A Comparison of Numerical Methods to price One-Factor American Put Options

Victor Lu
Supervisor: D. Duffy

March 24, 2019

Contents

1 Introduction

In this project, we are going to compare different numerical methods to price one-factor American put options. The methods in the project will be coded in C++, with visualizations made in Python.

2 Finite difference methods

Finite difference methods value a derivative by solving the differential equation that the derivative satisfies. The differential equation is converted into a set of difference equations, and the difference equations are solved iteratively.
Let's consider an American put option on a stock paying a dividend yield of q. The option must satisfy Black Scholes equation[1]:

$$-\frac{\partial f}{\partial t} + (r-q)S\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf$$

Suppose that the life of the option is T. We divide the time interval into N equally spaced intervals of length $\Delta t = T/N$. There are a total of $N+1$ times considered:

$$0, \Delta t, 2\Delta t, \ldots, T$$

S is defined on the semi-infinite interval $(0, \infty)$. However, a computer can only handle bounded domains, in this paper two ways are considered to have a bounded domain: Domain Truncation and Domain Transformation.

2.1 Domain Truncation

To truncate the space domain, we consider a price S_{max} sufficiently high such that when it is reached, the put has no value. We generally consider $S_{max} = 3K$, with K the strike price.
We then define M as the the total number of space intervals, we have $M+1$ equally spaced stock prices:

$$0, \Delta S, 2\Delta S, \ldots, S_{max}$$

To interpolate the value of the option at a point in between mesh points, we will use the cubic spline interpolation method.
In the end, the time points and the stock price points form a grid of $(M+1)(N+1)$ points. The point (i,j) on the grid corresponds to time $i\Delta t$ and stock price $j\Delta S$. We will also denote the value of the option at the (i,j) point by $f_{i,j}$. For all the methods that will be given, early exercise of the American put is taken into account by having:

$$f_{i,j} = \max(f^*_{i.j}, K - j\Delta S)$$

[1]This is the version of the Black-Scholes equation where the payoff at T corresponds here to $t=0$. This is obtained with the change of variable $t = T - t^*$

where $f^*_{i,j}$ the value of the point at point (i,j) obtained by solving the difference equations.

2.1.1 Explicit Euler

Let's consider the following approximations:

$$\begin{aligned}\frac{\partial f}{\partial S} &= \frac{f_{i,j+1} - f_{i,j-1}}{2\Delta S}\\ \frac{\partial f}{\partial t} &= \frac{f_{i+1,j} - f_{i,j}}{\Delta t}\\ \frac{\partial^2 f}{\partial S^2} &= \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\Delta S^2}\end{aligned}$$

Substituting these three approximations in Black Scholes equation, we get for $j = 1, \ldots, M-1$ and $i = 0, \ldots, N-1$:

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1} = f_{i+1,j}$$

with:

$$\begin{aligned}a_j &= \left(-\frac{r-q}{2}j + \frac{\sigma^2 j^2}{2}\right)\Delta t\\ b_j &= 1 - \sigma^2 j^2 \Delta t - r\Delta t\\ c_j &= \left(\frac{r-q}{2}j + \frac{\sigma^2 j^2}{2}\right)\Delta t\end{aligned}$$

The value of the put at time $t = 0$ is $\max(K - S, 0)$, with the grid values we have:

$$f_{0,j} = \max(K - j\Delta S, 0) \qquad j = 0, 1, \ldots, M$$

The value of the put option when the stock price is zero is K:

$$f_{i,0} = K \qquad i = 0, 1, \ldots, N$$

We also made the hypothesis that the value of the put option is zero when $S = S_{max}$:

$$f_{i,M} = 0 \qquad i = 0, 1, \ldots, N$$

In the difference equation, all the terms on the left hand side are known at time $t = 0$, we can compute the value $f_{1,j}$ for all $j \in \{1, \ldots, M-1\}$.
We iterate this process until we arrive at time $t = T$, the value of the put option is then given by the cubic spline interpolation at this time step.

2.1.1.1 Stability

Let's analyze the stability of the Explicit Euler method. The previous coefficients a_j, b_j and c_j sum to $1 - r\Delta t$:

$$\begin{aligned} a_j + b_j + c_j &= 1 - r\Delta t \\ \iff \frac{1}{1 - r\Delta t}(a_j + b_j + c_j) &= 1 \end{aligned}$$

It means that we can have the following probabilistic interpretation:

$$f_{i+1,j} = (1 - r\Delta t)E(f_{i,j})$$

It means that the value at the next time grid is the discounted expected value of the previous time grid, which makes sense because here time t is backwards.
If we consider $\frac{1}{1-r\Delta t}a_j$, $\frac{1}{1-r\Delta t}b_j$ and $\frac{1}{1-r\Delta t}c_j$ as probabilities, we require that for all j:

$$\begin{aligned} j &\geq \left| \frac{r-q}{\sigma^2} \right| \\ \Delta t &\leq \frac{1}{r + \sigma^2 j^2} \end{aligned}$$

2.1.2 Implicit Euler

For the Implicit Euler, we use the following approximations for the derivatives:

$$\begin{aligned} \frac{\partial f}{\partial S} &= \frac{f_{i+1,j+1} - f_{i+1,j-1}}{2\Delta S} \\ \frac{\partial f}{\partial t} &= \frac{f_{i+1,j} - f_{i,j}}{\Delta t} \\ \frac{\partial^2 f}{\partial S^2} &= \frac{f_{i+1,j+1} - 2f_{i+1,j} + f_{i+1,j-1}}{\Delta S^2} \end{aligned}$$

The difference with the Explicit Euler is that the spatial derivatives all are taken at time step $i+1$ instead of i. Substituting in the Black Scholes equation, we get for $j = 1, \ldots, M-1$ and $i = 0, \ldots, N-1$:

$$a_j^* f_{i+1,j-1} + b_j^* f_{i+1,j} + c_j^* f_{i+1,j+1} = f_{i,j}$$

with:

$$\begin{aligned} a_j^* &= \frac{\Delta t}{1 - r\Delta t}\left(\frac{r-q}{2}j - \frac{\sigma^2 j^2}{2}\right) \\ b_j^* &= \frac{1}{1 - r\Delta t}\left(1 + \sigma^2 j^2 \Delta t\right) \\ c_j^* &= \frac{\Delta t}{1 - r\Delta t}\left(-\frac{r-q}{2}j - \frac{\sigma^2 j^2}{2}\right) \end{aligned}$$

At each time step, we use the Double Sweep Method to solve the tri-diagonal system of equation and get $f_{i+1,j}$ for $j = 1, \dots, M-1$. We iterate the process until arriving at $t = T$, the value of the put option is then given by the cubic spline interpolation.

2.1.3 Theta Methods and Crank-Nicolson

The Theta methods generalize the Implicit and Explicit Euler methods, we have the following difference equation for $\theta \in [0,1]$:

$$\begin{aligned}\frac{f_{i+1,j} - f_{i,j}}{\Delta t} &= \theta \left[\frac{(r-q)}{2} j (f_{i+1,j+1} - f_{i+1,j-1}) + \frac{1}{2}\sigma^2 j^2 (f_{i+1,j+1} + f_{i+1,j-1} - 2f_{i+1,j}) - r f_{i,j} \right] \\ &+ (1-\theta) \left[\frac{(r-q)}{2} j (f_{i,j+1} - f_{i,j-1}) + \frac{1}{2}\sigma^2 j^2 (f_{i,j+1} + f_{i,j-1} - 2f_{i,j}) - r f_{i,j} \right]\end{aligned}$$

Recombining the terms we get in the end:

$$a_j^{**} f_{i+1,j-1} + b_j^{**} f_{i+1,j} + c_j^{**} f_{i+1,j+1} = \alpha_j f_{i,j-1} + \beta_j f_{i,j} + \gamma_j f_{i,j+1}$$

with

$$\begin{aligned}
a_j^{**} &= \theta \Delta t \left(\frac{r-q}{2} j - \frac{\sigma^2 j^2}{2} \right) \\
b_j^{**} &= 1 + \theta \Delta t \sigma^2 j^2 \\
c_j^{**} &= \theta \Delta t \left(\frac{r-q}{2} j - \frac{\sigma^2 j^2}{2} \right) \\
\alpha_j &= \frac{(1-\theta)\Delta t}{2} j(-(r-q) + \sigma^2 j) \\
\beta_j &= 1 - r\theta\Delta t - (1-\theta)\Delta t (r + \sigma^2 j^2) \\
\gamma_j &= \frac{(1-\theta)\Delta t}{2} j((r-q) + \sigma^2 j)
\end{aligned}$$

Starting from $t = 0$, all the terms on the right side of the difference equation are known, and for each time step, the values at the next time step are obtained by solving a tri-diagonal system with the Double Sweep method. The value of the put is given by the cubic spline interpolation. The Crank-Nicolson method correspond to the Theta method with $\theta = \frac{1}{2}$.

2.1.4 Alternating Directions Explicit (ADE)

The Alternating Directions Explicit (ADE) method is a unconditionally stable explit finite difference scheme. It consists of two sweeps (which can be executed in parallel), the first sweep considers the solution at time level $i+1$ at indices j and $j-1$, while the second considers the solution at time level $i+1$ at indices $j+1$ and j. In this project, we will use the Saulyev version of ADE.

In Black-Scholes equation, the first sweep (left-to-right sweep) approximates the second derivative by:

$$\frac{\partial^2 f}{\partial S^2} \sim \frac{1}{(\Delta S)^2}(f_{i,j+1} - f_{i,j} - f_{i+1,j} + f_{i+1,j-1})$$

The second sweep (right-to-left sweep) approximates the second derivative by:

$$\frac{\partial^2 f}{\partial S^2} \sim \frac{1}{(\Delta S)^2}(f_{i+1,j+1} - f_{i+1,j} - f_{i,j} + f_{i,j-1})$$

Substituting these expressions in Black-Scholes equation, we get for the left-to-right sweep, denoted U:

$$\mathrm{U}_{i+1,j} = \mathrm{e}_j(a_j U_{i,j-1} + b_j U_{i,j} + c_j U_{i,j+1} + d_j U_{i+1,j-1})$$

for $j \in \{1, \ldots, M-1\}$, $i \in \{0, \ldots, N-1\}$ and where:

$$\begin{aligned}
a_j &= \frac{r-q}{2} j\Delta t \\
b_j &= r\Delta t - 1 + \frac{1}{2}\sigma^2 j^2 \Delta t \\
c_j &= -\frac{r-q}{2} j\Delta t - \frac{1}{2}\sigma^2 j^2 \Delta t \\
d_j &= -\frac{1}{2}\sigma^2 j^2 \Delta t \\
e_j &= -\frac{1}{1 + \sigma^2 j^2 \Delta t/2}
\end{aligned}$$

For the right-to-left sweep, denoted V, we get:

$$\mathrm{V}_{i+1,j} = \epsilon_j(\alpha_j V_{i,j-1} + \beta_j V_{i,j} + \gamma_j V_{i,j+1} + \delta_j V_{i+1,j+1})$$

for $j \in \{1, \ldots, M-1\}$, $i \in \{0, \ldots, N-1\}$ and where:

$$\begin{aligned}
\alpha_j &= \frac{r-q}{2} j\Delta t - \frac{1}{2}\sigma^2 j^2 \Delta t \\
\beta_j &= r\Delta t - 1 + \frac{1}{2}\sigma^2 j^2 \Delta t \\
\gamma_j &= -\frac{r-q}{2} j\Delta t \\
\delta_j &= -\frac{1}{2}\sigma^2 j^2 \Delta t \\
\epsilon_j &= -\frac{1}{1 + \sigma^2 j^2 \Delta t/2}
\end{aligned}$$

We iterate the computations until we arrive at time $t = N\Delta t$, the put value is then given by:

$$f_{N,M} = \frac{U_{N,M} + V_{N,M}}{2}$$

2.2 Domain Transformation

The domain transformation technique allow us to map the semi-infinite domain $(0, \infty)$ to the bounded interval $(0, 1)$.
Let's consider the transformation:

$$y = \frac{S}{S+1}$$

The inverse transformation is:

$$S = \frac{y}{1-y}$$

With this transformation, the partial derivatives are:

$$\begin{aligned}\frac{\partial f}{\partial S} &= \frac{\partial f}{\partial y}\frac{\partial y}{\partial S} = (1-y)^2\frac{\partial f}{\partial y} \\ \frac{\partial^2 f}{\partial S^2} &= \frac{\partial((1-y)^2\frac{\partial f}{\partial y})}{\partial S} \\ &= -2(1-y)^3\frac{\partial f}{\partial y} + (1-y)^4\frac{\partial^2 f}{\partial y^2}\end{aligned}$$

With the new variable y, Black Scholes equation can be rewritten as:

$$-\frac{\partial f}{\partial t} + (ry(1-y) - \sigma^2 y^2(1-y))\frac{\partial f}{\partial y} + \frac{1}{2}\sigma^2 y^2(1-y)^2\frac{\partial^2 f}{\partial y^2} = rf$$

This is equivalent to:

$$-\frac{\partial f}{\partial t} + y(1-y)(r - \sigma^2 y)\frac{\partial f}{\partial y} + \frac{1}{2}\sigma^2 y^2(1-y)^2\frac{\partial^2 f}{\partial y^2} = rf$$

We will discretize this equation in time and in space with the steps:

$$\begin{aligned}\Delta t &= \frac{T}{N} \\ \Delta y &= \frac{1}{M}\end{aligned}$$

2.2.1 Explicit Euler

For the transformed Black-Scholes equation, the Explicit Euler method uses the same derivatives approximations as previously and it gives us:

$$f_{i+1,j} = a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1}$$

where

$$\begin{aligned}
a_j &= \Delta t\left(-\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y)+\frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)\\
b_j &= \Delta t\left(\frac{1}{\Delta t}-r-\sigma^2 j^2(1-j\Delta y)^2\right)\\
c_j &= \Delta t\left(\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y)+\frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)
\end{aligned}$$

2.2.1.1 Stability

The same probabilistic interpretation implies that for all j:

$$\Delta t \leq \frac{1}{r+\sigma^2 j^2(1-j\Delta y^2)}$$

The condition is less stringent that the condition on the truncated domain, this could lead to an improved stability of the Euler method on the transformed domain (see Results).

2.2.2 Implicit Euler

Applying the Implicit Euler gives us the difference equations:

$$a_j f_{i+1,j-1}+b_j f_{i+1,j}+c_j f_{i+1,j+1}=f_{i,j}$$

where

$$\begin{aligned}
a_j &= \frac{1}{r-1/\Delta t}\left(-\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y)+\frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)\\
b_j &= \frac{1}{r-1/\Delta t}\left(-\frac{1}{\Delta t}-\sigma^2 j^2(1-j\Delta y)^2\right)\\
c_j &= \frac{1}{r-1/\Delta t}\left(\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y)+\frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)
\end{aligned}$$

2.2.3 Theta Methods and Crank-Nicolson

The Theta methods gives us for $\theta \in (0,1)$:

$$a_j f_{i+1,j-1}+b_j f_{i+1,j}+c_j f_{i+1,j+1}=\alpha_j f_{i,j-1}+\beta_j f_{i,j}+\gamma_j f_{i,j+1}$$

where

$$\begin{aligned}
a_j &= \theta\left(\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y)-\frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)\\
b_j &= \frac{1}{\Delta t}+\theta\sigma^2 j^2(1-j\Delta y)^2
\end{aligned}$$

$$
\begin{aligned}
c_j &= \theta\left(-\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y) - \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)\\
\alpha_j &= (1-\theta)\left(-\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y) + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)\\
\beta_j &= \frac{1}{\Delta t} - r - (1-\theta)\sigma^2 j^2(1-j\Delta y)^2\\
\gamma_j &= (1-\theta)\left(\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y) + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)
\end{aligned}
$$

2.2.4 Alternating Direction Explicit (ADE)

The sweep left-to-right U is given by:

$$U_{i+1,j} = a_j U_{i,j-1} + b_j U_{i,j} + c_j U_{i,j+1} + d_j U_{i+1,j-1}$$

where

$$
\begin{aligned}
a_j &= -\frac{1}{\frac{1}{\Delta t} + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2}\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y)\\
b_j &= -\frac{1}{\frac{1}{\Delta t} + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2}\left(-\frac{1}{\Delta t} + r + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)\\
c_j &= -\frac{1}{\frac{1}{\Delta t} + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2}\left(-\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y) - \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)\\
d_j &= -\frac{1}{\frac{1}{\Delta t} + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2}\left(-\frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)
\end{aligned}
$$

The sweep right-to-left V is given by:

$$V_{i+1,j} = a_j V_{i,j-1} + b_j V_{i,j} + c_j V_{i,j+1} + d_j V_{i+1,j+1}$$

where

$$
\begin{aligned}
a_j &= -\frac{1}{\frac{1}{\Delta t} + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2}\left(\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y) - \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)\\
b_j &= -\frac{1}{\frac{1}{\Delta t} + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2}\left(-\frac{1}{\Delta t} + r + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)\\
c_j &= -\frac{1}{\frac{1}{\Delta t} + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2}\left(-\frac{j}{2}(1-j\Delta y)(r-\sigma^2 j\Delta y)\right)\\
d_j &= -\frac{1}{\frac{1}{\Delta t} + \frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2}\left(-\frac{1}{2}\sigma^2 j^2(1-j\Delta y)^2\right)
\end{aligned}
$$

The value of the put at the grid point (i,j) is then obtained by:

$$f_{i,j} = \frac{1}{2}(U_{i,j} + V_{i,j})$$

3 Binomial Trees

The Binomial Trees can be used to value either European or American options. Because there are no analytic valuations for American options, Binomial Trees are most useful for valuing these types of options. The Binomial Tree method divides the life of the option into time intervals of size Δt. In each interval, it assumes that the price of the underlying asset S can only move up to Su or down to Sd, with $u > 1$ and $d < 1$. The probability of an up movement is denoted by p. The probability of a down movement is $1 - p$.

3.1 Pseudo-code

To implement the Binomal Tree in C++, we will use the following pseudo-code:

```
function americanPut(T, S, K, r, sigma, q, n) {
    ' T... expiration time
    ' S... stock price
    ' K... strike price
    ' q... dividend yield
    ' n... height of the binomial tree

    deltaT := T / n;
    up := exp(sigma * sqrt(deltaT));

    p0 := (up*exp(-q * deltaT) - exp(-r * deltaT)) / (up^2 - 1);
    p1 := exp(-r * deltaT) - p0;

    ' initial values at time T
    for i := 0 to n {
        p[i] := K - S * up^(2*i - n);
        if p[i] < 0 then p[i] := 0;
    }

    ' move to earlier times
    for j := n-1 down to 0 {
        for i := 0 to j {
            p[i] := p0 * p[i+1] + p1 * p[i]; ' binomial value
            exercise := K - S * up^(2*i - j); ' exercise value
            if p[i] < exercise then p[i] := exercise;
        }
    }

    return americanPut := p[0];
}
```

4 Results

For the numerical calculations, we will want to price an American put with the following characteristics:

$$K = 50, \quad S_0 = 50, \quad r = 0.10, \quad q = 0, \quad \sigma = 0.4, \quad T = 1$$

The numerical computations will be done in C++, and the results will be plotted using the Matplotlib library of Python.

4.1 Binomial Tree

The Binomial Tree method ran for $n = 4096$ time steps give us the Figure 1.

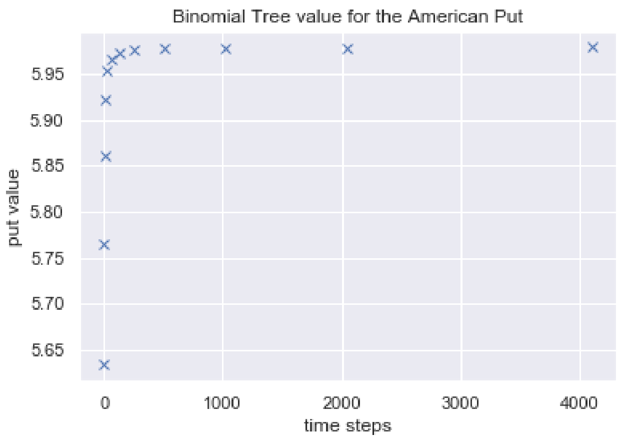


Figure 1: Binomal Tree valuation of the American put

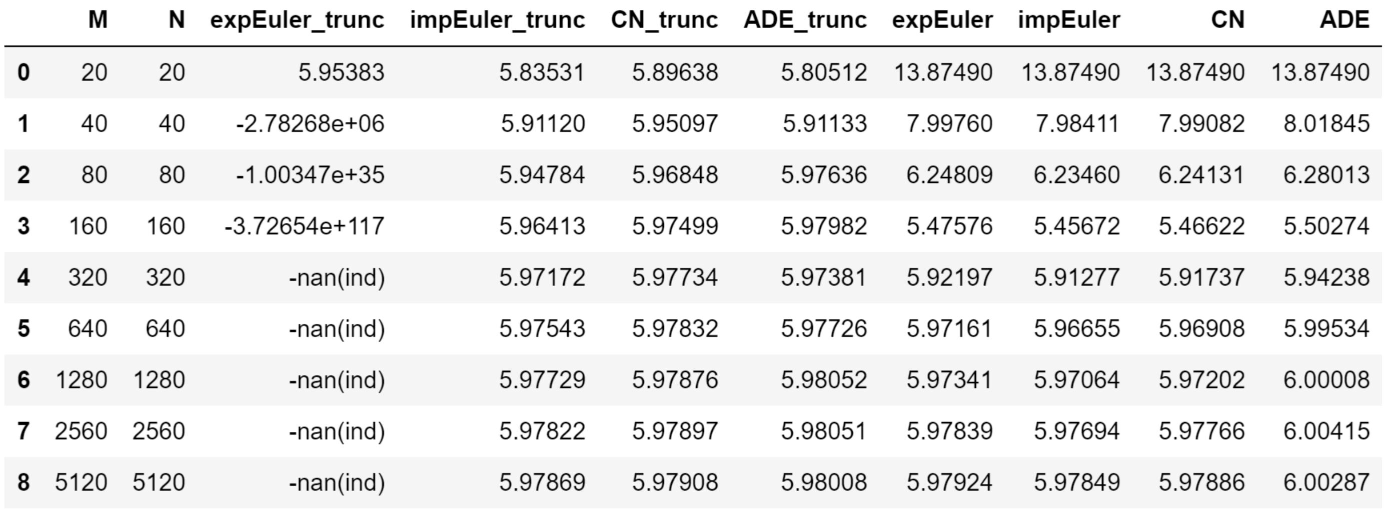
The Binomal Tree method converges to the value $f = 5.98$. We will consider this value as the true value of the American Put and set it as the baseline to compare the accuracy of other methods.

4.2 Finite difference methods

The results for the different finite difference methods considered are given in Figure 2. The values are interpolated at $S = S_0$ with the cubic spline method.

The following first remarks can be made:

- The explicit Euler is unstable on the truncated domain. As we increase the number of time and space steps, the output of the algorithm diverges.
- The transformed domain with the transformation $y = \frac{S}{S+1}$ give us quite bad results for small time and space steps. This is because the value we consider is $S_0 = 50$, which correspond to $y = \frac{50}{50+1} \simeq 0.98$. Because we use the cubic spline interpolation to interpolate, we would need a high number of time and space steps to get accurate results. A better transformation in our case would be: $y = \frac{S}{S+K}$ or $\frac{S}{S+S_0}$.



	M	N	expEuler_trunc	impEuler_trunc	CN_trunc	ADE_trunc	expEuler	impEuler	CN	ADE
0	20	20	5.95383	5.83531	5.89638	5.80512	13.87490	13.87490	13.87490	13.87490
1	40	40	-2.78268e+06	5.91120	5.95097	5.91133	7.99760	7.98411	7.99082	8.01845
2	80	80	-1.00347e+35	5.94784	5.96848	5.97636	6.24809	6.23460	6.24131	6.28013
3	160	160	-3.72654e+117	5.96413	5.97499	5.97982	5.47576	5.45672	5.46622	5.50274
4	320	320	-nan(ind)	5.97172	5.97734	5.97381	5.92197	5.91277	5.91737	5.94238
5	640	640	-nan(ind)	5.97543	5.97832	5.97726	5.97161	5.96655	5.96908	5.99534
6	1280	1280	-nan(ind)	5.97729	5.97876	5.98052	5.97341	5.97064	5.97202	6.00008
7	2560	2560	-nan(ind)	5.97822	5.97897	5.98051	5.97839	5.97694	5.97766	6.00415
8	5120	5120	-nan(ind)	5.97869	5.97908	5.98008	5.97924	5.97849	5.97886	6.00287

Figure 2: Values for different number of time and space discretizations given by the Explicit Euler, Implicit Euler, Crank-Nicolson, ADE on both truncated and transformed domain

4.2.1 Convergence

Let's consider the convergence of the Implicit Euler, Crank-Nicolson and ADE methods on the Truncated domain. We get the Figure 3. Among the three methods, the fastest converging one is the Crank-Nicolson method, followed by the Implicit Euler. Our ADE method shows some instability at the beginning and overshoots the baseline value. Our implementation is one version (Saulyev version) of the ADE, other methods which may be more stable can also be used.

We obtain the curves in Figure 4 on the Transformed domain. Here again, we see the biasedness of our ADE method. For the Explicit Euler, Implicit Euler and Crank-Nicolson however, we see that the curves are nearly confounded. The new domain made the Explicit Euler stable and made the Explicit Euler, Implicit Euler and Crank-Nicolson have nearly the same convergence speed.

4.2.2 Initial values

The values of the put for a given range of spot prices for $M = 100$ and $N = 1000$ are given in Figure 5. We see that the prices of the Implicit Euler, Crank-Nicolson and ADE roughly all follow the values given by the Binomial Tree methods.

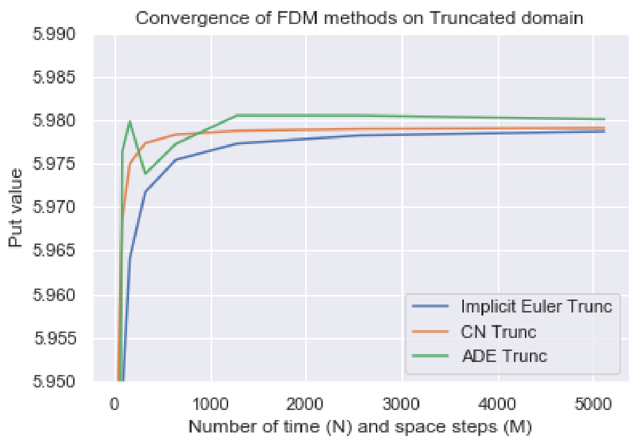


Figure 3: Convergence of the Implicit Euler, Crank-Nicolson and ADE methods on the Truncated domain

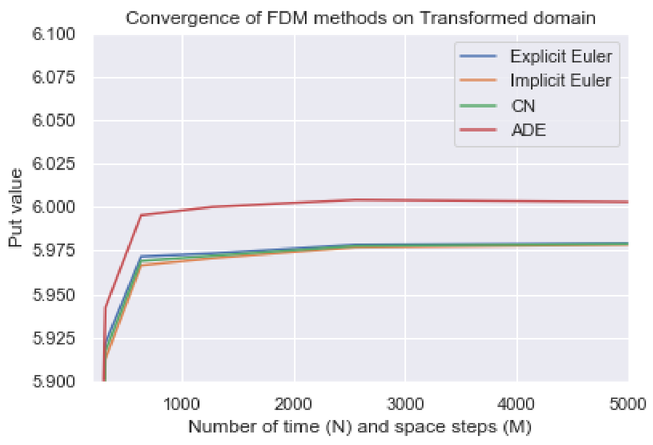


Figure 4: Convergence of the Implicit Euler, Crank-Nicolson and ADE methods on the Truncated domain

We tried the following different time steps: $M = 500, N = 100$, $M = N = 100$ for both the Truncated and Transformed domain. In each case the curves are similar to the one found in Figure 5. The methods seem to work on the whole range of spot prices.

5 Conclusion

In this project, we implementend the Explicit Euler, Implicit Euler, Crank-Nicolson and Alternating Direction Explicit (ADE) in C++ for both the truncated and transformed domain to price an American Put option. We compared the price given by the finite difference methods with the price given by the Binomial Tree method. This allowed us to verify the theoretical properties of the methods studied during the course. These are: the unstability of the Explicit Euler, the stability of the Implicit Euler and the best speed convergence of the Crank-Nicolson method. We also found that using the transformed domain instead of the truncated domain made the Explicit Euler, Implicit Euler and Crank-Nicolson methods share the same

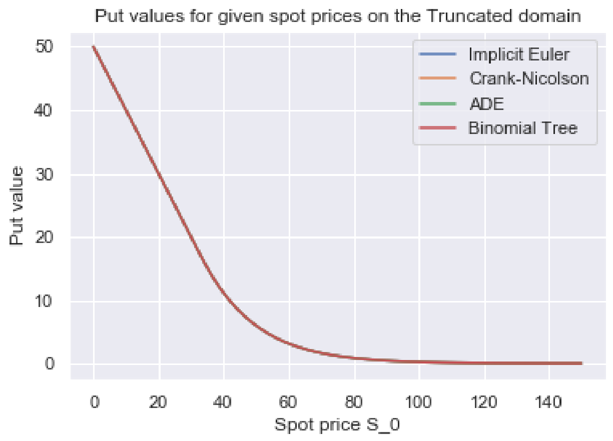


Figure 5: Put values for given spot prices on the Truncated domain

speed of convergence. Moreover, the finite difference methods considered show numerical results that are consistent with the Binomial Tree on the whole range of spot prices. The results in this paper are given only for given characteristics of the option, for further exploration of the stability and convergence of the finite difference methods considered in this paper, one could stress test the characteristics of the option and reiterate the steps in this paper.

References

[1] Daniel J. Duffy (2018) *Financial Instrument Pricing Using C++ 2e*, Wiley.

[2] Daniel J. Duffy (2006) *Finite Difference Methods in Financial Engineering*, Wiley.

[3] John C. Hull (2018) *Options, Futures and Other Derivatives*, Pearson.

[4] L.J. Campbell, B. Yin (2006) *On the Stability of Alternating-Direction Explicit Methods for Advection-Diffusion Equations*, Wiley InterScience.