. the nth order polynomial that has the smallest maximum deviation to the true function on $[0, \sqrt{\tau_{\max}}]$.

To concretely carry out interpolation of H between the Chebyshev nodes, let us normalize arguments to $[-1, 1]$ by writing $H(x) = q(z)$, $x = \sqrt{\tau_{\max}}(1 + z)/2$. It is known that the Chebyshev interpolant, q_C, to q can be expressed as

$$q_C(z) = \sum_{k=0}^{n} a_k T_k(z), \quad a_k = \frac{1}{2n}\left[q_0 + (-1)^n q_n\right] + \frac{2}{n}\sum_{i=1}^{n-1} q_i \cos\frac{ik\pi}{n}, \tag{51}$$

where

$$q_i = q(z_i) = H(x_i). \tag{52}$$

It can readily be verified that this relation ensures that $q_C(z_i) = q_i$, as desired. For an arbitrary $-1 < z < 1$, $q_C(z)$ can now be evaluated efficiently and stably using the

Clenshaw algorithm:[22]

$$\begin{aligned} b_n(z) &= a_n, \quad b_{n+1}(z) = 0, \\ b_k(z) &= a_k + 2zb_{k+1}(z) - b_{k+2}(z), \quad k = n-1, \ldots, 1, \\ q_C(z) &= a_0 + zb_1(z) - b_2(z). \end{aligned} \tag{53}$$

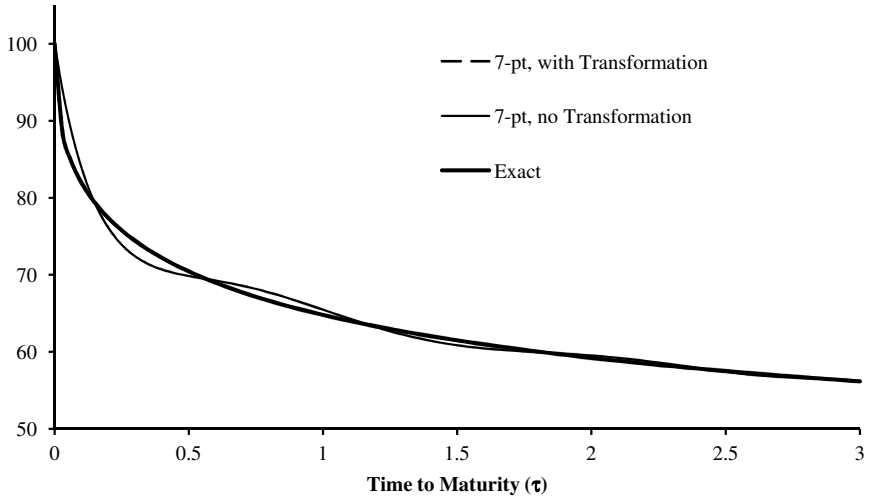
The grid $\{x_i\}_{i=0}^n$ establishes our interpolation nodes, and also defines our *collocation grid* $\{\tau_i\}_{i=0}^n$, i.e. the discrete points in τ-space at which (41) is run. Specifically, we have

$$\tau_i = x_i^2, \quad i = 0, \ldots, n, \tag{54}$$

and $B(\tau_0) = B(0+) = K \min(1, r/q)$.

Finally, for illustration Figure 7 below shows the exercise boundary interpolants from a naive low-dimensional collocation scheme applied directly on $\ln B$, compared to the one above applied to H. As one might have guessed from Figure 6, the latter avoids ringing and is overall far more accurate.

Figure 7: 3-Year Exercise Boundary



Notes: 7-point interpolated boundaries, with and without H-transformation. Model settings are $r = q = 5\%$, $\sigma = 25\%$, $K = 100$. The 7-point boundary with H-transformation cannot be distinguished from the exact boundary at the resolution of the figure.

5.5 Complete Algorithm, Computational Effort, and Convergence

Assume now that l, m, n are given, as is the desired maximum horizon $\tau_{\max}$. Let us summarize the complete algorithm for computing the optimal exercise boundary.

[22] This relation originates from the Chebyshev polynomial recurrence $T_n(z) = 2zT_{n-1}(z) - T_{n-2}(z)$.

1. Compute the Chebyshev nodes from (50). This establishes the collocation grid $\{\tau_i\}_{i=0}^n$, by (54).

2. Compute or look up the quadrature nodes y_k and weights w_k, for $k = 1, \ldots, l$.

3. Use one of the approximate methods (e.g., QD$^+$) referenced in Section 4.5 to establish an initial guess for[23] B on the n points $\{\tau_i\}_{i=1}^n$. Let the guess be denoted $B^{(0)}(\tau_i)$, $i = 0, \ldots, n$, with $B^{(0)}(\tau_0) = B(0+) = K \min(1, r/q)$.

4. For $j = 1$ to $j = m$, execute Steps 5-9 below.

5. Set $H(\sqrt{\tau}) = \ln\left(B^{(j-1)}(\tau)/X\right)^2$ and initialize the Chebyshev interpolation in Section 5.4 by computing the a_k in (51).

6. For each τ_i, $i = 1, \ldots, n$, use the Clenshaw algorithm (53) to establish (through q_C and H)
$$B^{(j-1)}\left(\tau_i - \tau_i\left(1 + y_k\right)^2/4\right), \quad k = 1, \ldots, l.$$

7. Use numerical quadrature (similar to (47)) to compute the integrals necessary to establish $N(\tau_i, B^{(j-1)})$ and $D(\tau_i, B^{(j-1)})$. Compute $f(\tau_i, B^{(j-1)})$, $i = 1, \ldots, n$.

8. Let $f'(\tau_i, B^{(j-1)})$ be defined as in (38). For $i = 1, \ldots, n$:

   (a) For a full Jacobi-Newton step, compute $N'(\tau_i, B^{(j-1)})$ and $D'(\tau_i, B^{(j-1)})$ per (39) and (40); the integral terms will be done by numerical integration, as in Step 7. Compute f' by (38).

   (b) For a partial Jacobi-Newton step, omit the integrals in (39) and (40). Compute f' by (38).

   (c) For an ordinary (Richardson) fixed point iteration, set $f' = 0$.

9. Compute $B^{(j)}(\tau_i)$, $i = 1, \ldots, n$, from (41).

10. We are done. The optimal exercise boundary at $\{\tau_i\}_{i=0}^n$ is approximated by $B(\tau_i) \approx B^{(m)}(\tau_i)$. For values of τ between nodes in $\{\tau_i\}_{i=0}^n$, the Chebyshev interpolant of $B^{(m)}$ (with a_k computed in Step 5) is used to approximate $B(\tau)$.

Upon completion of the algorithm, we are left with a continuous representation of $B(\tau)$ on the interval $[0, \tau_{\max}]$. This boundary may be cached for later use, or may be turned into put option prices at various maturities inside $[0, \tau_{\max}]$ by usage of (4). The integrals in (4) can be computed straightforwardly by any high-performance quadrature

[23] In the actual implementation, it is more efficient to work on H (49) than on B directly.

rule. In our numerical tests we found it was beneficial to apply the time transformation, $z = \sqrt{\tau - u}$, of section 5.2 to these integrals as well, so that we actually compute

$$V(\tau, S) = v(\tau, S) + 2rK \int_0^{\sqrt{\tau}} z\, e^{-r z^2} \Phi\left(-d_-\left(z^2, S/B(\tau - z^2)\right)\right)\, dz \\ - 2qS \int_0^{\sqrt{\tau}} z\, e^{-q z^2} \Phi\left(-d_+\left(z^2, S/B(\tau - z^2)\right)\right)\, dz \quad (55)$$

Let us briefly consider the computational effort of our scheme. Focusing first on the interpolation effort, for each boundary iteration, nl interpolations are required from (53), each involving an operation count of $O(n)$. In addition, the a_k must be computed once per boundary iteration and require $O(n^2)$ operations[24]. Note that there are only n distinct cosine values in the whole computation – or half that if we use $\cos(i\pi/n) = -\cos((n-i)\pi)/n$; these values may be computed once and stored for lookup, so the computation of the a_k is fast. For the total algorithm in Steps 1-10, we conclude that we have an interpolation cost of

$$C_{interp} = O(mn^2) + O(lmn^2) = O(lmn^2). \quad (56)$$

The additional cost of performing the boundary iteration involves a series of pre-processing steps (Steps 1-3), the operation count of which is $O(l) + O(n)$ and normally negligible. Computation of integrals by quadrature (Step 7) dominates the in-loop cost, and involves an operation cost of $O(ln)$ per boundary iteration. The total cost of numerical integration is therefore

$$C_{integral} = O(lmn). \quad (57)$$

Comparison of (56) and (57) shows that the interpolation cost nominally dominates the integration cost, for sufficiently large n. At realistic levels of n, however, one typically finds that (57) is larger than (56), due to the simplicity of the interpolation operations. Nevertheless, the form of (56) shows that the polynomial order n is perhaps more critical than the parameters l and m, and one should aim to keep it as low as possible (which we here do, in part, by executing the interpolation on the smooth function H in (49)).

Finally, a note on the expected convergence of our method. Provided that a) we use a spectral quadrature scheme (e.g., Gaussian quadrature); and b) the integrands are sufficiently smooth; then theoretically our algorithm would converge to the American option price at an exponential rate as l and n are increased. In practice, spectral collocation methods may or may not attain this theoretical ideal, but we would nevertheless expect the method to be highly competitive against, say, grid-based methods with algebraic convergence (see Footnote 10). This is no different from the basic application of, say, Gaussian quadrature in the numerical integration of smooth functions: the actual convergence order may or may not be at the peak theoretical limit – but virtually always much better than simple integration schemes (e.g., trapezoid). We empirically examine the convergence properties in Section 6.

[24] It is possible to reduce this to $O(n \log n)$ by using the fast cosine transform, but there is little reason to do this as the cost of the Clenshaw recurrence dominates (see (56)).

5.6 Notes to Algorithm

Let us briefly provide a few comments and suggestions to the algorithm outlined in Steps 1-10 above.

- It is not uncommon in the literature to use the Chebyshev nodes (Step 1) both for interpolation and for numerical integration, enforcing $l = n$. The resulting quadrature rule is known as *Clenshaw-Curtis quadrature* (see [73]). For flexibility and performance, however, our algorithm separates interpolation and integration, allowing for different node count and node placement; the computational overhead associated with this extension is minimal. Also, our tests found Gauss-Legendre quadrature (say) to always outperform Clenshaw-Curtis quadrature, although the differences between the methods were often relatively minor (consistent with the findings in [73]).

- In addition to Gauss-Legendre quadrature, we experimented with several integration methods, including Gauss-Kronrod quadrature. While other schemes may offer benefits when embedded in an adaptive quadrature algorithm, for exogenously specified l the simplicity and strong performance of the Gauss-Legendre quadrature makes it our default choice, especially for moderate values of l – say 10-15 or less. For high-precision option price estimates with large l (as in the tests in Section 6.1.1), we recommend usage of the *tanh-sinh quadrature rule*. While tanh-sinh quadrature is uncommon in finance applications, the method is robust and has strong convergence properties for integrands with singularities, see, e.g., [9].

- Our algorithm is designed to work with exogenously fixed values of l, m, n. In an industrial application, it is likely that one would upgrade the algorithm to work with stopping criteria based on exogenously specified tolerances. Adaptive integration rules would be useful for this. For our purposes in this paper – which focuses on testing and documentation – it is more relevant to work with fixed l, m, n.

- In Step 8, one has to choose between various flavors of Jacobi-Newton iteration and ordinary (Richardson) fixed point iteration. As there is some overhead associated with using Jacobi-Newton iteration, a reasonable strategy is often to use Jacobi-Newton stepping for the first one or two iterations, and then, when the boundary is close to the optimum value, switch to ordinary fixed point iteration. Given the result of Corollary 1, we expect the convergence of ordinary fixed point iteration to be rapid, once we are close to the root.

- If one has an accurate estimate for the convergence rate of American option prices as a function of (say) h, it is often possible to use Richardson extrapolation to ac-

celerate convergence[25]. For a spectral method (such as ours), the applicability of these techniques remain an open question. As we shall see, our algorithm converges so fast that usage of additional acceleration techniques are rarely, if ever, needed in practice.

- For the final conversion of the exercise boundary into option prices via (4), we recommend using the same integration rule as is used inside our collocations scheme, applied on the transformation (42). We also recommend using a relatively large number of integration nodes, p, especially if l and n are themselves large. A good rule of thumb is to use at least $p = l$ or even $p = 2l$ integration nodes for the final option value integral; the overhead of doing so is typically small relative to the cost of finding the exercise boundary in the first place.

6 Numerical Tests

In this section, we conduct a variety of tests of the algorithm in Section 5.5. Our emphasis is on the futures case $r = q$, covered in Section 6.1, but we also list results for other cases in Section 6.2. Most of our tests involve computing the optimal exercise boundary B at various configurations of the basic scheme, typically by varying the parameters n (the number of collocation nodes), m (the number of iterations), and l (the number of integration nodes). As the boundary itself is normally of less importance than American put option prices, the reported precision metric is mostly absolute and relative price errors of option prices.

Concretely, the method in 5 was coded in C++ and compiled with Visual C++ 2013 on a 2GHz PC. The default setup was as follows:

- QD^+ is used to establish the first guess for B at the n collocation nodes (Step 3).
- For the first iteration, we use a partial Jacobi-Newton step (option b. in Step 8). For subsequent iterations we use ordinary fixed point iteration (option c. in Step 8).
- l, m, n are specified exogenously, and no stopping criteria are used in the algorithm. We often set $l \approx 2n$.
- The number of integration nodes p used for computing the option price from a given boundary (through (4)) is also specified exogenously; we often set $p \approx 2l$.
- Unless otherwise specified, for benchmark "Exact" option values we use the fixed point method with $(l, m, n) = (131, 16, 64)$, $p = 131$, and tanh-sinh quadrature. We justify this choice in Section 6.1.1.

[25]That said, many market participants are generally concerned about the robustness of extrapolation methods, as they can sometimes produce poor sensitivities. Also, a small error in the estimation of convergence order can lead the extrapolation results astray.

We emphasize that our tests do *not* cache the computed exercise boundary for use with multiple options; rather, we recompute the boundary for each option (thereby setting $\tau_{\max}$ equal to the option maturity). For a truly optimized algorithm, implementation of a cache mechanism is a distinct possibility and could potentially lead to significant performance gains. On the other hand, systems overly reliant on caches may have difficulty with time-dependent parameters, an extension considered in Section 7.

6.1 Case $r = q$

6.1.1 PDEs and High Precision Baseline Values

For our later speed tests against the literature, we shall need to establish high-precision benchmark values which we can consider the "exact" American option values. While it is common in the literature to use a high-dimensional (e.g., 10,000 steps) binomial tree for this purpose, the resulting benchmarks are far too imprecise for our purposes here. Instead, we shall use our own algorithm (fixed point system A).

We initially considered using alternative methods for establishing benchmark values, but our tests showed that no other algorithm could produce sufficiently precise benchmark values quickly enough for the thousands of option tests we wanted to run. To show a typical result of our tests, consider using a production-quality Crank-Nicolson finite difference method to establish the benchmark value for a 1-year American put option. Table 1 below lists the American put premium[26] for various grid sizes, and shows that even a comparatively modest precision setting for our method will produce an option value that coincides to 7 significant digits with the price computed by a 250,000×250,000 step (!) finite difference grid. However, while our method completes its calculation in about 0.003 seconds, the finite difference grid takes about 40 minutes to run – a relative speed difference in the order of one million.

It can be verified that the finite difference grid solver used in Table 1 has convergence order around 2 for a moderate number of steps, but the convergence ultimately tapers off for large grids. In fact, adding more steps in the finite difference grid does *not* lead to meaningful precision improvements after one reaches about 100,000×100,000 nodes; instead, rounding errors cause results to oscillate randomly up and down at about the 7th or 8th digit after the floating point. The FP-A method, on the other hand, displays no such behavior, and the 10 or 11 first digits after the floating point rapidly stabilize; see Table 2.

For those that are curious about whether the finite difference grid perhaps becomes more competitive a lower precision requirements, we shall show more comparisons later on – but basically the answer is no. For instance, for the example above, the FP-A method with $(l, m, n) = (7, 2, 5)$ is 5 times more precise than the 500×500 finite difference grid, but runs more than 400 times faster than the finite difference grid.

[26] That is, the difference between the American and European put prices.

Method	American Premium	Relative Error	CPU Seconds
PDE 50	0.102249534103	4.6E-02	NA
PDE 100	0.105487101741	1.4E-02	2.9E-03
PDE 500	0.106872668265	7.5E-04	9.6E-03
PDE 1k	0.106927949843	2.3E-04	3.1E-02
PDE 5k	0.106950984474	1.6E-05	7.3E-01
PDE 10k	0.106952141410	5.3E-06	2.9E+00
PDE 50k	0.106952659976	4.0E-07	7.6E+01
PDE 250k	0.106952688738	1.3E-07	2.3E+03
FP-A (65,8,32)	0.106952702747	NA	3.0E-03

Table 1: Estimated 1-year American premium for $K = S = 100$. Model settings were $r = q = 5\%$ and $\sigma = 0.25$. The "PDE" method is a Crank-Nicolson finite difference grid with an equal number of asset steps and time steps, set as indicated in the table. The "FP-A" method is fixed point system A, with $(l, m, n) = (65, 8, 32)$ and $p = 101$ option price quadrature nodes; the integration scheme was tanh-sinh quadrature. Relative errors of the PDE method are measured against the FP-A premium.

(l,m,n)	p	American Premium	Relative Error	CPU Seconds
(5,1,4)	15	0.106783919132	1.5E-03	9.9E-06
(7,2,5)	20	0.106934846948	1.7E-04	2.3E-05
(11,2,5)	31	0.106939863585	1.2E-04	3.1E-05
(15,2,6)	41	0.106954833468	2.0E-05	4.5E-05
(15,3,7)	41	0.106952928838	2.1E-06	7.1E-05
(25,4,9)	51	0.106952731254	2.7E-07	1.7E-04
(25,5,12)	61	0.106952704598	1.7E-08	2.9E-04
(25,6,15)	61	0.106952703049	2.8E-09	4.6E-04
(35,8,16)	81	0.106952702764	1.6E-10	8.4E-04
(51,8,24)	101	0.106952702748	1.5E-11	2.1E-03
(65,8,32)	101	0.106952702747	8.5E-13	3.0E-03

Table 2: Estimated 1-year American premium for $K = S = 100$. Model settings were $r = q = 5\%$ and $\sigma = 0.25$. All numbers were computed using fixed point system A, with (l, m, n) and p as given in the table. Relative errors are measured against the American premium computed with $(l, m, n) = (201, 16, 64)$ and $p = 201$. Results for $(l, m, n) = (5, 1, 4)$ and $(l, m, n) = (7, 2, 5)$ were computed with Gauss-Legendre quadrature; all other results were computed with tanh-sinh quadrature.

6.1.2 Some Literature Comparisons

To expand the performance of our scheme against competing methods, we start off with a warm-up example on one of the more challenging option examples in [33], namely a 3-year option with $r = q = 4\%$, $K = 100$, and $\sigma = 0.2$. Results from a variety of algorithms are compared against this benchmark in Table 3 below.

At the given discretization levels, it is clear from the table that the most accurate method is our fixed point system A (FP-A) method, which is one or two orders of magnitude more precise than the FIK-F(400), Bin-BS, and FP-B methods which (roughly)

Spot S	True Price	Bin 1,000	Bin-BS 15,000	FIK-F 60	FIK-F 400	KJK 32	FP-A (10,3,7)	FP-B (10,3,7)
80	23.22834	23.22864	23.22837	23.22921	23.2284	23.22855	23.22834	23.22823
		3.1E-04	*3.1E-05*	*8.7E-04*	*6.1E-05*	*2.1E-04*	*2.4E-06*	*1.1E-04*
100	12.60521	12.60282	12.60529	12.60592	12.60526	12.60548	12.60521	12.60517
		2.4E-03	*7.8E-05*	*7.1E-04*	*4.8E-05*	*2.7E-04*	*2.2E-06*	*3.9E-05*
120	6.482425	6.48423	6.48247	6.48289	6.48246	6.48268	6.482425	6.482414
		1.8E-03	*4.5E-05*	*4.7E-04*	*3.5E-05*	*2.6E-04*	*1.5E-07*	*1.0E-05*

Table 3: Estimated 3-year put option price values for $K = 100$ and S as listed in the table. Model settings were $r = q = 4\%$ and $\sigma = 0.2$. First row: name of method; second row: number of steps used in algorithm. "Bin" is a regular binomial tree with 1,000 time steps; "Bin-BS" is an accelerated binomial tree with 15,000 time steps; "FIK-F" is the recommended method in [33], with the number of integration nodes set to 60 or 400; "KJK" is the method in [53] with 32 integration nodes; "FP-A" and "FP-B" is the method of this paper applied to fixed point systems A and B, respectively, $(l, m, n) = (10, 3, 7)$ and $p = 25$ (Gauss-Legendre quadrature). Italicized numbers are absolute price errors, compared against a high-precision "Exact" estimate (from FP-A with $(l, m, n) = (131, 16, 64)$). The "Bin-BS", "FIK-F" and "KJK" numbers were taken from Table 1 in [33].

tie for second place. More importantly, it is clear that the FP-A and FP-B methods are both much more computationally efficient than the FIK-F(400) and Bin-BS methods. For instance, according to [33], the average number of iterations used to establish the boundary is about 5 or more, hence the complexity of the FIK-F(400) method (excluding spline interpolation) is, as explained in Section 3.1, $O(5 \times 400 \times 400) = O(800,000)$. In contrast, the FP-A and FP-B methods have complexity (excluding polynomial interpolation) $O(3 \times 10 \times 7) = O(210)$ or about 3-4 orders of magnitude less than FIK-F(400). (The Bin-BS method has complexity $O(15,000^2)$ and is not competitive from an efficiency perspective).

Note that clean timing comparisons of FP-A/B against FIK-F could not be accomplished due to several issues, including the fact that [33] parallelized[27] their algorithms on Matlab, whereas we used a C++ algorithm on a single CPU. Nevertheless, it appears (from their Table 4) that the FIK-F(400) routine prices about 5 options per second. As we shall see later, in Table 6, the FP-A/B algorithm is about 3-4 orders of magnitude faster than this, at comparable or better accuracy levels.

In Table 4 we extend the analysis to all 12 options in Tables 2-3 of [33]. Rather than reporting individual pricing errors, we here just report the RMS and relative RMS (RRMS) errors across all option prices. For reference, we also include results from a finite difference grid solver.

The conclusions here are similar to those above: at the chosen discretization levels, the FP-A method is the most accurate, with the FP-B and 1,000-step PDE solver tied for second place. Once again, the computational complexity of the FP-A/B method is far less

[27] While the authors state (p. 22) that their method is "fast because iterations are performed in parallel", it is hard to figure out precisely which of their timing results relied on parallelization and which ones did not. Additionally, it is not stated how many cores were used for those results that relied on parallel processing.

	PDE 100	PDE 1,000	Bin 1,000	Bin 10,000	FIK-F 60	FIK-F 400	KJK 32	FP-A (10,3,7)	FP-B (10,3,7)
RMS	2.9E-03	4.6E-05	2.7E-04	3.0E-04	9.3E-04	6.5E-05	2.8E-03	6.8E-06	7.4E-05
RRMS	1.5E-04	2.7E-06	8.1E-04	8.8E-05	2.7E-04	1.9E-05	8.3E-04	2.5E-07	2.6E-06

Table 4: Estimated 0.5- and 3-year put option price errors (RMS and RRMS) for $K = 100$ and S equal to 80, 100, or 120. Model settings were $r = q = 4\%$ and σ either 0.2 or 0.5. The "PDE" method is a Crank-Nicolson finite difference grid with an equal number of asset steps and time steps, set as indicated in the table. The first and second row naming conventions and settings are otherwise identical to those used in Table 3. The "FIK-F" and "KJK" numbers were extracted from Tables 2 and 3 in [33].

than its competitors.

6.1.3 Timing and Convergence

To expand on our timing and convergence results, we now move away from individual option tests and instead focus on a large set of option payouts and model parameters. This set is found by generating all possible combinations of the parameter ranges in Table 5.

Parameter	Range
$r = q$	{2%,4%,6%,8%,10%}
S	{25,50,80,90,100,110,120,150,175,200}
T	{1/12,0.25,0.5,0.75,1,2,3}
σ	{0.1,0.2,0.3,0.4,0.5,0.6}

Table 5: Model and contract parameter ranges for timing and precision tests in Tables 6-7. In all cases $K = 100$.

In Table 6 below, we show select timing and error statistics for FP-A, on the set of options in Table 5; notice that we extended the error reporting to now also include maximum absolute error (MAE) and maximum relative error (MRE). For brevity, we omitted the FP-B numbers which were, on average, about 5-10 times less precise and had similar computation times. For better intuition about the timing numbers, Table 7 lists errors and computation times for a Crank-Nicolson finite difference grid method, a popular method used in many banks.

Tables 6 and 7 confirm our earlier observation: the FP-A method outperforms the finite difference solver at all levels and measures of precision, with the outperformance growing rapidly as a function of required precision. For instance, at a (fairly crude) RMSE precision tolerance of approximately 0.5×10^{-3}, the FP-A method is around 4-500 times faster than a PDE solver; but at a (high) precision tolerance of 10^{-6}, the FP-A method is about 25,000 times faster than the finite difference grid. We note in passing that if one is happy with the 10^{-3} RMSE threshold, our (only casually optimized) FP-A routine running on a middle-of-the-road PC has a throughput in the order of 100,000 option prices per second.

(l,m,n)	(3,1,2)	(5,1,4)	(7,2,6)	(12,3,8)	(20,4,10)	(33,5,12)	(41,6,16)
p	5	8	15	25	50	65	81
RMSE	7.2E-03	3.6E-04	3.3E-05	4.2E-06	3.3E-07	6.6E-08	7.5E-09
RRMSE	4.2E-04	1.4E-05	2.7E-06	3.6E-07	2.3E-08	1.8E-09	1.8E-10
MAE	6.2E-02	3.2E-03	4.1E-04	3.7E-05	2.7E-06	1.5E-06	1.6E-07
MRE	2.7E-03	8.7E-05	2.3E-05	2.8E-06	1.8E-07	2.9E-08	3.3E-09
Opt/Sec	192,307	96,154	37,594	13,928	5,734	2,732	1,399

Table 6: American put errors and timing results for fixed point system A ("FP-A"), using various values of l, m, n, p and covering the parameter ranges in Table 5. The last two columns used tanh-sinh quadrature; all other columns used Gauss-Legendre quadrature. The "Opt/Sec" row lists average calculation times in option prices per second. As in [33], we removed very low option prices from the computations; the minimum price threshold was set to 0.5. This left a total of 1,675 options in the test set.

PDE steps	250	500	1,000	5,000	10,000
RMSE	4.4E-04	1.2E-04	3.5E-05	2.5E-06	8.6E-07
RRMSE	3.3E-05	1.0E-05	3.4E-06	2.7E-07	9.3E-08
MAE	4.2E-03	9.8E-04	2.1E-04	8.6E-06	2.9E-06
MRE	1.1E-04	3.8E-05	1.3E-05	1.2E-06	4.4E-07
Opt/Sec	270	109	32	1.4	0.35

Table 7: American put errors and timing results for a Crank-Nicolson finite difference grid method, covering the parameter ranges in Table 5. The number of time and asset steps are set equal, at the values listed in the table. As in Table 6, options with prices less than 0.5 were removed from the set.

6.2 General Case

We now turn to the general case, where no longer necessarily $r = q$. Unless $r \gg q$, it is still the case that the FP-A method is more accurate than FP-B, by roughly one order of magnitude. However, as we hinted at earlier in Section 3.3, the FP-A method has a tendency to generate oscillations – and sometimes outright blow-up – for strongly drift-dominated dynamics, i.e. when $r - q$ is large relative to σ. This behavior is somewhat similar to, say, that of many finite difference methods; see, e.g., [70]. An earlier version of this paper discussed how to eliminate the FP-A oscillations by introducing dampening factors in the fixed point iterations, but introduction of such remedies sometimes renders the FP-A method slower (and more complex) than the FP-B method. A safe and robust recommendation is instead to always *use the FP-B method for the non-futures case*, i.e. whenever $r \neq q$. While one can obviously do better through intelligent branching between FP-A and FP-B, say, the speed and robustness of the FP-B method most likely will make such complications unnecessary in practice. For simplicity, all tests below are therefore based on the FP-B method.

6.2.1 Bulk Test

In our first test, we first consider a bulk setup similar to the one in Section 6.1.3. For this, we use the following parameter ranges:

Parameter	Range
r	{2%,4%,6%,8%,10%}
q	{0%, 4%, 8%, 12%}
S	{25,50,80,90,100,110,120,150,175,200}
T	{1/12,0.25,0.5,0.75,1}
σ	{0.1,0.2,0.3,0.4,0.5,0.6}

Table 8: Model and contract parameter ranges for timing and precision tests in Tables 9-10. In all cases $K = 100$.

We report errors and timing results for method FP-B in Table 9 below. For comparison, Table 10 lists results for a binomial tree and a finite difference grid.

(l,m,n)	(5,2,4)	(8,4,6)	(21,6,10)	(25,8,12)	(31,16,16)
p	8	15	41	51	61
RMSE	7.5E-04	1.2E-04	1.9E-06	3.6E-07	4.5E-08
RRMSE	1.3E-04	2.5E-05	3.4E-07	7.7E-08	4.3E-09
MAE	5.0E-03	6.9E-04	4.2E-05	1.1E-05	1.2E-06
MRE	1.2E-03	2.3E-04	1.1E-05	3.6E-06	1.3E-07
Opt/Sec	68,027	20,001	3,650	2,058	641

Table 9: Various error measures on American put prices for fixed point method B (FP-B), using various values of l, m, n, p. Numbers in first two columns used Gauss-Legendre quadrature; remaining numbers used tanh-sinh quadrature. Parameter ranges were as in Table 8, but with all option prices less than 0.5 removed from the data set. This left a total of 4,495 options in the test set.

Method	Bin	Bin	PDE	PDE	PDE	PDE	PDE
Steps	100	1,000	250	500	1,000	5,000	10,000
RMSE	1.4E-02	1.4E-03	4.2E-04	1.2E-04	3.7E-05	3.1E-06	1.2E-06
RRMSE	3.5E-03	2.9E-04	6.7E-05	2.1E-05	6.9E-06	6.6E-07	2.7E-07
MAE	5.9E-02	6.0E-03	2.1E-03	6.3E-04	1.5E-05	1.3E-05	5.3E-06
MRE	3.5E-02	3.3E-03	5.7E-04	1.6E-04	4.7E-05	4.3E-06	1.8E-06
Opt/Sec	8,718	603	270	104	32	1.37	0.34

Table 10: Various error measures on American put prices for a binomial tree ("Bin") and for a Crank-Nicolson finite difference grid ("PDE"). The finite difference grid is set to have an equal number of time- and asset-steps. The set of options was identical to that used in Table 9.

As evidenced by the tables, our earlier conclusions carry over to the case $r \neq q$: for a given level of precision, the fixed point method is several orders of magnitude faster than binomial trees and finite difference grids.

6.2.2 Literature Comparison: Fast Trees

For low- to medium- precision levels, there is evidence that binomial trees can be effectively "tuned" to the problem of American option pricing. For instance, [47] and [48] recently presented binomial tree results that by some[28] are apparently considered the current benchmarks for high-efficiency American option pricing in the Black-Scholes model. [47] ranks 220 different binomial tree methods, including Leisen-Reimer and Tian trees, as well as a multitude of extrapolation and smoothing techniques. [48] refines the best-performing method further, for an additional 50% performance gain, to arrive at what the authors characterize as "the most efficient known numerical method to value American put options in the Black-Scholes model at the important accuracy levels of 10^{-3} to 10^{-4}". Table 6.2.2 below compares[29] our fixed-point method B to the timing results in [48].

RMS Target	Best Tree	FP-B
1.00E-03	725.3	51,813
5.00E-04	350.0	35,714
1.00E-04	60.3	11,655

Table 11: American put pricing speed, in options per second, for a given maximal RMS error target. The test set consists of 2,000 American out options, generated randomly by the prescription in [48]; note that $q = 0$ for all options. The "Best Tree" results were the fastest results reported in Figure 9 of [48]; the FP-B results were computed using fixed point method B with Gauss-Legendre quadrature. The (l, m, n, p) settings used were (5,2,5,11), (6,3,5,15), and (13,4,7,31); the resulting RMS errors were 9.2e-4, 4.0e-4, and 8.6e-5, respectively.

As the table demonstrates, for the RMS precision range of $10^{-3} - 10^{-4}$ to which the trees in [48] were specifically tuned, the FP-B method is here on average about 120 times more efficient than the best results in [48]. Of course, should one desire tighter tolerances than in [48], then the speed advantage of the FP-B method would become even more pronounced, as we experienced in Section 6.1.1. For very crude tolerances (e.g., 10^{-2}), the QD+ first boundary guess is typically sufficient (i.e. $m = 0$), which of course outperforms any tree algorithm by a substantial margin.

7 Extensions

While our numerical scheme was developed for American puts (and calls, through put-call symmetry) on an underlying following a constant-parameter log-normal process, certain extensions to both the payout and to the process are possible. For instance, [4] and [7] (among others), discuss applications of integral equation to American and Bermudan swaptions in Gaussian or near-Gaussian interest rate models. We discuss a few other extensions here.

[28] We thank a referee for pointing this out to us.

[29] [47] and [48] use a 3GHz Pentium 4 processor, which for a low-memory single-threaded analytics application should be comparable our 2GHz Xeon processor.

7.1 American Exchange Options

An *American Exchange Option* on two underlyings, S_1 and S_2, has the payout

$$p(v) = (c_1 S_1(v) - c_2 S_2(v))^+$$

where c_1 and c_2 are non-zero constants, and where $v \in [0, T]$ is the time at which the option contract gets exercised by its holder. If we assume that S_1 and S_2 are correlated GBM processes with constant volatilities, drifts, and correlation, then a simple measure change allows one (see [15]) to recast the American exchange option as an ordinary American put option which may then readily be priced by the methods in Section 5.

7.2 Time-Dependent Parameters

Consider now again the regular American put, but assume that r, q, and σ in (1) are no longer constants but instead smooth, well-behaved deterministic functions of time:

$$dS(t)/S(t) = (r(t) - q(t))\, dt + \sigma(t)\, dW(t). \tag{58}$$

For a T-maturity American put, there will, as in (3), exist an optimal T-indexed critical stock price function $S_T^*(t)$, below which the option should be exercised. With time-dependent parameters, we may no longer represent the critical stock price as a boundary function B depending solely on $T-t$. However, we can always tag the boundary function with a specific maturity index T, writing $B_T(\tau) = S_T^*(t)$, with $\tau = T - t$. As this boundary function now only applies to a single maturity T, it is less natural to work in reversed time than was the case for time-homogenous parameters; nevertheless, for consistency and comparison with earlier results, we will work with B_T, rather than with S_T^*. Notice that introduction of time-dependency invalidates usage of a single exercise boundary function for all maturities, something that is rarely of consequence in practical applications where it is common to compute the exercise boundary anew for each option, rather than use a cached "universal" boundary for all maturities (see the discussion at the beginning of Section 6).

For the process (58) we may rely on (6) to establish the put value. To simplify notation, define

$$P(t,T) = e^{-\int_t^T r(u)\,du}, \quad Q(t,T) = e^{-\int_t^T q(u)\,du}, \quad \Sigma(t,T) = \int_t^T \sigma(u)^2\, du,$$

which allows us to write, for the T-maturity put price, with $\tau = T - t$,

$$\begin{aligned} V_T(\tau, S) &= v_T(\tau, S) + \int_t^T \mathrm{E}\left(P(t,s)\,(r(s)K - q(s)S(s)) \cdot 1_{S(s)<B_T(T-s)}\,|S(t) = S\right)\, ds \\ &= v_T(\tau, S) + \int_0^\tau P(T-\tau, T-u) K r(T-u) \Phi\left(-d_-\left(\tau - u, S/B_T(u), T-\tau\right)\right)\, du \\ &\quad - \int_0^\tau Q(T-\tau, T-u) S q(T-u) \Phi\left(-d_+\left(\tau - u, S/B_T(u), T-\tau\right)\right)\, du. \end{aligned} \tag{59}$$

Here, both the American put price (V_T) and the European put price (v_T) are sub-indexed with the maturity T to reflect the time-dependence in the process parameters. Also, it is necessary to re-define d_+ and d_- to have an extra time argument:

$$\begin{aligned} d_\pm(\delta, z, t) &= \frac{\ln z + \int_t^{t+\delta} (r(u) - q(u))\, du \pm \frac{1}{2}\int_t^{t+\delta} \sigma(u)^2\, du}{\sqrt{\int_t^{t+\delta} \sigma(u)^2\, du}} \\ &= \frac{\ln\left(z \cdot Q(t, t+\delta)/P(t, t+\delta)\right) \pm \frac{1}{2}\Sigma(t, t+\delta)}{\sqrt{\Sigma(t, t+\delta)}}, \end{aligned}$$

with the European put price now being

$$v_T(\tau, S) = P(t,T)K\Phi\left(-d_-\left(\tau, S/K, T-\tau\right)\right) - Q(t,T)S\Phi\left(-d_+\left(\tau, S/K, T-\tau\right)\right).$$

Using (59) at $S = B_T(\tau)$, we get a boundary representation similar to (12). Further rearranging (as in (14)), we get the time-dependent version of fixed point system B (23)-(24):

$$B_T(\tau) = K\frac{P(T-\tau, T)}{Q(T-\tau, T)}\frac{N_T(\tau, B_T)}{D_T(\tau, B_T)}, \tag{60}$$

with

$$\begin{aligned} N_T(\tau, B_T) &= \Phi\left(d_-(\tau, B_T(\tau)/K, T-\tau)\right) \\ &\quad + \int_0^\tau r(T-u)\frac{P(T-\tau, T-u)}{P(T-\tau, T)}\Phi\left(d_-\left(\tau-u, B_T(\tau)/B_T(u), T-\tau\right)\right) du, \end{aligned} \tag{61}$$

$$\begin{aligned} D_T(\tau, B_T) &= \Phi\left(d_+(\tau, B_T(\tau)/K, T-\tau)\right) \\ &\quad + \int_0^\tau q(T-u)\frac{Q(T-\tau, T-u)}{Q(T-\tau, T)}\Phi\left(d_+\left(\tau-u, B_T(\tau)/B_T(u), T-\tau\right)\right) du. \end{aligned} \tag{62}$$

Not surprisingly, (61)-(62) generalizes (23)-(24) by, essentially, replacing the integration kernel terms $\sigma^2 \cdot (\tau - u)$, re^{ru} and qe^{qu} with

$$\Sigma\left(T-\tau, T-u\right), \quad r(T-u)\frac{P(T-\tau, T-u)}{P(T-\tau, T)}, \quad q(T-u)\frac{Q(T-\tau, T-u)}{Q(T-\tau, T)}, \tag{63}$$

respectively.

The substitutions in (63) carry over directly to fixed point system A; for brevity we omit the obvious expressions.

Numerical computation of the boundary B_T from (60) and (61)-(62) (or a version for fixed point system A) can be done by a straightforward adjustment of the spectral collocation method in Section 5, essentially by modification of the integration kernels as indicated above. We trust that the reader can complete the scheme, but offer several remarks.

First, when making an initial guess for $B_T(\tau)$, it is useful to compute approximating constant values of r, q, and σ, for usage in either QD$^+$ or the analytical approximation scheme in Section 4.5. For instance, when pricing the option at time $t = 0$, we might use parameter averages computed as

$$\bar{x} = \frac{1}{\bar{T}} \int_0^{\bar{T}} x(u)\, du, \quad x = r, q, \sigma^2, \tag{64}$$

where the dimensioning time horizon $\bar{T}$ could, for instance, be set to T or to some fraction of T (e.g., $T/2$).

Second, with the inclusion of additional time-dependent terms in the integrals, it is likely that one, for a given level of precision, needs more collocation nodes (n) and integration steps (m) than in the constant-parameter setup[30]. How much additional work is required would depend on how strongly r, q, and σ depend on time, so, while brute force is always an option, a sophisticated algorithm would likely be adaptive in its choice of n and m, with a user-specified stopping criterion. It should be noted that some quadrature and interpolation rules are better suited for adaptive algorithms than others, in the sense that these schemes make adding incremental nodes to an existing grid straightforward. The Chebyshev spacing (50) is well-suited for an adaptive algorithm, but the Gauss-Legendre quadrature method is not; instead one could use Clenshaw-Curtis or Gauss-Kronrod quadrature. As mentioned earlier, the former method conveniently uses Chebyshev spacing and, despite some theoretical drawbacks, appear to offer practical performance that is nearly as good as Gauss-Legendre quadrature.

Finally, as the initial guess based on (64) is likely less accurate than in the constant-parameter case, one would expect to need more iterations l to converge to an accurate solution of the collocation iteration. Again, it might be useful to introduce a user-specified stopping criterion that stops the iterations when improvements are below a certain absolute or relative error threshold.

We note that one extreme case of time-dependence occurs if dividends (q) are paid discretely, rather than continuously in time. For this case, however, exercise the boundary properties change materially, and the algorithm outlined in this paper requires substantial modifications. For this case, we refer to [6].

7.3 Other Extensions

So far we have only dealt with log-normal processes, but (6) applies to any sufficiently regular SDE for S. In practice, however, we would need substantial analytical tractability in order to use the scheme[31]. For instance, if r and q are deterministic, we need closed-form expressions – or at least fast and accurate approximations – for the two quantities ($s > t$)

$$\mathrm{E}\left(1_{S(s)<S_T^*(s)} \,|S(t) = S\right), \quad \mathrm{E}\left(S(s) \cdot 1_{S(s)<S_T^*(s)} \,|S(t) = S\right). \tag{65}$$

[30] This effect will obviously affect all numerical routines, not just the one in this paper.

[31] For general SDEs, a finite difference method would most likely be the method of choice.

Besides the log-normal specification, a specific model where this is possible is the CEV process in [34]

$$dS(t) = \mu S(t)\,dt + \sigma S(t)^p\,dW(t)$$

or the displaced diffusion process

$$dS(t) = \mu S(t)\,dt + \sigma(bS(t) + (1-b)S(0))\,dW(t).$$

The development of collocation schemes for these processes (and for their extensions to time-dependent parameters) follows closely the steps in Sections 5 and 7.2.

Extensions to S-processes that include jumps are also possible, although the decomposition (4) of the American option price will now contain a new term originating from the fact that S may move between the exercise and continuation regions in discrete jumps that cross over the boundary; see [65] and [41] for the details. Numerical methods for jump-diffusion case are still in relative infancy, and application of the methods in Section 5 is an interesting area of future research.

Finally, let us touch upon the extensions to stochastic volatility models, such as the Heston process in [42]. For this type of process, the exercise boundary B will depend on the state of a stochastic volatility driver ϑ as well as on time to maturity, a significant increase in both dimension and complexity. Several authors (e.g., [25]) have attempted to simplify the problem by simplifying this dependency

$$\ln B(\tau, \vartheta) \approx b_0(\tau) + \vartheta b_1(\tau).$$

With this *ansatz*, a computationally efficient decomposition similar to (4) is possible[32], although the necessary terms in (65) will have to be computed by potentially slow inverse Fourier methods. The competitiveness of the resulting scheme against, say, modern finite difference methods is, in our opinion, still an open question.

8 Conclusion

In this paper, we have introduced a new algorithm for very fast computation of American put and call option prices in a log-normal (Black-Scholes) diffusion setting. The method proposed here is based on an integral equation formulation for the fixed boundary, an approach that recently has seen renewed interest after promising results in [53] and [33]. Combining modern methods for integral equations with a set of carefully designed transformations of the boundary, we are able to speed up computational efficiency by several orders of magnitude, compared with competing approaches. Our method is straightforward to implement, fast enough for real-time ticking risk systems, and, if properly tuned, capable of producing high-precision price estimates at a rate of many 10,000's options per second per CPU.

[32] For each τ, [25] consider two carefully chosen levels of ϑ, to form a linked system of integral equations for b_0 and b_1.

While a significant improvement of all known (to us) speed and accuracy records of a classical benchmark problem should, in itself, be of interest to many, the primary objective was a concrete practical one: the real-time risk management of American option portfolios. As discussed in Section 1, the Black-Scholes dynamics remain a standard reference model for virtually all practical trading activity in American puts and calls, especially on exchanges and for short- and medium-dated OTC structures.

While the development of our method, as well as our test cases, focused on the Black-Scholes dynamics, extensions are certainly possible, as we briefly discussed in Section 7. An obvious avenue for future research would here be a fuller development (and testing) of the various extensions in Section 7, perhaps especially the challenging problems of handling jumps, discrete dividends, and stochastic volatility. Extensions to more payouts more complicated than puts and calls is also an interesting problem.

Beyond these topics, certain questions about the constant-parameter log-normal case fell outside the scope of this paper, and could be considered in future work. For instance, the logic (and level of adaptiveness) around choosing values for l, m, n, p to optimize the precision-speed trade-off for any specific option was only addressed cursorily in our paper through rough "rules of thumb"; while this may be adequate for many applications, a more sophisticated analysis would surely improve our computational results even further. Finally, it remains an open question whether any of the many different possible integral equations for the exercise boundary (see Section 2.3) may have even better numerical properties than those considered in our paper.

References

[1] C. Ahn, H. Choe and K. Lee (2009), "A Long Time Asymptotic Behavior of the Free Boundary for an American Put," *Proceedings of AMS*, 137, 3425-3436.

[2] AitSahlia, F. and P. Carr (1997), "American Options: A Comparison of Numerical Methods," in *Numerical Methods in Finance*, 67-87.

[3] AitSahlia, F. and T. Lai (2001), "Exercise Boundaries and Efficient Approximation to American Option Prices and Hedge Parameters," *Journal of Computational Finance*, 4, 85-104.

[4] Andersen, L. (2007), "Notes on fast Bermudan Swaption Pricing with Fees," Working Paper, Bank of America.

[5] Andersen, L. and M. Broadie (2004), "Primal-Dual Simulation Algorithm for Pricing Multidimensional American Options," *Management Science*, 50, 1222-1234.

[6] Andersen, L., M. Lake, and D. Offengenden (2014), "High Performance American Option Pricing: Discrete Dividends," Working Paper, Bank of America, *in preparation*.

[7] Andreasen, J. (2007), "The Fastest Bermudan Pricer in the West," Working Paper, Bank of America .

[8] Atkinson, K. (1992), "A Survey of Numerical Methods for Solving Nonlinear Integral Equations," *Journal of Integral Equations and Applications*, 4 (1), 15-46.

[9] Bailey, D., X. Li and K. Jeyabalan (2005), "A Comparison of Three High-Precision Quadrature Schemes," *Experimental Mathematics*, 14 (3), 317âĂŞ329.

[10] Barles, G., J. Burdeau, M. Romano and N. Samsoen (1995), "Critical Stock Price Near Expiration," *Applied Mathematical Finance*, 5, 77-85.

[11] Barone-Adesi, G. and R.J. Elliott (1991), "Approximations for the Values of American Options," *Stochastic Analysis and Applications*, 9(2), 115-131.

[12] Barone-Adesi, G., and R. Whaley (1987), "Efficient Analytical Approximation of American Option Values," *Journal of Finance*, 42, 301-320.

[13] Berrut J.-P. and N. Trefethen (2004), "Barycentric Lagrange Interpolation," *SIAM Review*, 46 (3), 501-517.

[14] Bjerksund, P. and G. Stensland (1993), "Closed Form Approximation of American Options," *Scandinavian Journal of Management*, 9, Suppl., 87-99.

[15] Bjerksund, P. and G. Stensland (1993), "American Exchange Options and a Put-Call Transformation: A Note," *Journal of Business Finance & Accounting*, 20(5), 761 - 764.

[16] Bayraktar, E. and H. Xing (2009), "Analysis of the Optimal Exercise Boundary of American Options for Jump Diffusions," *SIAM Journal on Mathematical Analysis*, 41 (2), 825-860.

[17] Brennan, M. and E. Schwartz (1978), "Finite Difference Methods and Jump Processes Arising in the Pricing of Contingent Claims: a Synthesis," *Journal of Financial and Quantitative Analysis*, 3 (13), 461–474.

[18] Broadie, M. and J.B. Detemple (1996), "American Option Valuation: New Bounds, Approximations, and a Comparison of Existing Methods," *Review of Financial Studies,* 9, 1211-1250.

[19] Brunner, H. (2004), *Collocation Methods for Volterra Integral and Related Functional Equations*, Cambridge University Press.

[20] Brunner, H. (1985), "The Numerical Solution of Weakly Singular Volterra Integral Equations by Collocation on Graded Meshes," *Mathematics of Computation*, 172 (42), 417-437.

[21] Brunner, H. (1984), "The Numerical Solution of Integral Equations with Weakly Singular Kernels," in

Numerical Analysis, Dundee 1983 (D. F. Griffiths, ed.), Lecture Notes in Mathematics, 1066, Springer-Verlag, 50-71.

[22] Bunch, D. and H. Johnson (2000), "The American Put Option and its Critical Stock Price," *Journal of Finance*, 55(5), 2333-2356.

[23] Carr, P. (1998), "Randomization and the American put," *Review of Financial Studies*, 11, 597-626.

[24] Carr, P. and D. Faguet (1994), "Fast Accurate Valuation of American Options," Working Paper, Cornell University.

[25] Ziogas, A and Chiarella, C. (2005), "Pricing American Options under Stochastic Volatility," Computing in Economics and Finance, 77, Society for Computational Economics.

[26] Chadam, J. and X. Chen, "Analytical and Numerical Approximations for the Early Exercise Boundary for American Put Options (2003)," *Dynamics of Continuous, Discrete and Impulsive Systems*, Series A: Math. Anal., 10, 649-660.

[27] Chen, X. and J. Chadam (2007), "A Mathematical Analysis for the Optimal Exercise Boundary of American Put Options," *SIAM Journal of Mathematical Analysis*, 38, 1613–1641.

[28] Chen, X., H. Cheng and J. Chadam (2013), "Non-Convexity of the Optimal Exercise Boundary for an American Put Option on a Dividend-Paying Asset, *Mathematical Finance*, 23(1), 169-185.

[29] Chen, X., H. Cheng and J. Chadam (2011), "Far-From-Expiry Behavior of the American Put Option on a Dividend-Paying Asset," *Proceedings of the AMS*, 1 39, 273–282.

[30] Cheng, J. and J. Zhang (2012), "Analytical pricing of American Options," *Review of Derivatives Research*, 15, 157–192.

[31] Chadam, J., "Integral Equation Methods for Free Boundary Problems," *Encyclopedia of Quantitative Finance*.

[32] Cook, J. (2009), *Asymptotic Analysis of American-Style Options*, PhD Thesis, University of Bath.

[33] Cortazar, G., L. Medina, and L. Naranjo (2013), "A Parallel Algorithm for Pricing American Options," Working Paper, Pontificia Universidad Católica de Chile.

[34] Cox, J. (1975), "Notes on Option Pricing I: Constant Elasticity of Variance Diffusions," Working Paper, Stanford University.

[35] Cox, J., S. Ross, and M. Rubinstein (1979), "Option Pricing: A Simplified Approach," *Journal of Financial Economics*, 7 (3), 229-263.

[36] Elnagar, G. and M. Kazemi (1996), "Chebyshev Spectral Solution of Nonlinear Volterra-Hammerstein Integral Equations," *Journal of Computational and Applied Mathematics*, 147-158.

[37] Evans, J.., R. Kuske and J. Keller (2002), "American Options on Assets with Dividends Near Expiry," *Mathematical Finance*, 12, 219–237.

[38] Forsyth, P. and K. Vetzal (2002), "Quadratic Convergence of a Penalty Method for Valuing American Options," *SIAM Journal on Scientific Computation*, 23, 2096-2123.

[39] Frontczak, R. (2013), "Simple Analytical Approximations for the Critical Stock Price of American Options," Working Paper, Landesbank Baden-Wurttemberg.

[40] J. Goodman and D. Ostrov (2002), "On the Early Exercise Boundary of the American Put Option," *SIAM Journal of Applied Mathematics*, 62 (5), 1823-1835.

[41] C. Gukhal (2001), "Analytical Valuation of American Options on Jump-Diffusion Processes," *Mathematical Finance*, 97-115.

[42] Heston, S. (1993), "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Option," *Review of Financial Studies*, 6(2), 327-343.

[43] Huang, J.-Z., M. Subrahmanyam, and G. Yu (1996), "Pricing and Hedging American Options: A Recursive Integration Method," *Review of Financial Studies*, 9, 277–300.

[44] Jacka, S. (1991), "Optimal Stopping and the American Put," *Mathematical Finance,* 1, 1-14.

[45] Jamshidian, F. (1992), "An Analysis of American Options," *Review of Futures Markets*, 11, 72-80.

[46] Johnson, H. (1983), "An Analytic Approximation for the American Put Price," *Journal of Financial and Quantitative Analysis*, 18 (1), 141-148.

[47] Joshi, M. (2009), "The Convergence of Binomial Trees for Pricing the American Put," *Journal of Risk*, 11 (4), 87-108.

[48] Chen, T. and M. Joshi (2010), "Truncation and Acceleration of the Tian Tree for the Pricing of American Put Options," Working Paper, University of Melbourne, Australia.

[49] Ju, N. (1998), "Pricing an American Option by Approximating its Early Exercise Boundary as a Multi-Piece Exponential Function," *Review of Financial Studies*, 11, 627-646.

[50] Ju, N. and R. Zhong (1999), "An Approximate Formula for Pricing American Options," *Journal of Derivatives*, 7, 31-40.

[51] Kallast, S. and A. Kivinukk (2003), "Pricing and Hedging American Options using Approximations by Kim Integral Equations," *European Finance Review,* 7 (3), 361–383.

[52] Kim, T. (1990), "The Analytical Valuation of American Options," *Review of Financial Studies*, 3, 547-672.

[53] Kim, I., B.-G. Jang, and K. Kim (2013), "A Simple Iterative Method for the Valuation of American Options," *Quantitative Finance,* 13 (6), 885–895.

[54] Kwok, Y.-K. (2008), *Mathematical Models of Financial Derivatives*, 2nd Edition, Springer Verlag.

[55] Kuske R. and J. Keller (1998), "Optimal Exercise Boundary for an American Put Option," *Applied Mathematical Finance*, 5, 107-116.

[56] Lee, J. and D. Paxson (2003), Continued Exponential Approximations for the Valuation of American Options," *European Journal of Finance*, 9, 449-474.

[57] Leisen, D. and M. Reimer (1996), "Binomial Models for Option Valuation: Examining and Improving Convergence", *Applied Mathematical Finance*, 3 (4), 319-346.

[58] Li, M. (2008), "A Quasi-Analytical Interpolation Method for Pricing American Options," Working paper, Georgia Institute of Technology.

[59] Li, M. (2009), "Analytical Approximations for the Critical Stock Prices of American Options: A Performance Comparison," Working paper, Georgia Institute of Technology.

[60] Little, T., V. Pant and C. Hou (2000), "A new Integral Representation of the Early Exercise Boundary for American Put Options," *Journal of Computational Finance*, 3 (3), 73-96.

[61] Longstaff, F. and E. Schwartz (2001), "Valuing American Options by Simulation: a Simple Least Squares Approach," *Review of Financial Studies,* 14, 113-147.

[62] McDonald, R. and M. Schroder (1998), "A Parity Result for American Options," *Journal of Computational Finance*, 1, 5–13.

[63] MacMillan, L.W. (1986), "An Analytic Approximation for the American Put Price," *Advances in Futures and Options Research*, 1, 119-139.

[64] Merton, R. (1973, "Theory of Rational Option Pricing," *Bell Journal of Economics and Management Science*, 4, 141-183.

[65] Pham, H. (1997), "Optimal Stopping, Free Boundary, and American Option in a Jump-Diffusion Model," *Applied Mathematics and Optimization*, 35, 145–164.

[66] Press, W., S. Teukolsky, W. Vetterling and B. Flannery (1992), *Numerical Recipes in C*, 2nd Edition, Cambridge University Press.

[67] Staunton, M. (2020), "A Thanksgiving Meal for Americans", *Wilmott Magazine*, May, *forthcoming*.

[68] Subrahmanyam, M. and G. Yu (1993), "Pricing and Hedging American Options: A Unified Method and its Efficient Implementation," Working Paper, New York University.

[69] Tang, T., X. Xu and J. Cheng (2008), "On Spectral Methods for Volterra Integral Equations and the Convergence Analysis," *Journal of Computational Mathematics*, 26 (6), 825–837.

[70] Tavella, D. and C. Randall (2000), *Pricing Financial Instruments: The Finite Difference Method*, John Wiley & Sons.

[71] Traub, J. (1964), *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs.

[72] Trefethen, L. (2011), "Six Myths of Polynomial Interpolation and Quadrature," Working Paper, Oxford University.

[73] Trefethen, L. (2008), "Is Gauss Quadrature better than Clenshaw-Curtis?," *SIAM Review*, 50 (1), 67-87.

[74] Tsitsiklis, J. and B. Van Roy (1999), "Optimal Stopping of Markov processes: Hilbert Space Theory, Approximation Algorithms, and an Application to Pricing High-Dimensional Financial Derivatives," *IEEE Transactions on Automatic Control*, 44, 1840-1851.

[75] Wilmott, P., J. Dewynne, and S. Howison (1995), *The Mathematics of Financial Derivatives*, Cambridge University Press.

[76] Xie, D., D. Edwards, G. Scheiniger, and Q. Zhu (2011), "Characterization of the American Put Option Using Convexity," *Applied Mathematical Finance*, 18(4), 353-365.

[77] J. Zhang and T. Li (2010), Pricing and Hedging American Options Analytically: A Perturbation Method," *Mathematical Finance*, 20 (1), 59–87.

[78] Zhu, S. (2006), "A new Analytical Approximation Formula for the Optimal Exercise Boundary of American Put Options," *International Journal of Theoretical and Applied Finance*, 9(7), 1141-1177.

[79] Zhu, S. and Z. He (2007), "Calculating the Early Exercise Boundary of American Put Options with an Approximation Formula," *International Journal of Theoretical and Applied Finance*, 10(7), 1203-1227.

A QD$^+$ Approximation

In [59], the author refines an approach in [50] to arrive at the following approximate implicit equation for $B(\tau)$:

$$-e^{-q\tau}\Phi\left(-d_+\left(\tau, B(\tau)/K\right)\right) + \frac{\left(\lambda(h)+c_0\right)\left(K - B(\tau) - v(\tau, B(\tau))\right)}{B(\tau)} = -1, \tag{66}$$

where v is the European option price in (5), $h = 1 - e^{-r\tau}$, $\omega = 2(r-q)\sigma^{-2}$,

$$c_0 = -\frac{(1-h)^{\frac{2r}{\sigma^2}}}{2\lambda + \omega - 1}\left(\frac{1}{h} - \frac{e^{r\tau}\Theta\left(\tau, B(\tau)\right)}{r(K - B(\tau) - v(\tau, B(\tau))} + \frac{\lambda'}{2\lambda+\omega-1}\right),$$

and

$$\lambda = \frac{-(\omega-1) - \sqrt{(\omega-1)^2 + \frac{8r}{\sigma^2 h}}}{2}, \quad \lambda' = \frac{2r}{\sigma^2 h^2\sqrt{(\omega-1)^2 + \frac{8r}{\sigma^2 h}}}.$$

The function Θ is the theta (time derivative) of the European put price:

$$\begin{aligned}\Theta\left(\tau, B(\tau)\right) = {} & rKe^{-r\tau}\Phi\left(-d_-\left(\tau, B(\tau)/K\right)\right) \\ & - qB(\tau)e^{-q\tau}\Phi\left(-d_+\left(\tau, B(\tau)/K\right)\right) - \frac{\sigma B(\tau)}{2\sqrt{\tau}}e^{-q\tau}\phi\left(d_+\left(\tau, B(\tau)/K\right)\right).\end{aligned}$$

For a given value of τ, (66) can be solved for $B(\tau)$ by a numerical root search algorithm, e.g., Newton's or Householder's method. Empirically, we find that Halley's method (see [71]) works well for (66).