Article

Double sweep LU decomposition for American options under negative rates

Fabien Le Floc'h

* Correspondence: fabien@2ipi.com

Abstract: The classic Brennan-Schwartz algorithm to solve the linear complementary problem, which arises from the finite difference discretization of the partial differential equation related to American option pricing does not lead to the exact solution under negative interest rates. This is due to the two exercise boundaries which may appear under negative interest rate, while the algorithm was proven to lead to the exact solution in the case of a single exercise boundary only. This paper explains that two sweeps of the Brennan-Schwartz algorithm in two directions is enough to recover the exact solution.

Keywords: American options; finite difference method; quantitative finance; pricing.

1. Introduction

American options allow the holder of the contract to exercise their right to buy (for a call option) or sell (for a put option) the underlying asset S at a fixed strike price K, at any time prior to the maturity date T of the option contract. In contrast to the European option, where exercise is only possible at the maturity date, the early-exercise feature introduces a non-linearity in the valuation of American options and numerical techniques must be used.

A common technique to price American option contracts is to discretize the partial differential equation (PDE) of the chosen model, such as Black-Scholes [Black and Scholes 1973], local volatility [Dupire 1994], or stochastic volatility [Heston 1993], with the finite difference method [Ikonen and Toivanen 2007, Le Floc'h 2014, Le Floc'h 2021, O'Sullivan and O'Sullivan 2009, Wilmott et al. 1993]. Then, a linear complementary problem (LCP) must be solved at each time-step.

In the context of implicit finite difference schemes, there are many ways to solve the LCP: the Brennan-Schwartz algorithm [Brennan and Schwartz 1977], front-tracking [Pantazopoulos et al. 1998], the penalty method [Nielsen et al. 2002], operator splitting [Ikonen and Toivanen 2004], the projected SOR [Wilmott et al. 1993], and more recently, the policy iteration of Reisinger and Witte [2012]. The simplest way is to solve the system without considering the free boundary and then to apply the early exercise condition explicitly through currentPrice = max(payoff, currentPrice). While this keeps the second order accuracy on explicit schemes, it is only first order accurate in time on implicit schemes [O'Sullivan and O'Sullivan 2009]. The Brennan-Schwartz algorithm, for its performance and simplicity, is perhaps the most popular algorithm to solve the discrete LCP exactly, in the most common case of a tridiagonal system. But it suffers from known shortcomings [Jaillet et al. 1990], it does not work for slightly more exotic American contracts, typically with a non-monotonic payoff such as F(x) = |x - K|, and may also break under negative interest rates for a regular vanilla American option. The policy iteration algorithm resolves those shortcomings.

In this paper, we propose an alternative, non-iterative algorithm which still works under negative interest rates, as well as for non-monotonic payoffs. The main idea is to apply the Brennan-Schwartz algorithm in two sweeps: one downward sweep (the classic sweep for an American put option) and one upward sweep (the classic sweep for an American call option). Instead of using the original Brennan-Schwartz algorithm, we prefer the LU

decomposition formulation of Ikonen and Toivanen [2007], as the LU decomposition stage does not necessarily need to be done every single time, and may thus lead to some interesting performance improvements.

2. The LCP under the Black-Scholes model

2.1. PDE formulation

Let \mathcal{L} be the Black-Scholes-Merton operator defined by:

$$\mathcal{L}\left(f(x,t),x,t\right) = -\frac{1}{2}\sigma(x,t)^2 x^2 \frac{\partial^2 f}{\partial x^2} - \mu(x,t) x \frac{\partial f}{\partial x} + r(x,t) f(x,t), \tag{1}$$

where x is the underlying price, μ is the underlying drift, σ its volatility and r the interest rate, F(x) = f(x, T) the option payoff at maturity and f(x, t) is the option price at time t for an underlying asset spot price of x.

With this notation, the Black-Scholes-Merton equation is

$$\frac{\partial f}{\partial t}(x,t) = \mathcal{L}\left(f(x,t), x, t\right). \tag{2}$$

The early exercise feature of the option adds a free boundary on top of the Black-Scholes-Merton partial differential equation. Let f be the option price, the following system of partial differential inequalities is satisfied [Lamberton and Lapeyre 1996]:

$$\frac{\partial f}{\partial t}(x,t) \leq \mathcal{L}\left(f(x,t),x,t\right),$$

$$\left(\frac{\partial f}{\partial t}(x,t) - \mathcal{L}\left(f(x,t),x,t\right)\right) \cdot (f-F) = 0,$$

$$f \geq F,$$
(3)

where $(x, t) \in [0, X] \times [0, T]$, with boundary conditions

$$f(x,T) = F(x), (4)$$

$$\frac{\partial^2 f}{\partial x^2}(0,t) = 0, \tag{5}$$

$$\frac{\partial^2 f}{\partial x^2}(X, t) = 0. ag{6}$$

For a vanilla American call, we have $F(x) = \max(x - K, 0)$ and for a put, we have $F(x) = \max(K - x, 0)$ where K is the strike price.

2.2. TR-BDF2 Discretization

For a time discretization defined by $(t_j)_{j \in \{0,\dots,n\}}$, $k_j = t_j - t_{j-1}$ where $t_0 = 0$ is typically the valuation time and $t_n = T$ the option expiry, the discretization of the Black-Scholes-Merton PDE by the TR-BDF2 scheme reads [Le Floc'h 2014]

$$f^{\star} = f^{n} + \frac{\alpha k_{n}}{2} \left(\mathcal{L}(f^{n}) + \mathcal{L}(f^{\star}) \right), \tag{7a}$$

$$f^{n-1} = \frac{1}{2-\alpha} \left(\frac{1}{\alpha} f^* - \frac{(1-\alpha)^2}{\alpha} f^n + (1-\alpha) k_n \mathcal{L}(f^{n-1}) \right), \tag{7b}$$

with $\alpha = 2 - \sqrt{2}$ and $f^{j}(x) = f(x, t_{j})$.

A second-order central discretization in space on the full domain, and first-order for the boundary conditions 5 and 6 leads to the following two implicit stages for j = n, ..., 1

$$M^j f^* = g^j, (8a)$$

$$M^j f^{j-1} = h^j, (8b)$$

where, at the time-step j, M^j is a tridiagonal matrix with lower diagonal a_i^j for $i \in \{1,...,m\}$, upper diagonal c_i^j for $i \in \{0,...,m-1\}$ and diagonal b_i^j for $i \in \{0,...,m\}$ and

$$\begin{split} a_i^j &= \frac{\alpha k_j}{2\Delta x_{i-1} \left(\Delta x_{i-1} + \Delta x_i\right)} \left(\mu_j x_i \Delta x_i - \sigma_{i,j}^2 x_i^2\right), \\ b_i^j &= 1 + \frac{\alpha k_j}{2} \left(r_j + \frac{\mu_j (\Delta x_{i-1} - \Delta x_i) x_i + \sigma_{i,j}^2 x_i^2}{\Delta x_i \Delta x_{i-1}}\right), \\ c_i^j &= -\frac{\alpha k_j}{2\Delta x_i \left(\Delta x_{i-1} + \Delta x_i\right)} \left(\mu_j x_i \Delta x_{i-1} + \sigma_{i,j}^2 x_i^2\right), \\ g_i^j &= -a_i^j f_{i-1}^j + (2 - b_i^j) f_i^j - c_i^j f_{i+1}^j, \\ h_i^j &= \frac{1}{2 - \alpha} \left(\frac{1}{\alpha} f_i^{\star} - \frac{(1 - \alpha)^2}{\alpha} f_i^j\right), \end{split}$$

for i = 1, ..., m-1 with $f^j = (f_0^j, ..., f_m^j)^\top$, $g^j = (g_0^j, ..., g_m^j)^\top$, $h^j = (h_0^j, ..., h_m^j)^\top$, $\Delta x_i = x_{i+1} - x_i$. The boundary conditions lead to

$$\begin{split} b_0^j &= 1 + \frac{\alpha k_j}{2} \left(r_j + \frac{\mu_j x_0}{\Delta x_0} \right), \quad c_0^j = -\alpha k_j \frac{\mu_j x_0}{2\Delta x_0} \,, \\ a_m^j &= \alpha k_j \frac{\mu_j x_m}{2\Delta x_{m-1}} \,, \quad b_m^j = 1 + \frac{\alpha k_j}{2} \left(r_j - \frac{\mu_j x_m}{\Delta x_{m-1}} \right). \end{split}$$

The corresponding linear complimentary problem (3) discretization reads, for j = n, ..., 1

$$\begin{pmatrix}
M^{j} f^{*} \geq g^{j} \\
f^{*} \geq F(x)
\end{pmatrix} \qquad \text{Trapezoidal stage,} \qquad (9a)$$

$$\begin{pmatrix}
M^{j} f^{*} - g^{j}
\end{pmatrix}^{\top} (f^{*} - F(x)) = 0$$

$$M^{j} f^{j-1} \geq h^{j} \\
f^{j-1} \geq F(x)
\end{pmatrix} \qquad \text{BDF2 stage.} \qquad (9b)$$

$$\begin{pmatrix}
M^{j} f^{j-1} - h^{j}
\end{pmatrix}^{\top} (f^{j-1} - F(x)) = 0$$

The Brennan-Schwartz and the policy iteration algorithms are only valid if the matrix M^j has the following properties [Jaillet et al. 1990]:

- the lower and upper diagonals are negative: $a_{i,j} \le 0$ and $c_{i,j} \le 0$ for $i \in \{1,...,m-1\}, c_{0,j} \le 0, a_{m,j} \le 0$
- the diagonal is dominant: $a_{i,j} + b_{i,j} + c_{i,j} \ge 0$ for $i \in \{1, ..., m-1\}$, $b_{0,j} + c_{0,j} \ge 0$, $a_{m,j} + b_{m,j} \ge 0$ and $b_{i,j} > 0$ for $i \in \{0, ..., m\}$.

In other terms, M^j must be an irreducible Minkowski matrix (also known as M matrix). For our TR-BDF2 discretization, this translates to for $i \in \{1, ..., m-1\}$:

$$-\frac{\sigma_{i,j}^2 x_i}{\Delta x_{i-1}} \le \mu_j \le \frac{\sigma_{i,j}^2 x_i}{\Delta x_i},\tag{10a}$$

$$0 \le 1 + \frac{\alpha k_j}{2} r_{i,j} \,. \tag{10b}$$

And for the boundaries:

$$\mu_j x_0 \ge 0, \quad \mu_j x_m \le 0.$$
 (11)

Except for the boundaries, those conditions are almost always verified in practice. Furthermore one can always make Δx_i small enough so that 10a holds. We may also impose this condition in the general case via exponential fitting [Healy 2021, Il'in 1969]. If we choose $x_0 = 0$, which also helps in improving the overall accuracy for American options, then only the upper boundary may be problematic.

3. Double sweep LU decomposition

The LU decomposition algorithm is slightly easier to analyze on the following reformulated equivalent problem:

with

$$z = f^* - F(x), \quad v = g^j - M^j F(x)$$
 (13)

for the TR-BDF2 stage and

$$z = f^{j-1} - F(x), \quad v = h^j - M^j F(x)$$
 (14)

for the BDF2 stage.

The algorithm presented in [Ikonen and Toivanen 2007], valid for an American call payoff, will decompose M^j such that $M^j = LU$ with L lower triangular, U upper triangular and solve first Ly = v, then Uz = y. Similarly, the algorithm of an American put payoff will decompose M^j such that $M^j = \bar{U}\bar{L}$ with \bar{L} lower triangular, \bar{U} upper triangular and solve first $\bar{U}y = v$, then $\bar{L}z = y$. It is during the last step that the Brennan-Schwartz algorithm differs from the classic algorithm for a linear tridiagonal system, by enforcing the non-linear constraint.

The correctness of the original Brennan-Schwartz algorithm is proven in [Jaillet et al. 1990, Propostion 5.6] when M^j is an M-matrix using specific assumptions on the shape of the solution. For an American put option, on the transformed problem formulation, the assumption reads

$$\exists k \in \{0, ..., m\} \mid \forall i \le k, z_i = 0 \text{ and } \forall i > k, z_i > 0.$$
 (15)

Under positive interest rates, a single early-exercise boundary exists for an American call or put option, and this justifies the validity of the $\bar{U}\bar{L}$ back-solving in Algorithm 2. In particular, the original Brennan-Schwartz algorithm is not valid in the context of negative interest rates, as for a vanilla American call or put option, two

Algorithm 1: LUUL Decomposition for the transformed problem

```
// start LU decomposition.
1 l_{00} = b_0
 u_{01} = c_0/l_{00}
 3 for i ← l to m-1 do
        l_{ii-1} = a_i
        l_{ii} = b_i - l_{ii-1} u_{i-1i}
       u_{ii+1} = c_i/l_{ii}
 7 end for
 8 l_{mm-1} = a_m
 9 l_{mm} = b_m - l_{mm-1}u_{m-1m}
                                                                                                          // start \bar{U}\bar{L} decomposition.
10 \bar{u}_{mm} = b_m
11 l_{mm-1} = a_m/\bar{u}_{mm}
12 for i \leftarrow m - 1 \ 1 \ to \ 1 \ do
        \bar{u}_{i\,i+1}=c_i
13
        \bar{u}_{ii} = b_i - \bar{u}_{ii+1}\bar{l}_{i+1i}
14
15 \bar{l}_{ii-1} = a_i/\bar{u}_{ii}
16 end for
17 \bar{u}_{01} = c_0
18 \bar{u}_{00} = b_0 - \bar{u}_{01} \bar{l}_{10}
```

Algorithm 2: Brennan and Schwartz algorithm with LUUL Decomposition for the transformed problem

```
1 y_0 = v_0 / l_{00}
                                                                                                      // start LU back-solve.
 2 for i \leftarrow 1 to m do
 y_i = (v_i - l_{ii-1}y_{i-1})/l_{ii}
 4 end for
 5 z_m = y_m
  z_m = \max(z_m, 0) 
 7 for i \leftarrow m-1 to 0 do
       z_i = y_i - u_{i\,i+1}z_{i+1}
 9 z_i = \max(z_i, 0)
10 end for
                                                                                                      // start ar{U}ar{L} back-solve.
11 y_m = v_m / \bar{u}_{mm}
12 for i \leftarrow m-1 to 0 do
13 y_i = (v_i - \bar{u}_{ii+1}y_{i+1})/\bar{u}_{ii}
14 end for
15 \bar{z}_0 = y_0
16 z_0 = \max(z_0, \bar{z}_0)
17 for i \leftarrow 1 to m do
       \bar{z}_i = y_i - \bar{l}_{i\,i-1}z_{i-1}
19 z_i = \max(z_i, \bar{z}_i)
20 end for
```

early-exercise boundaries appear when respectively $r < r - \mu < 0$ or $r - \mu < r < 0$ for a constant interest rate r and drift μ [Andersen and Lake 2021].

Andersen and Lake [2021] show that the two boundaries may be solved independently. It can also easily be seen from the PDE formulation as a free-boundary problem [Healy 2021]:

$$\mathcal{L}f(x,t) = \frac{\partial f}{\partial t}(x,t) \quad \text{for } (x,t) \in \mathcal{C},$$
(16)

with initial condition

$$\lim_{t \to T} f(x, t) = F(x), \tag{17}$$

and boundary conditions

$$f(x,t) = F(x) \quad \text{for } x = u(t) \text{ or } x = l(t), \tag{18}$$

$$\frac{\partial f}{\partial x} = -1 \quad \text{for } x = u(t) \text{ or } x = l(t), \tag{19}$$

$$f(x,t) > F(x) \quad \text{in } \mathscr{C},$$
 (20)

$$f(x,t) = F(x) \quad \text{in } \mathcal{D},$$
 (21)

where l(t) and u(t) represent respectively the lower and upper early-exercise boundaries and

$$\mathcal{C} = \{(x, t) \in [0, \infty) \times [0, T] : l(t) < x < u(t)\},$$
(22)

$$\mathcal{D} = \{(x, t) \in [0, \infty) \times [0, T] : x < l(t)\} \cup \{(x, t) \in [0, \infty) \times [0, T] : x > u(t)\}. \tag{23}$$

As long as the boundaries do not yet intersect, we can split the problem in two separate domains $\{(x,t) \in [0,\infty) \times [0,T] : x \le l(t)\}$ and $\{(x,t) \in [0,\infty) \times [0,T] : x \ge u(t)\}$. When the boundaries intersect at time t^{\dagger} , early-exercise is never optimal for $t < t^{\dagger}$, and the algorithm is applicable on $(t^{\dagger},T]$.

This split motivates the two back-solving sweeps of Algorithm 2. Now let us prove the validity of the algorithm for the case of two boundaries.

Proposition 1. If the solution z of System 12 satisfies

$$\exists (k_1, k_2) \in \{0, ..., m\}^2 \mid \forall k_1 \le i \le k_2, z_i = 0 \text{ and } \forall i \in \{0, ..., m\} \setminus \{k_1, ..., k_2\}, z_i > 0,$$
(24)

and M^{j} is an M-matrix, then Algorithm 2 finds the exact solution.

Proof. The LU sweep leads to z_i , $i > k_2$ and the $\bar{U}\bar{L}$ sweep to z_i , $i < k_1$. In between we know that $z_i = 0$. Because M^j is an M-matrix, we know from Cottle and Sacher [1976], that the solution is unique. \Box

Remark 1. In Algorithm 2, in general, we can not stop the loops at k'_1 and k'_2 where k'_1 is the first index such that $y_i \le 0$ and k'_2 the first index such that $\bar{y}_i \le 0$ in the spirit of the algorithm of Elliot and Ockendon [1985, p. 115-116].

For example, stopping early breaks when the vector v is such that

$$\begin{cases} v_{i} > 0 \text{ for } i < k_{1}^{"}, \\ v_{i} < 0 \text{ for } k_{1}^{"} \le i < k_{2}^{"}, \\ v_{i} > 0 \text{ for } k_{2}^{"} \le i < k_{3}^{"}, \\ v_{i} = 0 \text{ for } k_{3}^{"} \le i, \end{cases}$$
(25)

for some k_1'', k_2'', k_3'' such that $0 < k_1'' < k_2'' < k_3'' < m$ as in the case of an American put option where $r - \mu < r < 0$.

The double sweep algorithm will also lead to a very good estimate of the solution for a butterfly American option, while an Elliot and Ockendon [1985, p. 115-116] like algorithm will not. The latter would require more transitions.

Proposition 2. For an American call option under positive interest rates, the Brennan-Schwartz algorithm is still valid when $c_0 > 0$ or $a_m > 0$, if $v_0 \le 0$.

Proof. If $c_0 > 0$, we have $l_{ii} > 0$ as long as $l_{11} = b_1 - a_1 u_{01} = b_1 - a_1 c_0 / b_0 > 0$, which is true since $a_1 < 0$ and $b_i > 0$. Thus $y_0 > 0$ and the sign of y_i is unchanged compared to the case $c_0 < 0$, for $i \ge 1$. Similarly, the sign of z_i is unchanged for $i \ge 1$. Only at i = 0 we may have an inconsistency, but for a call, it is never optimal to exercise using practical grid bounds as the early exercise payoff is essentially 0.

If $a_m > 0$, we have $l_{mm} = b_m - a_m u_{m-1m} = b_m - a_m * c_{m-1}/l_{m-1m-1}$. We know that $c_{m-1} < 0$ and thus $l_{mm} > 0$, $u_{m-1m} < 0$. The value of y_m may still be strictly negative. Let k_1 be the first index such that $y_i \le 0$. In practice, $k_1 < m$, unless it is never optimal to early-exercise. Since y_i becomes negative for $k_1 \le i < m$, the back-solving loops may be stopped at index k < m and $z_i = 0$. The value y_m is effectively not used. \square

The same reasoning is obviously applicable to the Brennan-Schwartz algorithm for the American put, as well as to Algorithm 2.

In order to improve the performance of the algorithm on trivial cases, we may check for the number of sign changes and the sign of the first element of v. If there is zero or one sign change, we may run only the UL back-solve when the sign is positive, and the LU back-solve when the sign is negative.

4. Numerical examples

4.1. Negative interest rates

We consider the example from [Andersen and Lake 2021] of an American put option of strike K = 100 and various maturities with an underlying asset spot price of S = 100, a constant interest rate of r = -1.2%, a drift $\mu = 0.4\%$ and volatility $\sigma = 10\%$.

We price each option with the TR-BDF2 scheme applied on a grid composed of n=100 steps in the time dimension and m=2000 steps in the asset dimension. In the asset dimension, we use a non-uniform hyperbolic grid, with more points close to the strike price and less points at the boundaries. In the time dimension, we consider two different discretizations, one with constant steps, and one with the time step size following a uniform square root law: $t_j = T - (n-j)^2/n^2T$. In the former time-discretization, the system matrix may be computed once, and the LU factorization may be reused across time-steps latter, while in the latter the system matrix must be updated at each time-step. The latter is representative of the more general case of non-constant rates, drift or volatility. The TR-BDF2 scheme however involves the same matrix decomposition in each of its two internal stages, and the LU factorization is still beneficial in practice.

As expected, the double sweep algorithm takes twice the time of the original Brennan-Schwartz algorithm (Table 1). The tridiagonal policy iteration solver of Reisinger and Witte [2012] is around 50% slower on this

example. We found the policy iteration solver to be even slower (around twice) for call options under negative

Table 1. Error in the price of an American put of five distinct maturities, when computed with the TR-BDF2 scheme and various solvers for the LCP. PI, LUUL, BS stand respectively for the tridiagonal policy iteration solver, the double sweep LU decomposition solver and the classic Brennan-Schwartz solver.

T	Reference Price	Time-steps	PI Error (Time)	LUUL Error (Time)	BS Error (Time)
45/365	1.380533089	Varying	-1.0e-5 (27 ms)	-1.0e-5 (20 ms)	-2.0e-3 (10 ms)
		Constant	-2.9e-5 (29 ms)	-2.9e-5 (14 ms)	-2.0e-3 (9 ms)
90/365	1.942381237	Varying	-3.1e-5 (29 ms)	-3.1e-5 (13 ms)	-7.1e-3 (9 ms)
		Constant	1.0e-7 (30 ms)	1.0e-7 (19 ms)	-3.8e-3 (10 ms)
180/365	2.729267252	Varying	-8.2e-6 (35 ms)	-8.2e-6 (19 ms)	-7.1e-3 (11 ms)
		Constant	-5.9e-5 (36 ms)	-5.9e-5 (13 ms)	-7.1e-3 (9 ms)
360/365	3.830520425	Varying	1.4e-6 (40 ms)	1.4e-6 (18 ms)	-1.2e-2 (11 ms)
		Constant	8.1e-5 (36 ms)	8.1e-5 (14 ms)	-1.2e-2 (9 ms)
3600/365	12.189323541	Varying	-3.6e-6 (29 ms)	-3.6e-6 (17 ms)	-1.4e-2 (10 ms)
		Constant	-4.5e-4 (25 ms)	-4.5e-4 (10 ms)	-1.4e-2 (9 ms)

rates.

Table 2 verifies, using a small grid, that the double sweep LU decomposition leads to exactly the same solution as the policy iteration solver.

Table 2. Price of an American option on a small grid of 20 time-steps and 20 space steps, when computed with the TR-BDF2 scheme and various solvers for the LCP. PI, LUUL, BS stand respectively for the tridiagonal policy iteration solver, the double sweep LU decomposition solver and the classic Brennan-Schwartz solver. $S = 90, K = 100, \sigma = 8\%, r = 1\%, \mu = 0.5\%$.

	PI	LUUL	Difference
Call	0.2924244529450148	0.29242445294501457	2.2 e-16
Put	10.635776477887287	10.635776477887285	1.8e-15

4.2. American Butterfly

We consider now an American butterfly option of strikes $K_1 = 90$ and $K_2 = 110$, maturity T = 0.25 using the following market data: $r = \mu = 1\%$, $\sigma = 100\%$, S = 110. We use a fixed uniform discretization in the asset price dimension composed of 301 points from $x_0 = 0$ to $x_m = 300$ and vary the number of time-step using uniform steps. In particular, x_m is less than three standard deviations away from the spot price, which allows to put in evidence the error of the double sweep algorithm. The number of points does not change the scale of the error.

Table 3 shows that the double sweep algorithm is still very accurate in practice. It does not perturb the order of convergence in contrast to the classic Brennan-Schwartz algorithm which results in a significantly larger error in price. The policy iteration algorithm is however nearly as fast as the double sweep on this example, especially when the number of time-steps is larger than 16.

5. Conclusion

We have shown that using two sweeps of the traditional Brennan-Schwartz algorithm constitute a simple and exact algorithm to solve the linear complementary problem arising in the pricing of American options under negative interest rates when the system involves a tridiagonal matrix (the most common case in practice). It is particularly relevant to price vanilla American options under non-constant interest rate, or underlying asset drift, as well as for stocks paying discrete dividends.

Table 3. The column "Difference" is the price obtained by the Successive Over Relaxation method subtracted to the price obtained by the specific solver.

n	Solver	Price	Difference	Time
4	BS	6.163251	-2.74e+00	110µs
	LUUL	8.900522	-1.52e-06	126 µs
	PI	8.900523	4.44e-14	210 µs
8	BS	7.030902	-1.83e+00	$235 \mu s$
	LUUL	8.865021	-2.81e-07	$228\mu s$
	PI	8.865021	-6.04e-14	296 µs
16	BS	7.596790	-1.27e+00	306µs
	LUUL	8.863211	-1.51e-08	$465 \mu s$
	PI	8.863211	1.03e-13	$486\mu s$
32	BS	7.972106	-8.91e-01	795 µs
	LUUL	8.862836	-1.56e-10	964 μs
	PI	8.862836	-1.78e-14	806 µs
64	BS	8.248415	-6.14e-01	924 µs
	LUUL	8.862750	-1.79e-13	$1139\mu s$
	PI	8.862750	3.55e-14	$1387\mu s$

It is faster in general than the policy iteration algorithm optimized for tridiagonal systems, while being straightforward to implement. It is however not exact anymore for non-monotonic payoffs, such as American butterfly options, but we found it to lead to very accurate results in practice nonetheless.

Andersen, Leif and Mark Lake. 2021. Fast american option pricing: The double-boundary case. *Wilmott 2021*(116), 30–41. Black, Fischer and Myron Scholes. 1973. The pricing of options and corporate liabilities. *Journal of political economy 81*(3), 637–654.

Brennan, M.J. and E.S. Schwartz. 1977. The valuation of American put options. Journal of Finance 32(2), 449-462.

Cottle, Richard W and Richard S Sacher. 1976. On the solution of large, structured linear complementarity problems: The tridiagonal case. *Applied Mathematics and Optimization 3*(4), 321–340.

Dupire, Bruno. 1994. Pricing with a smile. Risk 7(1), 18-20.

Elliot, C and JR Ockendon. 1985. Weak and variational methods for free boundary problems. Pitman, London.

 $Healy, Jherek.\ 2021.\ Applied\ Quantitative\ Finance\ for\ Equity\ Derivatives\ (3\ ed.).\ Amazon.$

Heston, S.L.. 1993. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of financial studies*, 327–343.

Ikonen, S. and J. Toivanen. 2004. Operator splitting methods for American option pricing. *Applied Mathematics Letters* 17(7), 809–814.

Ikonen, Samuli and Jari Toivanen. 2007. Pricing american options using lu decomposition. *Applied Mathematical Sciences* 1(51), 2529–2551.

Il'in, Arlen Mikhailovich. 1969. Differencing scheme for a differential equation with a small parameter affecting the highest derivative. *Mathematical Notes of the Academy of Sciences of the USSR 6*(2), 596–602.

Jaillet, P., D. Lamberton, and B. Lapeyre. 1990. Variational inequalities and the pricing of American options. *Acta Applicandae Mathematicae* 21(3), 263–289.

Lamberton, D. and B. Lapeyre. 1996. Introduction to Stochastic Calculus Applied to Finance. Chapman and Hall.

Le Floc'h, Fabien. 2014. Tr-bdf2 for fast stable american option pricing. Journal of Computational Finance 17(3), 31-56.

Le Floc'h, Fabien. 2021. Pricing american options with the runge-kutta-legendre finite difference scheme. *International Journal of Theoretical and Applied Finance* 24(3), 2150018.

Nielsen, B.F., O. Skavhaug, and A. Tveito. 2002. Penalty and front-fixing methods for the numerical solution of American option problems. *Journal of Computational Finance* 5(4), 69–98.

O'Sullivan, Stephen and Conall O'Sullivan. 2009. On the acceleration of explicit finite difference methods for option pricing. *Quantitative Finance 1469-7696.*

Pantazopoulos, KN, EN Houstis, and S. Kortesis. 1998. Front-tracking finite difference methods for the valuation of american options. *Computational Economics* 12(3), 255–273.

Reisinger, Christoph and Jan Hendrik Witte. 2012. On the use of policy iteration as an easy way of pricing american options. *SIAM Journal on Financial Mathematics* 3(1), 459–478.

Wilmott, P., J. Dewynne, and S. Howison. 1993. *Option Pricing: Mathematical Models and Computation*. Oxford Financial Press.

Appendix A Inside the butterfly example

The inaccuracy of the double sweep algorithm on the American butterfly example is already visible if we reduce the number of steps in the asset price dimension to 15, and use 3 time-steps.

The tridiagonal matrix is as follows:

```
a = (0, -0.012081845276054914, -0.0485714587865642, -0.10946884053152789,
```

- -0.19477399051094593, -0.3044869087248183, -0.4386075951731451, -0.5971360498559263,
- -1.7558948467866478, -2.0608699108761552, -2.3902527432001173, 0.0036611652351682144),
- b = (1.0002440776823445, 1.024651845916799, 1.097875150620162, 1.219913991792434,
- 1.3907683694336146, 1.6104382835437039, 1.878923734122702, 2.196224721170609,
- 4.514962703443775, 5.125156909305134, 5.784166651635402, 0.9965829124471763),
- c = (0, -0.012325922958399462, -0.0490596141512533, -0.1102010735785615,
- -0.19575030124032408, -0.30570729713654105, -0.44007206126721243, -0.5988445936323381,
- -0.7820248942319182, -0.9896129630659527, -1.2216088001344414,
- -1.4780124054373847, -1.7588237789747825, -2.064042920746634, -2.3936698307529403, 0),

with initial vector

```
g = (0,0,0,0,

1.9575030124032409,3.895617164562961,4.386075951731451,0,

0,0,0,0,

0,0,0,0),
```

and lower boundary

```
F = (0, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).
```

The nearly exact solution found by the policy iteration algorithm reads

```
f^{\star} = (0,0.00013908409255599,0.011562036583884633,0.2586020570733455,\\ 2.8512116514642054,10,5.021713994073349,1.507175220503007,\\ 0.5200832132225875,0.20066505470872006,0.0847766655914132,0.038534316489836615,\\ 0.018454045388137823,0.008902039586522732,0.0036786845579122227,0).
```

The error of the double sweep algorithm reads

```
\begin{split} f^{\star}_{LUUL} - f^{\star} &= (0, 0, 0, 0, 0, 0, 2.53 \cdot 10^{-9}, 1.08 \cdot 10^{-8}, \\ &3.71 \cdot 10^{-8}, 1.11 \cdot 10^{-7}, 2.96 \cdot 10^{-7}, 7.24 \cdot 10^{-7}, 1.64 \cdot 10^{-6}, 3.49 \cdot 10^{-6}, 7.02 \cdot 10^{-6}, 0). \end{split}
```

Appendix B Combined double-sweep Brennan-Schwartz

Here we give the more compact algorithm corresponding the double sweep Brennan-Schwartz technique, where the LU decomposition and back-solve are merged. It is not faster when using the TR-BDF2 scheme as the LU decomposition is reused among the two stages of the scheme, but may be faster for the implicit Euler (BDF1 or BDF2) schemes when the problem to solve includes time-dependent coefficients.

Algorithm 3: Fast double sweep Brennan and Schwartz algorithm for the transformed problem

```
// start fast LU back-solve.
y_0 = b_0
2 z_0 = v_0
3 for i ← 1 to m do
       y_i = b_i - a_i c_{i-1} / y_{i-1}
     z_i = v_i - a_i z_{i-1} / y_{i-1}
 6 end for
 z_m = z_m / y_m
z_m = \max(z_m, 0)
9 for i \leftarrow m-1 to 0 do
       z_i = (z_i - c_i z_{i+1}) / y_i
       z_i = \max(z_i, 0)
11
12 end for
                                                                                              // start fast ar{U}ar{L} back-solve.
13 y_m = b_m
14 \bar{z}_m = v_m
15 for i \leftarrow m-1 to 0 do
       y_i = b_i - c_i a_{i+1} / y_{i+1}
17 | \bar{z}_i = v_i - c_i \bar{z}_{i+1} / y_{i+1}
18 end for
19 \bar{z}_0 = \bar{z}_0 / y_0
20 z_0 = \max(z_0, \bar{z}_0)
21 for i \leftarrow 1 to m do
        \bar{z}_i = (\bar{z}_i - a_i z_{i-1})/y_i
       z_i = \max(z_i, \bar{z}_i)
24 end for
```