Dauphine | PSL
UNIVERSITÉ PARIS

Pricing American Put Options

Project Monte-Carlo

Iacob Jessica, Mourre Grégoire

Teacher : Prof. Claisse Julien

MIDO

January 2024

Contents

1 Introduction

All the code, as well as the report, is available on the GitHub site
`github.com/gregoiremrr/Monte-Carlo-for-American-Options`.

American options form a family of derivatives well known in finance. They give to the owner the right to exercise at any time before the maturity T, in opposition to, for instance, European options where the owner can only exercise at maturity. The general form of the price at time 0 of such option is the following:

$$\sup_{\tau \in \mathcal{T}} \mathbb{E}[e^{-r\tau} g(X_\tau)],$$

where $\mathcal{T}$ is the set of $[0, T]$-valued stopping times with $t_0 = 0$ and $t_N = T$, r is the risk free rate (which is supposed to be constant) and g is the function of payoff. To simplify notations, let us define:

$$Y_t = e^{-rt} g(X_t), \ \forall t \in [0, T].$$

To solve the above maximisation problem, the natural idea is to discretized in time the problem: we replace $\mathcal{T}$ by $\bar{\mathcal{T}}$ the $\{t_0, \cdots, t_N\}$-valued stopping times, with $0 = t_0 < t_1 < \cdots < t_N = T$. This leads to a discretized optimal stochastic control problem. Those problems are well-known and the Snell envelope $(V_i)_{i \in \{0, \cdots, N\}}$ gives us a representation of the value function of the problem:

$$\begin{aligned} V_N(X_{t_N}) &= Y_{t_N} \\ V_k(X_{t_k}) &= \max\Big(Y_{t_k}, \mathbb{E}[V_{k+1}(X_{t_{k+1}})|X_{t_k}]\Big). \end{aligned}$$

Thus, the dynamic principle gives us that:

$$V_0(X_{t_0}) = \sup_{\tau \in \bar{\mathcal{T}}} \mathbb{E}[Y_\tau] = \sup_{\tau \in \mathcal{T}} \mathbb{E}[e^{-r\tau} g(X_\tau)],$$

which is a good approximation of the solution to our initial problem.

We will apply our results on American Puts for the numerical simulations. The payoff of an American Put is $g : x \mapsto (K - x)_+ := \max(0, K - x)$ where K is the strike of the option. We have seen on the course how to apply the Tsitsiklis & Van Roy and Longstaff & Schwartz to compute a lower bond for the price. In this project we will explain the Duality method which gives us a good upper bound.

2 Duality

In this section, we present a duality result that will be helpful for finding an upper bound on the price of an american option. The duality result comes from [Rogers, 2002]. In his paper, he presented the result in continuous time. Here, we will present [Glasserman, 2004]'s approach, so we will study the problem in discrete time. We want to approximate

$$V_0(X_{t_0}) = \sup_{\tau \in \bar{\mathcal{T}}} \mathbb{E}[Y_\tau].$$

For each martingale $(M_i)_{i \in \{0, \cdots, N\}}$ starting from 0 at time 0, we have $\mathbb{E}[M_i] = 0$ for all $i \in \{0, \cdots, N\}$. So for all $\tau \in \bar{\mathcal{T}}$,

$$\mathbb{E}[Y_\tau] = \mathbb{E}[Y_\tau - M_\tau] \leq \mathbb{E}[\max_{k \in \{0, \cdots, N\}} \Big(Y_{t_k} - M_k\Big)].$$

Taking the supremum on all $\tau \in \bar{\mathcal{T}}$ on the right side and the infimum on all $(M_i)_{i\in\{0,\cdots,N\}}$ martingales starting from 0, we get

$$\sup_{\tau\in\bar{\mathcal{T}}} \mathbb{E}[Y_\tau] \leq \inf_{M\in\mathcal{M}_0} \mathbb{E}[\max_{k\in\{0,\cdots,N\}} \Big(Y_{t_k} - M_k\Big)],$$

where $\mathcal{M}_0$ is the set of martingale starting from 0. In fact, we can show the reverse inequality. Let us define the martingale $M_0 = 0$ and $M_k = \Delta_1 + \cdots + \Delta_k$ for all $k \in \{1, \cdots, N\}$, with

$$\Delta_k = V_k(X_{t_k}) - \mathbb{E}[V_k(X_{t_k})|X_{t_{k-1}}].$$

Such a process is indeed a martingale with respect to its natural filtration: for all $k \in \{1, \cdots, N\}$,

$$\mathbb{E}[M_k - M_{k-1}|\mathcal{F}_{t_{k-1}}] = \mathbb{E}[\Delta_k|\mathcal{F}_{t_{k-1}}] = \mathbb{E}[V_k(X_{t_k})|\mathcal{F}_{t_{k-1}}] - \mathbb{E}[V_k(X_{t_k})|X_{t_{k-1}}] = 0,$$

as the process is also a Markov chain. By definition, the martingale starts from 0. We now need to check that

$$V_0(X_{t_0}) = \mathbb{E}[\max_{k\in\{0,\cdots,N\}} \Big(Y_{t_k} - M_k\Big)].$$

In fact we will prove more: for all $k \in \{0, \cdots, N\}$,

$$V_k(X_{t_k}) = \max\Big\{Y_{t_k},\ Y_{t_{k+1}} - \Delta_{k+1},\ \cdots,\ Y_{t_N} - \Delta_N - \cdots - \Delta_{k+1}\Big\}.$$

For $k = N$, the above formula is obvious as

$$V_N(X_{t_N}) = Y_{t_N}.$$

By backward induction, let us suppose that the formula is verified for $k+1 \in \{1, \cdots, N\}$ fixed. Then,

$$\begin{aligned}
V_k(X_{t_k}) &= \max\Big(Y_{t_k}, \mathbb{E}[V_{k+1}(X_{t_{k+1}})|X_{t_k}]\Big) \\
&= \max\Big(Y_{t_k}, V_{k+1}(X_{t_{k+1}}) - \Delta_{k+1}\Big) \\
&= \max\Big(Y_{t_k}, \max\Big\{Y_{t_{k+1}},\ \cdots,\ Y_{t_N} - \Delta_N - \cdots - \Delta_{k+2}\Big\} - \Delta_{k+1}\Big) \\
&= \max\Big(Y_{t_k}, Y_{t_{k+1}} - \Delta_{k+1},\ \cdots,\ Y_{t_N} - \Delta_N - \cdots - \Delta_{k+1}\Big).
\end{aligned}$$

Thus, the formula is satisfied for all $k \in \{1, \cdots, N\}$ and with $k = 0$, we get:

$$V_0(X_{t_0}) = \max_{k\in\{0,\cdots,N\}} \Big\{Y_{t_k} - M_k\Big\}$$

So instead of our initial maximisation problem over stopping times, we now have a minimisation problem over $\mathcal{M}_0$. Moreover, we know what the optimal martingal M^* for this new minimisation problem looks like. By approximating M^*, we will find a good upper bound on $V_0(X_{t_0})$. In fact, we are just studying a special case of the Doob-Meyer decomposition for super-martingale.

3 Approximation of Continuation Value Functions

In this section, we will present several tools that will be usefull in the next sections (4 and 5). The main goal of the presented tools will be to approximate the quantities:

$$C_k : x \mapsto \mathbb{E}[V_{k+1}(X_{t_{k+1}})|X_{t_k} = x], \quad \forall k \in \{0, \cdots, N-1\},$$

which are called the continuation functions. First, let us notice that by definition of the conditionnal expectation in $\mathbb{L}^2$,

$$\begin{aligned}\mathbb{E}[V_{k+1}(X_{t_{k+1}})|X_{t_k}] &= \underset{Z\in\mathbb{L}^2(\Omega,\sigma(X_{t_k}),\mathbb{P})}{\arg\min}\ \mathbb{E}\Big[\Big(V_{k+1}(X_{t_{k+1}}) - Z\Big)^2\Big] \\ &= \underset{\Phi(X_{t_k})\in\mathbb{L}^2}{\arg\min}\ \mathbb{E}\Big[\Big(V_{k+1}(X_{t_{k+1}}) - \Phi(X_{t_k})\Big)^2\Big].\end{aligned}$$

3.1 Non-parametric regression

A first idea is to approximate this new (arg-)minimisation problem with a Monte-Carlo method:

- at time t_{N-1},

$$\mathbb{E}[V_{k+1}(X_{t_{k+1}})|X_{t_k}] \approx \underset{\Phi(X_{t_k})\in\mathbb{L}^2}{\arg\min}\ \frac{1}{n}\sum_{j=1}^{n}\Big(Y^j_{t_{k+1}} - \Phi(X^j_{t_k})\Big)^2,$$

- at time t_k with $k \in \{0,\cdots,N-1\}$,

$$\mathbb{E}[V_{k+1}(X_{t_{k+1}})|X_{t_k}] \approx \underset{\Phi(X_{t_k})\in\mathbb{L}^2}{\arg\min}\ \frac{1}{n}\sum_{j=1}^{n}\Big(V_{k+1}(X^j_{t_{k+1}}) - \Phi(X^j_{t_k})\Big)^2,$$

  with $(X^j_{t_k})_{k\in\{0,\cdots,N\}}$ following the law of $(X_{t_k})_{k\in\{0,\cdots,N\}}$ for each $j \in \{1,\cdots,n\}$.

Choosing the right Φ is an infinite dimensional problem. In order to simplify it, we can look for a "good" Φ in a parametrised class of functions:

$$\begin{aligned}\mathcal{A} &= \Big\{x \mapsto \sum_{i=1}^{l}\alpha_i\phi_i(x),\ \text{with } \alpha_1,\cdots,\alpha_l \in \mathbb{R}\Big\} \\ &= \text{Span}(\phi_1,\cdots,\phi_l)\end{aligned}$$

with $l \in \mathbb{N}^*$ and $\phi_1,\cdots,\phi_l$ functions (called base functions) choosen beforehand. Those base functions should be taken so they can easily approximate the real continuation functions. With this in mind, we will try in the following sections the start of several basis of polynoms and we will sometimes add the payoff function (with some power). Here are the base functions that we will try:
With the canonical basis of $P[X]$:

- $\mathcal{A}_{1,1} = \text{Span}(1, X, X^2, X^3)$
- $\mathcal{A}_{1,2} = \text{Span}(1, X, X^2, X^3, g)$
- $\mathcal{A}_{1,3} = \text{Span}(1, X, X^2, X^3, X^4, g)$
- $\mathcal{A}_{1,4} = \text{Span}(1, X, X^2, X^3, g, g^2)$
- $\mathcal{A}_{1,5} = \text{Span}(1, X, X^2, X^3, g, g^2, g^3)$

With Legendre's basis of $P[X]$:

- $\mathcal{A}_{2,1} = \text{Span}(1, X, \frac{1}{2}(3X^2-1), \frac{1}{2}(5X^3-3X))$
- $\mathcal{A}_{2,2} = \text{Span}(1, X, \frac{1}{2}(3X^2-1), \frac{1}{2}(5X^3-3X), g)$
- $\mathcal{A}_{2,3} = \text{Span}(1, X, \frac{1}{2}(3X^2-1), \frac{1}{2}(5X^3-3X), \frac{1}{8}(35X^4-30X^2+3), g)$

- $\mathcal{A}_{2,4} = \text{Span}(1, X, \frac{1}{2}(3X^2-1), \frac{1}{2}(5X^3-3X), g, g^2)$
- $\mathcal{A}_{2,5} = \text{Span}(1, X, \frac{1}{2}(3X^2-1), \frac{1}{2}(5X^3-3X), g, g^2, g^3)$

With Hermite's basis of $P[X]$:

- $\mathcal{A}_{3,1} = \text{Span}(1, X, X^2-1, X^3-3X)$
- $\mathcal{A}_{3,2} = \text{Span}(1, X, X^2-1, X^3-3X, g)$
- $\mathcal{A}_{3,3} = \text{Span}(1, X, X^2-1, X^3-3X, X^4-6X^2+3, g)$
- $\mathcal{A}_{3,4} = \text{Span}(1, X, X^2-1, X^3-3X, g, g^2)$
- $\mathcal{A}_{3,5} = \text{Span}(1, X, X^2-1, X^3-3X, g, g^2, g^3)$

With Laguerre's basis of $P[X]$:

- $\mathcal{A}_{4,1} = \text{Span}(1, 1-X, \frac{1}{2}(X^2-4X+2), \frac{1}{6}(-X^3+9X^2-18X+6))$
- $\mathcal{A}_{4,2} = \text{Span}(1, 1-X, \frac{1}{2}(X^2-4X+2), \frac{1}{6}(-X^3+9X^2-18X+6), g)$
- $\mathcal{A}_{4,3} = \text{Span}(1, 1-X, \frac{1}{2}(X^2-4X+2), \frac{1}{6}(-X^3+9X^2-18X+6), \frac{1}{24}(X^4-16X^3+72X^2-96X+24), g)$
- $\mathcal{A}_{4,4} = \text{Span}(1, 1-X, \frac{1}{2}(X^2-4X+2), \frac{1}{6}(-X^3+9X^2-18X+6), g, g^2)$
- $\mathcal{A}_{4,5} = \text{Span}(1, 1-X, \frac{1}{2}(X^2-4X+2), \frac{1}{6}(-X^3+9X^2-18X+6), g, g^2, g^3)$

To conclude this section, we will finally use in the algorithm presented in sections 4 and 5 the approximate value function:

$$\tilde{C}_k : x \mapsto \sum_{i=1}^{l} \alpha_i \phi_i(x),$$

for all $k \in \{0, \cdots, N-1\}$ in the case of the regression.

3.2 Neural networks

An other idea to approximate Φ is to train a neural network. For the numerical simulations part, we will train a fully connected neural network with 4 hidden layers which have respective width 3, 5, 5 and 3, as the functions we want to approximate are note to complicated.

4 Martingales from Approximate Value Functions

As seen before, the objective of the duality method is to transform the American option pricing problem into a problem of martingale maximization, i.e

$$V_0(X_{t_0}) = \inf_{M \in \mathcal{M}_0} \mathbb{E}[\max_{k \in \{0,\cdots,N\}} (Y_{t_k} - M_k)]$$

We know that the infimum is attained with the martingale $(M_k = \Delta_1 + \cdots + \Delta_k)_{k \in \{0,\cdots,N\}}$ where $\Delta_k = V_k(X_{t_k}) - \mathbb{E}[V_k(X_{t_k})|X_{t_{k-1}}]$ for all $k \in \{1, \cdots, N\}$. As in section 3, we denote by $C_{k-1} : x \mapsto \mathbb{E}[V_k(X_{t_k})|X_{t_{k-1}} = x]$ the continuation value. Thus we can rewrite Δ_k as $V_k(X_{t_k}) - C_{k-1}(X_{t_{k-1}})$.

So, the Snell envelope can be approximated by:

$$\tilde{V}_k(x) = \max(e^{-rt_k} g(x), \tilde{C}_k(x))$$

where $\tilde{C}_k$ is an approximation of the continuation value C_k for $k = 1, \cdots, N-1$, as saw in section 3.

An issue with the approximation is that $\tilde{\Delta}_k = \tilde{V}_k(X_{t_k}) - \tilde{C}_{k-1}(X_{t_{k-1}})$ is no longer necessarily a martingale increment: in general we do not have the equality $\mathbb{E}[\tilde{\Delta}_k | X_{t_{k-1}}] = 0$. Thus, $\tilde{M}_k = \tilde{\Delta}_1 + \cdots + \tilde{\Delta}_k$ has no reason to be a martingale and we will not get a valid upper bound.

The Martingales from Approximate Value Functions method suggests to extract martingale differences from an approximation of value function, so we should estimate directly

$$\tilde{\Delta}_k = \tilde{V}_k(X_{t_k}) - \mathbb{E}[\tilde{V}_k(X_{t_k}) | X_{t_{k-1}}],$$

with

$$\tilde{V}_k(X_{t_k}) = \max(Y_{t_k}, \tilde{C}_k(X_{t_k})).$$

The implementation of this method is as follows:

1. Simulate a path $X_{t_0}, \cdots, X_{t_N}$ for the underlying Markov chain
2. At each step X_{t_k}, $k = 1, \cdots, N-1$ of the Markov chain:
   - Compute $\tilde{V}_k(X_{t_k}) = \max(Y_{t_k}, \tilde{C}_k(X_{t_k}))$ (the function $\tilde{C}_k$ has been evaluated beforehand, for instance during the estimation of the lower bound in Longstaff & Schwartz's algorithm)
   - Generate m successors $X_{t_k}^1, \cdots, X_{t_k}^m$ of $X_{t_{k-1}}$
   - Compute $\frac{1}{n}\sum_{j=1}^m \tilde{V}_k(X_{t_k}^j)$ to approximate $\mathbb{E}[\tilde{V}_k(X_{t_k}) | X_{t_{k-1}}]$
   - Set $\tilde{\Delta}_k = \tilde{V}_k(X_{t_k}) - \frac{1}{n}\sum_{j=1}^m \tilde{V}_k(X_{t_k}^j)$
3. Do (2) again for step X_{t_N} but with $\tilde{V}_N(X_{t_N}) = Y_{t_N}$ and $\tilde{V}_N(X_{t_N}^j) = Y_{t_N}^j$.
4. Sum to obtain the martingale $(\tilde{M}_k = \tilde{\Delta}_1 + \cdots + \tilde{\Delta}_k)_{k \in \{0, \cdots, N\}}$
5. Evaluate $\max_{k \in \{0, \cdots, N\}}(Y_{t_k} - \tilde{M}_k)$

If we could evaluate perfectly the optimal martingale, as prooven in section 2, we would need to repeat this only once. But, as $\tilde{M}$ is only an approximation of the optimal martingale, we have to repeat the algorithm several times and take the mean on all the results of (5).

5 Martingales from Stopping Rules

In comparison with previous method, we will make our reasoning in terms of stopping times as in Longstaff & Schwartz's algorithm.
Here, instead of approximate Δ_k by $\tilde{\Delta}_k = \tilde{V}_k(X_{t_k}) - \frac{1}{n}\sum_{j=1}^m \tilde{V}_k(X_{t_k}^j)$, we will use that the Δ_k in the optimal martingale re-writes as:

$$\begin{aligned}\Delta_k &= V_k(X_{t_k}) - \mathbb{E}[V_k(X_{t_k}) | X_{t_{k-1}}] \\ &= \mathbb{E}[Y_{\tau_k} | X_{t_k}] - \mathbb{E}[Y_{\tau_k} | X_{t_{k-1}}]\end{aligned}$$

with the optimal stopping policy in discrete time given by the dynamical programming principle:

$$\begin{aligned}\tau_i &= \min\Big\{t_k \in \{t_i, \cdots, t_N\} \text{ such that } Y_{t_k} \geq \mathbb{E}[V_{k+1}(X_{t_{k+1}}) | X_{t_k}]\Big\} \\ &= \min\Big\{t_k \in \{t_i, \cdots, t_N\} \text{ such that } Y_{t_k} \geq C_k(X_{t_k})\Big\}.\end{aligned}$$

As in the previous method (or in the Tsitsiklis & Van Roy's algorithm or the Longstaff & Schwartz's algorithm, see [Claisse, 2021]), we will use an approximate continuation value function $\tilde{C}_k$ (see section 3) and therefore an approximation of the optimal stopping rule:

$$\tilde{\tau}_i = \min\Big\{t_k \in \{t_i, \cdots, t_N\} \text{ such that } Y_{t_k} \geq \tilde{C}_k(X_{t_k})\Big\},$$

and approximate Δ_k:

$$\tilde{\Delta}_k = \mathbb{E}[Y_{\tilde{\tau}_k}|X_{t_k}] - \mathbb{E}[Y_{\tilde{\tau}_k}|X_{t_{k-1}}].$$

A first remark is that the process defined by $(\tilde{M}_k = \tilde{\Delta}_1 + \cdots + \tilde{\Delta}_k)_{k\in\{0,\cdots,N\}}$ is indeed a martingale, as $\mathbb{E}[\tilde{\Delta}_k|X_{t_{k-1}}] = \mathbb{E}[\mathbb{E}[Y_{\tilde{\tau}_k}|X_{t_k}]|X_{t_{k-1}}] - \mathbb{E}[Y_{\tilde{\tau}_k}|X_{t_{k-1}}] = 0$. Thus, the approximate optimal martingale will produce a valid upper bound. A second remark is that the first term in the definition of $\tilde{\Delta}_k$ can be re-written as:

$$\mathbb{E}[Y_{\tilde{\tau}_k}|X_{t_k}] = \begin{cases} Y_{t_k} & \text{if } Y_{t_k} \geq \tilde{C}_k(X_{t_k}) \\ \mathbb{E}[Y_{\tilde{\tau}_{k+1}}|X_{t_k}] & \text{otherwise} \end{cases}$$

and thus, we only have to estimate $\mathbb{E}[Y_{\tilde{\tau}_{k+1}}|X_{t_k}]$ for $k \in \{0, \cdots, N-1\}$. As in Longstaff & Schwartz's algorithm (see [Claisse, 2021]), we do not actually need to define a function for each stopping time. In fact, $Y_{\tilde{\tau}_k}$ can be re-written as:

$$Y_{\tilde{\tau}_k} := \begin{cases} Y_{t_k} & \text{if } Y_{t_k} \geq \tilde{C}_k(X_{t_k}) \\ Y_{\tilde{\tau}_{k+1}}. & \end{cases}$$

The implementation of this method is as follows:

1. Simulate a path $X_{t_0}, \cdots, X_{t_N}$ for the underlying Markov chain
2. At each step X_{t_k}, for $k = 0, 1, \cdots, N-1$:
   - Simulate m subpaths $(X^1_{t_{k+j}})_{j\in\{1,\cdots,m-k\}}, \cdots, (X^m_{t_{k+j}})_{j\in\{1,\cdots,m-k\}}$ starting from X_{t_k}
   - Evaluate $Y^j_{\tilde{\tau}_{k+1}}$ for $j = 1, \cdots, m$, and use $\frac{1}{m}\sum_{j=1}^m Y^j_{\tilde{\tau}_{k+1}}$ as an estimator of $\mathbb{E}[Y_{\tilde{\tau}_{k+1}}|X_{t_k}]$
   - Evaluate the quantity

     $$W_k := \begin{cases} Y_{t_k} & \text{if } Y_{t_k} \geq \tilde{C}_k(X_{t_k}) \\ \frac{1}{m}\sum_{j=1}^m Y^j_{\tilde{\tau}_{k+1}} & \text{otherwise} \end{cases}$$

     and use it as an estimator of $\mathbb{E}[Y_{\tilde{\tau}_k}|X_{t_k}]$
   - Set $\tilde{\Delta}_k := W_k - \frac{1}{m}\sum_{j=1}^m Y^j_{\tilde{\tau}_k}$ (with the $(Y^j_{\tilde{\tau}_k})_{j\in\{1,\cdots,m\}}$ simulated from $X_{t_{k-1}}$ in the previous step)
3. Sum to obtain the martingale $(\tilde{M}_k = \tilde{\Delta}_1 + \cdots + \tilde{\Delta}_k)_{k\in\{0,\cdots,N\}}$
4. Evaluate $\max_{k\in\{0,\cdots,N\}}(Y_{t_k} - \tilde{M}_k)$

As in the section 4, if we could evaluate perfectly the optimal martingale, we would need to repeat this only once. But, as $\tilde{M}$ is only an approximation of the optimal martingale, we have to repeat the algorithm several times and take the mean on all the results of (4).

6 Numerical simulations of the Monte-Carlo methods

6.1 With non-parametric regression

In this section, we are going to study the influence of all the parameters of the algorithms from sections 4 and 5 on the results. The code of this part can be found in the file `main.ipynb`. For the result to be reproductible, we have set the seed of the generator to 0. First, let us recap all the parameters (see Table 1).

Algorithms	Approximate Value Function	Stopping Rule
Base functions $\phi_1, \cdots, \phi_l$	Function `phi`	Function `phi`
Number of time $t_0, \cdots, t_N$	`m`	`m`
Number of paths simulated	`n`	`n`
Number of subpaths simulated	`n2`	`n_subpaths`

Table 1: Table of all the parameters in the algorithms and their name in the code.

First, we will study the influence of the base functions, with the other parameters fixed: `n = 10000, m = 12, n2 = 1000, n_subpaths = 500`. We will estimate the price of an American put option with the parameters of the model: $K = 100, r = 0.06, T = 0.5, \sigma = 0.4$ and $X_{t_0} = 80$. The reel price of the option is ≈ 21.6059 (see [Rogers, 2002], Table 4.1).

-	Approximate Value Functions			Stopping Rule		
-	Upper bound	STD	Time	Upper bound	STD	Time
$\mathcal{A}_{1,1}$	21.82199	0.00841	3.4 s	21.73372	0.00631	29.6 s
$\mathcal{A}_{1,2}$	21.72501	0.00738	4.0 s	21.74073	0.00609	28.9 s
$\mathcal{A}_{1,3}$	21.68976	0.00619	5.0 s	21.72331	0.00607	32.8 s
$\mathcal{A}_{1,4}$	21.71974	0.00748	4.3 s	21.72331	0.00598	30.5 s
$\mathcal{A}_{1,5}$	21.67081	0.00549	5.5 s	21.71392	0.00579	35.2 s
$\mathcal{A}_{2,1}$	21.82199	0.00841	3.8 s	21.73372	0.00631	28.4 s
$\mathcal{A}_{2,2}$	21.72501	0.00738	4.1 s	21.74072	0.00609	29.6 s
$\mathcal{A}_{2,3}$	21.68976	0.00619	5.4 s	21.72331	0.00607	35.1 s
$\mathcal{A}_{2,4}$	21.71974	0.00748	4.7 s	21.72330	0.00598	31.6 s
$\mathcal{A}_{2,5}$	21.67041	0.00549	5.9 s	21.71384	0.00579	35.2 s
$\mathcal{A}_{3,1}$	21.82199	0.00841	3.6 s	21.73373	0.00631	26.2 s
$\mathcal{A}_{3,2}$	21.72501	0.00738	4.1 s	21.74073	0.00609	29.6 s
$\mathcal{A}_{3,3}$	21.68976	0.00619	5.2 s	21.72331	0.00607	33.7 s
$\mathcal{A}_{3,4}$	21.71974	0.00748	4.4 s	21.72330	0.00598	31.8 s
$\mathcal{A}_{3,5}$	21.67094	0.00550	5.8 s	21.71392	0.00579	34.6 s
$\mathcal{A}_{4,1}$	21.82199	0.00841	4.1 s	21.73372	0.00631	29.4 s
$\mathcal{A}_{4,2}$	21.72501	0.00738	4.5 s	21.74074	0.00609	33.4 s
$\mathcal{A}_{4,3}$	21.68976	0.00619	6.9 s	21.72331	0.00607	42.6 s
$\mathcal{A}_{4,4}$	21.71974	0.00748	4.9 s	21.72330	0.00598	35.7 s
$\mathcal{A}_{4,5}$	21.67126	0.00550	6.0 s	21.71401	0.00579	40.2 s

Figure 1: Table of all the upper bounds and standard deviations with respect to the family of base functions.

From Figure 1, we can see that the stopping rule algorithm is not impacted too much by the familly of base fonctions used. The upper bound stays approximately 0.1 above the reel

value. On the other hand, the approximate value function algorithm gets closer and closer when we add functions to the base $\mathcal{A}$. It is approximatively 0.02 above the real value with the $\mathcal{A}_{i,5}$-families. From the time of execution of each algorithm, we can observe that the Stopping Rule algorithm has a much higher complexity. This is due to the fact that at each time step i, we need to simulate sample paths at times $i+1, \cdots, \texttt{m}$ (with `m` the index of theh final time). On the other hand, in the Martingale from Approximate Value Functions algorithm, at each time, we only need to simulates sample plaths of the next time.

Now, let us fix the family $\mathcal{A}_{1,5}$ and the parameters: `n = 10000, n2 = 1000, n_subpaths = 500`. We will try several values for `m` (see Figure 2).

-	Approximate Value Functions			Stopping Rule		
-	Upper bound	STD	Time	Upper bound	STD	Time
$\texttt{m} = 3$	21.41145	0.00452	1.2 s	21.44011	0.00605	1.8 s
$\texttt{m} = 5$	21.53178	0.00479	2.1 s	21.58186	0.00594	4.2 s
$\texttt{m} = 10$	21.62896	0.00492	4.5 s	21.68813	0.00587	20.6 s
$\texttt{m} = 11$	21.66594	0.00556	5.1 s	21.69998	0.00586	27.1 s
$\texttt{m} = 12$	21.67081	0.00549	5.6 s	21.71392	0.00579	43.1 s
$\texttt{m} = 15$	21.68842	0.00571	6.9 s	21.75519	0.00596	106.8 s
$\texttt{m} = 20$	21.71342	0.00596	10.1 s	21.78757	0.00591	216.6 s

Figure 2: Table of all the upper bounds and standard deviations with respect to `m`.

We can remark that when `m` is to small (≤ 10 for instance), the price is underestimated. $\texttt{m} \in \{11, \cdots, 14\}$ seems to be a good choice. The fact that the estimated price increases with `m` was expected because giving more freedom to the option's holder (more time where the option can used) increases the price. Once again, we can observe that the time complexity grows linearly in `m` for the Approximate Value Functions algorithm, and exponentially in `m` for the Stopping Rule algorithm, as expected.

Let us now study the impact of the number of subpaths in each algorithm. We will always use smaller value for `n_subpaths` than for `n2` because in the stopping rule algorithm, we need to simulate the whole subpath starting from X_{t_k} and this is more costly than only simulating the next step (as in the approximate value function algorithm). We fix the family of base functions $\mathcal{A}_{1,5}$ and the parameters: `n = 10000, m = 12`.

-	Approximate Value Functions			Stopping Rule		
`n2`/ `n_subpaths`	Upper bound	STD	Time	Upper bound	STD	Time
200 / 100	21.88965	0.00914	1.2 s	22.18451	0.01307	6.2 s
400 / 200	21.75476	0.00709	2.1 s	21.89817	0.00924	10.4 s
600 / 300	21.70843	0.00626	3.2	21.80254	0.00751	18.7 s
800 / 400	21.68687	0.00580	4.2	21.74008	0.00653	25.2 s
1000 / 500	21.67081	0.00549	5.3 s	21.71392	0.00579	37.4 s
1500 / 600	21.65134	0.00512	8.3 s	21.66960	0.00489	59.2 s

Figure 3: Table of all the upper bounds and standard deviations with respect to `n2` and `n_subpaths`.

We can see (Figure 3) that those parameters definitly have the bigger impact on the upper bound. In practical terms, increasing `n2` and `n_subpaths` means computing a martingale which

is closer to the optimal one. As in the previous Figures (1 and 2), the time complexity increases linearly for the first algorithm and exponentially for the second one.

To conclude this section, let us change the parameter `n` to observe its impact. Let us fix the family of base functions $\mathcal{A}_{1,5}$ and the parameters: `n2 = 1000, n_subpaths = 500, m = 12`.

-	Approximate Value Functions			Stopping Rule		
-	Upper bound	STD	Time	Upper bound	STD	Time
`n` = 1000	21.99514	0.02982	0.5 s	21.78501	0.02209	2.7 s
`n` = 3000	21.70631	0.01143	1.7 s	21.75219	0.01089	7.3 s
`n` = 6000	21.68539	0.00763	3.3 s	21.73110	0.00769	15.8 s
`n` = 10000	21.67081	0.00549	5.6 s	21.71392	0.00579	36.7 s
`n` = 15000	21.64745	0.00390	9.3 s	21.71359	0.00481	82.4 s
`n` = 20000	21.64141	0.00348	14.2 s	21.70614	0.00411	180.1 s

Figure 4: Table of all the upper bounds and standard deviations with respect to `n`.

This parameter is definitly the one that have the bigger impact on the standard deviation (see Figure 4). This is due to the fact that `n` is the parameter of the Monte-Carlo method (we take the mean of an `n`-sample).

6.2 With neural network approximation

As presented in the section 3, there are several ways to approximate the continuation value functions. We had the idea to replace the non-parametric regression by a small neural network (small as the function we want to approximate are quite simple). The code can be found in the file `main_nn_sklearn.ipynb`. The structure choosen is a neural network with 4 hidden layers with widths 3, 5, 5 and 3. It is probably not an optimal structure but we still find back the result we had with a non-parametric regression, with an upper bound of 21.75629 and a standard deviation of 0.00922, with the algorithm "Martingale from Approximate Value Functions" (section 4), and an upper bound of 21.77267 and a standard deviation of 0.00637 with the algorithm "Martingale from stopping rule" (section 5). Those results could be greatly improved by taking a more optimal sructure of neural network.

7 Finite difference method

7.1 Presentation of the method

We have to come back to a more general theory to explain the finite difference method. One more time, we focus on pricing an American put option in a finite time $[0, T]$.

We proceed like in [Lamberton and Lapeyre, 2008]. Assume that $\mathbb{Q}$ is an equivalent martingale measure (EMM), $(W_t)_{t\in[0,T]}$ is a $\mathbb{Q}$-Brownian Motion and $S_t = S_0 e^{(r-\frac{\sigma^2}{2})t+\sigma W_t}$ solves the Black-Scholes SDE which takes the following form under the probability $\mathbb{Q}$

$$\begin{cases} dS_t = S_t(rdt + \sigma dW_t) \\ S_0 = x \end{cases} \tag{1}$$

Denote X_t the log of the solution, i.e. $X_t = log(S_0)+(r-\frac{\sigma^2}{2})t+\sigma W_t$ and $dX_t = \left(r-\frac{\sigma^2}{2}\right)dt+\sigma dW_t$. We will apply the finite difference method to this process.

Like for the previous methods, we approximate the continuous time case by a discrete time problem: the number of exercise dates is finite ($t \in [0,T]$). The American payout is vanilla, i.e. the payoff is the same function Φ for all times. Here, as we consider X_t and not S_t, the price of the American option at time t is function of (t, X_t) and we use $\Phi(x) = (K - e^x)_+$ instead of g introduced in previous sections. So denote $u(t, X_t)$ the price of the option at time t and $u(T, x) = \Phi(x)$.

The principle of this kind of American option is, because we are in finite number of exercise dates (no exercise between time t_i and t_{i+1}), at each time, the holder chooses to exercise the option and get $\Phi(X_{t_i})$ or to hold it and he owns $u(t_i^+, X_{t_i})$ (which is the price of the option on (t_i, t_{i+1})). So we can rewrite $u(t_i, X_{t_i}) = \max(u(t_i^+, X_{t_i}), \Phi(X_{t_i}))$. We can also consider an optimal strategy given by a stopping time $\bar{\tau} = \inf(s \in \{t_1, \cdots, t_N\} \cap [0,T], u(s, X_s) = \Phi(X_s))$ which is the holder should not exercise at times $s < \bar{\tau}$. In mathematical terms, we have the variational inequality

$$\max\left(\partial_t u(t,x) + \frac{\sigma^2}{2}\partial^2_{xx}u(t,x) + (r - \frac{\sigma^2}{2})\partial_x u(t,x) - ru(t,x), \Phi(x) - u(t,x)\right) = 0 \tag{2}$$

$$\iff \max\left(\partial_t u(t,x) + \mathcal{A}u(t,x), \Phi(x) - u(t,x)\right) = 0$$

where $\mathcal{A}_X f = \frac{\sigma^2}{2}\partial^2_{xx} f + (r - \frac{\sigma^2}{2})\partial_x f$ is the infinitesimal generator of the process X and we consider $\mathcal{A}f = \frac{\sigma^2}{2}\partial^2_{xx} f + (r - \frac{\sigma^2}{2})\partial_x f - rf$ (remark: it is not time dependent).

We want to discretize directly the variational inequality in order to implement it. We localize the inequality to the space $\mathcal{O}_z = (-z, z)$ and add additional restrictions like Neumann condition $\partial_x v(t, \pm z) = 0$.

We use the same notations for the discretization as in [Claisse, 2021]: we introduce a mesh $\{(t_n, x_i) := (nh, -z + i\delta);\ 0 \le n \le m,\ 0 \le i \le l+1\}$ where $h = \frac{T}{m}$ and $\delta = \frac{2z}{l+1}$, and $(u_i^n)_{0\le n\le m,\ 0\le i\le l+1}$ is an approximation of $(u(t,x))_{t\in[0,T], x\in(-z,z)}$. We set $u^m = \Phi_\delta = (\Phi(x_i))_{1\le i\le l}$ (and denote Φ_i the approximation of $\Phi(x_i)$) and recursively for $n = m-1, \cdots, 0$, given u^{n+1}, we want to find u^n satisfying for all $1 \le i \le l$:

$$\max\left(\frac{u_i^{n+1} - u_i^n}{h} + \frac{\sigma^2}{2}\frac{u_{i+1}^n + u_{i-1}^n - 2u_i^n}{\delta^2} + (r - \frac{\sigma^2}{2})\frac{u_{i+1}^n - u_{i-1}^n}{2\delta} - ru_i^n, \Phi_i - u_i^n\right) = 0. \tag{3}$$

Considering the term on the left, we have :

$$\frac{u^{n+1} - u^n}{h} + A_\delta u^n = 0 \iff u^{n+1} - (I - hA_\delta)u^n = 0$$

where $A_\delta \in \mathbb{R}^{l\times l}$ is a tridiagonal matrix with coefficients $a_{i,i-1} = \frac{\sigma^2}{2\delta^2} - \frac{1}{2\delta}\left(r - \frac{\sigma^2}{2}\right)$, $a_{i,i} = -\left(\frac{\sigma^2}{\delta^2} + r\right)$, $a_{i,i+1} = \frac{\sigma^2}{2\delta^2} + \frac{1}{2\delta}\left(r - \frac{\sigma^2}{2}\right)$.
Because for $h > 0$, $\max(x, y) = 0 \iff \max(hx, y) = 0$, we can rewrite (3) as

$$\max\left(u^{n+1} - (I - h\mathcal{A}_\delta)u^n, \Phi_\delta - u^n\right) = 0 \tag{4}$$

Let $0 \le k \le \frac{\delta^2}{\sigma^2 + \delta^2 r}$ and denote $P_\delta = I + kA_\delta \in \mathbb{R}^{l\times l}$, we obtain :

$$\max\left(u^{n+1} + \frac{h}{k}P_\delta u^n - (1 + \frac{h}{k})u^n, \Phi_\delta - u^n\right) = 0$$

$$\iff \max\left(\frac{k}{k+h}\left(u^{n+1} + \frac{h}{k}P_\delta u^n\right) - u^n, \Phi_\delta - u^n\right) = 0$$

$$\iff \max\Big(\frac{k}{k+h}\Big(u^{n+1}+\frac{h}{k}P_\delta u^n\Big),\Phi_\delta\Big)=u^n$$

Denote $G(v)=\max\Big(\frac{k}{k+h}\Big(u^{n+1}+\frac{h}{k}P_\delta v\Big),\Phi_\delta\Big)$, $v\in\mathbb{R}^l$, we have (4) is equivalent to $u^n=G(u^n)$. Moreover,

$$|G(v)-G(\tilde{v})|_\infty \leq \frac{h}{h+k}|P_\delta(v-\tilde{v}))|_\infty$$

and

$$|P_\delta(v-\tilde{v}))|_\infty \leq \|P_\delta\|_\infty.|v-\tilde{v}|_\infty \leq |v-\tilde{v}|_\infty,$$

where $|\cdot|_\infty$ denotes the infinity norm on $\mathbb{R}^l$.
The third inequality holds if $\|P_\delta\|_\infty=\max_{i=1,\cdots,l}\sum_{j=1}^l|p_{ij}|\leq 1$. The elements of matrix P_δ are $p_{i,i-1}=k(\frac{\sigma^2}{\delta^2}-\frac{1}{2\delta}(r-\frac{\sigma^2}{2}))$, $p_{i,i}=1-k(\frac{\sigma^2}{\delta^2}+r)$, $p_{i,i+1}=k(\frac{\sigma^2}{\delta^2}+\frac{1}{2\delta}(r-\frac{\sigma^2}{2}))$. So the condition above is satisfied as $p_{i,j}\geq 0\ \forall(i,j)\in\{0,\cdots,l\}^2$ and $\sum_{j=1}^l p_{i,j}\leq 1\ \forall i\in\{0,\cdots,l\}$ for $\delta|r-\frac{\sigma^2}{2}|\leq\sigma^2$ $\Big(\iff \delta|b|\leq\sigma^2$ under [Claisse, 2021] notations$\Big)$. Thus, G is a contraction and admits an unique fixed-point u^*, and every sequence $(u^n)_{n\in\mathbb{N}}$ defined by $u^n=G(u^n)$ converges to u. So there is uniqueness and existence to this problem. Moreover, as we are in the case of an implicit scheme with condition $\delta|r-\frac{\sigma^2}{2}|\leq\sigma^2$ satisfied, the scheme is unconditionnaly converging and verifies the conditions of consistency and stability.

More generally, we can discretize (2) by a θ-scheme where $\theta\in[0,1]$. And instead of (4), we obtain

$$\max\Big((I-h\theta A_\delta)u^n-(I+h(1-\theta)A_\delta)u^{n+1},\Phi_\delta-u^n\Big)=0$$

For $\theta=0$, we have an explicit scheme. For $\theta=\frac{1}{2}$ we have the Crank-Nicholson scheme. Finally, for $\theta=1$, we find back the implicit scheme as in (4). To have properties of monotony, stability and consistency, we have to add the condition $\sigma^2(1-\theta)h\leq\delta^2$. Moreover, if $\theta<1$, the convergence holds if $\lim\limits_{h\to0,k\to0}\frac{h}{\delta^2}=0$ by [Claisse, 2021].

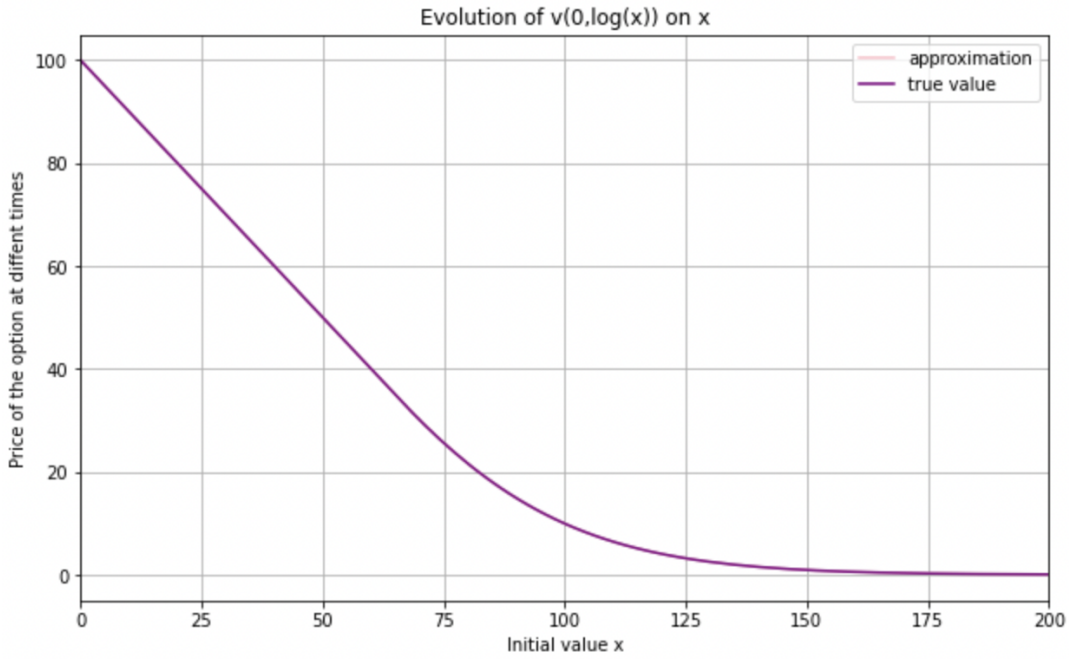
The implementation of this method is as follow:

1. Define discretization parametres.
2. Define matrix A_δ with the coefficient given above, the terminal time $u^m=\Phi_\delta$ and Neumann conditions $u_0^n=u_1^n$ and $u_l^n=u_{l+1}^n$ for each time $n\in\{0,\cdots,m\}$.
3. For each time, knowing u^{n+1}, solve $(I-h\theta A_\delta)u^{n+\frac{1}{2}}-(I+h(1-\theta)A_\delta)u^{n+1}=0$.
4. The solution we expect is given by $u^n:=\max\Big(u^{n+\frac{1}{2}},\Phi_\delta\Big)$ at each time (with $u^{n+\frac{1}{2}}$ found at previous step).

7.2 Numerical simulations of the finite difference method

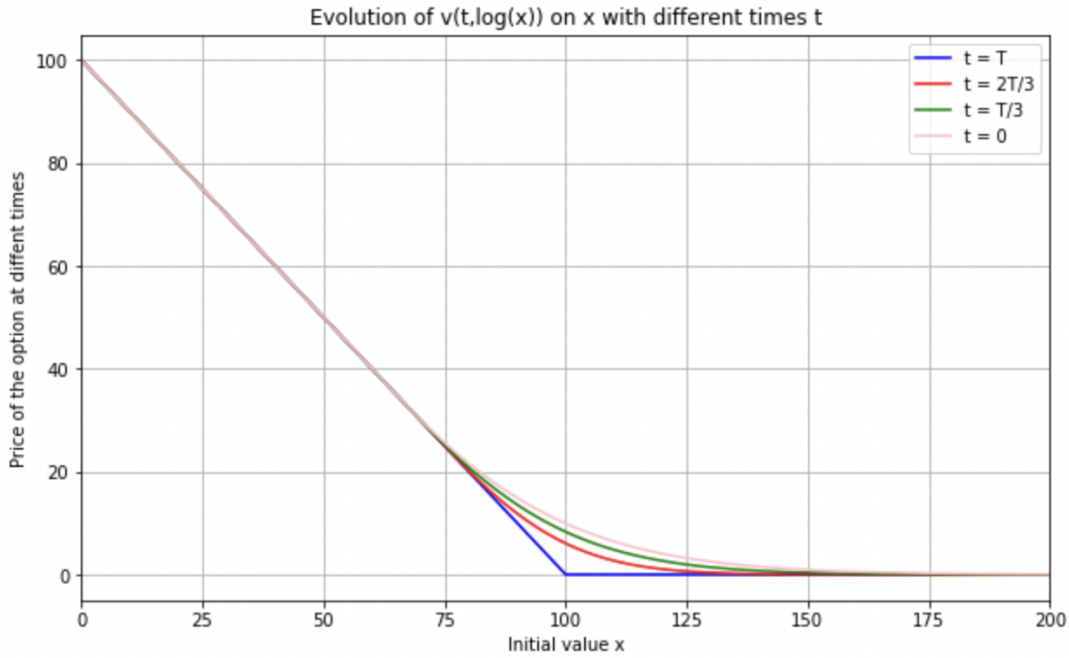
In this section, we will see the implementation of the finite difference method on an American put with parameters of the model: $K=100, r=0.06, T=0.5, \sigma=0.4$ and $X_{t_0}=80$ (as in the previous section 6). The code can be found at the end of the file `main.ipynb`.

For all θ-schemes mentioned (implicit scheme, explicit scheme and Crank-Nicholson scheme), we obtain the following value function:



This method seems to be a good approximation of the value function of this American put. Remark that we find back the real price (≈ 21.6059, see [Rogers, 2002], Table 4.1).

Watching the evolution of the solution given by this method (for the 3 scheme) at several times, we can observe how the finite difference method converge to the solution when the parameters are well set i.e $\delta|r - \frac{\sigma^2}{2}| \leq \sigma^2$ and $\sigma^2(1-\theta)h \leq \delta^2$:



Now, we will try to change the parameters to see how this method is impacted. First, we modify the space parameter l. We fix the time parameter $m = 1000$ and try $l = 15$, $l = 100$ and $l = 1000$ (see Figure 5). In the first case, the finite difference method is not a good approximation. Indeed the space step $\delta \approx 1.25$ is too large, so space discretization is not precize enough. When $l = 100$, the approximation is close to the solution but not enough smooth and $\delta \approx 0.2$. So we have to decrease the space step. Trying $l = 1000$, we obtain $\delta \approx 0.02$ and the approximation is very close to the solution. For larger space parameter, the implementation time is too long.

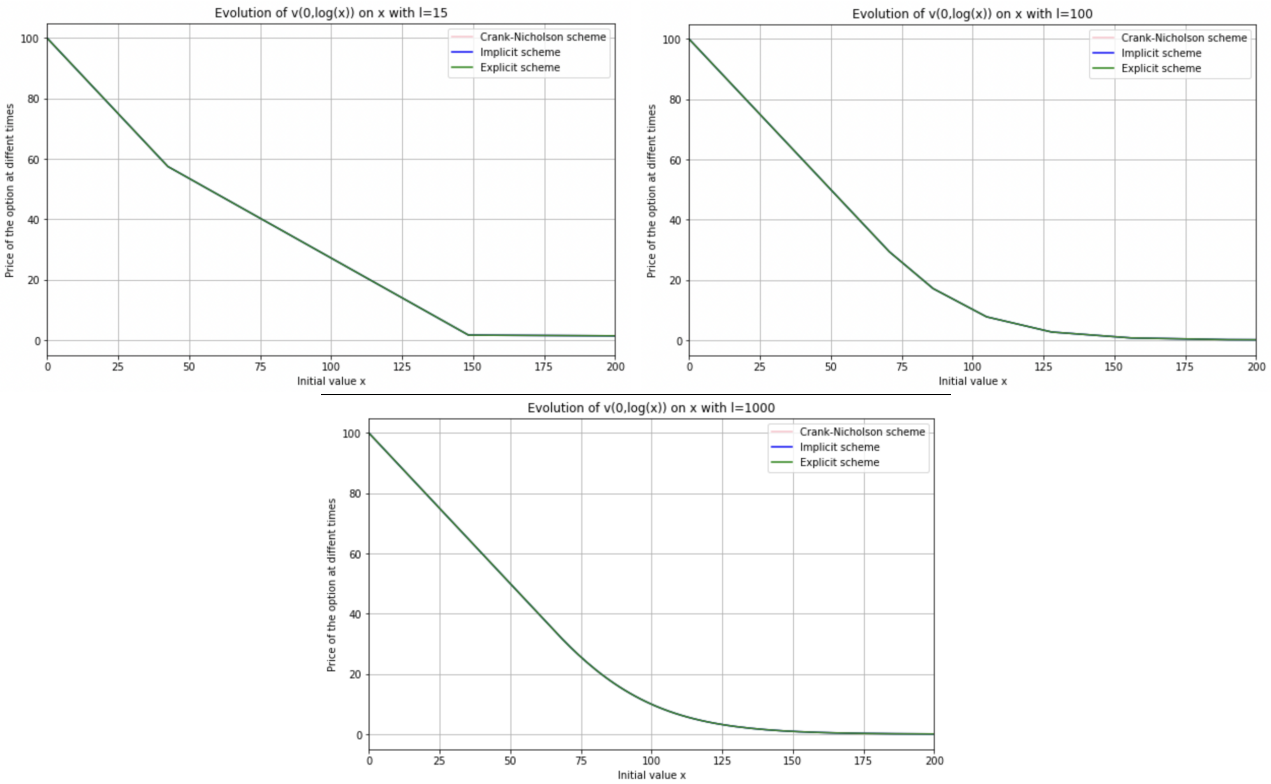


Figure 5: Evolution of $v(0, \log(x))$ for different time steps

Then, we modify the time parameter m (see Figure 6). Now, we fix the space parameter $l = 1000$, as seen in previous case, and pick different time parameters $m = 1$, $m = 10$, $m = 1000$. According [Lamberton and Lapeyre, 2008] (or Glowinsky et al. (1996) before), the objectif is to get $\frac{h}{\delta^2}$ as small as possible to get the convergence of the scheme (if $\theta \leq 1$ because if $\theta = 1$ the scheme is unconditionnaly convergent). Taking $m = 1$, we observe the method does not converge to the solution. Indeed, time discretization is not small enough as $\frac{h}{\delta^2} \approx 1252$ which is quite too large. For $m = 10$, we have $\frac{h}{\delta^2} \approx 125.2$. It seems to be too large but the solution by approximation looks close to the solution. And for $m = 1000$, the approximation is smoother and $\frac{h}{\delta^2} \approx 1.25$ which explains it.

So we fix parameters $m = 1000$ and $l = 1000$ to have the best approximation with reasonable implementation time.

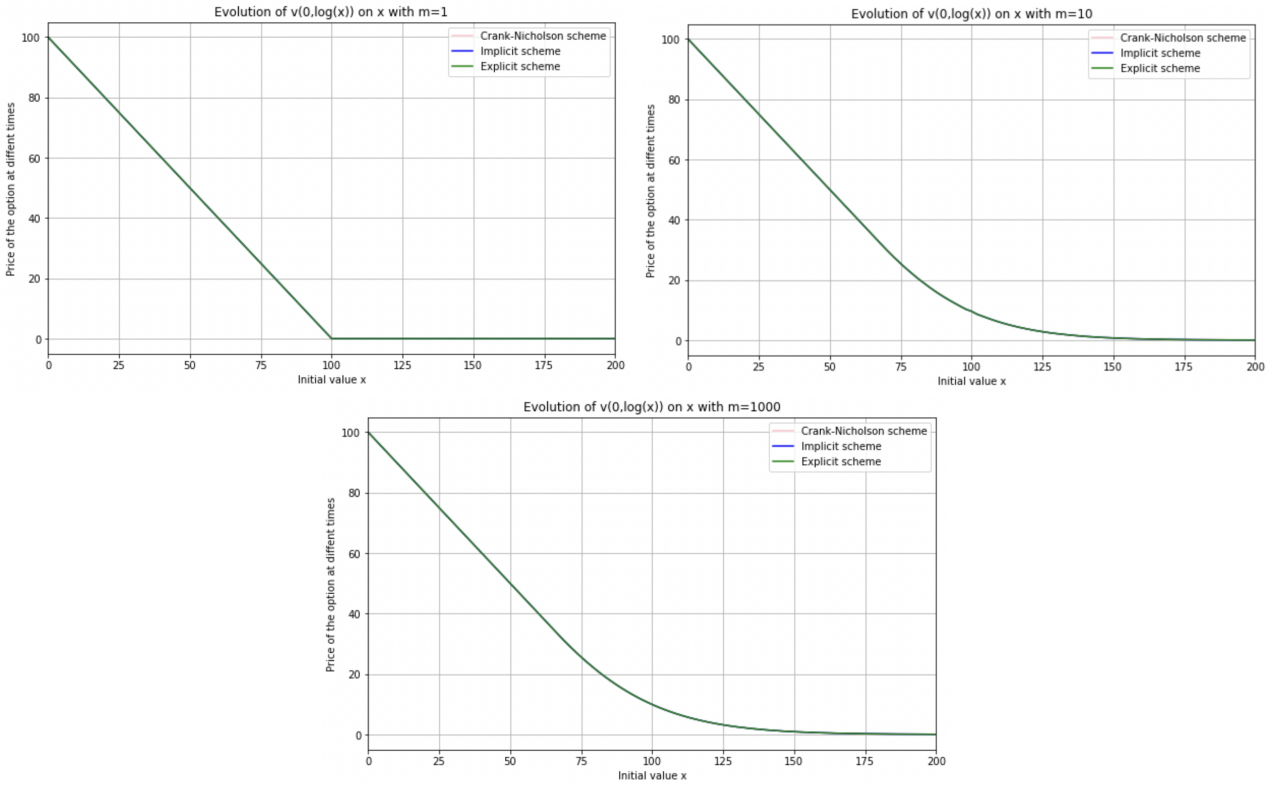


Figure 6: Evolution of $v(0, \log(x))$ for different space steps

8 Conclusion

In the evaluation of various computational techniques applied to option pricing, we conclude with notable observations and recommendations. The Martingale by Approximate Value Function method offers linear complexity with respect to most of the parameters. Thus, it is a viable method for high estimations of price. However, this method is highly sensitive to most of the parameters (the base function family, `m`, `n` or `n2`).

On the other hand, Stopping Rule algorithm offers exponential complexity, due to the fact that at each time i, we need to sample several sub paths on $i+1, \cdots, m$. However, this method is less sensitive to the base function family and the parameter `n`. So we could focus more on the other parameters `m` and `n_subpaths`.

The finte difference schemes stands out with its time complexity to solve the partial differential equation for pricing across the entire space of initial value of the asset. Unlike the other methods, the price of the option is only a single value and not a confidence interval.

We could improve some aspects such as the use of the Brennan-Schwartz algorithm instead of directly inversing the matrix in the finite difference methods. In the approximation of the continuation value functions by neural networks, we could choose a better structure by performing a cross validation, to get better approximation. Finally, we could have experimented different boundary conditions, e.g. Dirichlet conditions.

Our study does not include variance reduction methods which could decrease greatly the standard deviation for a acceptable time. The pricing of multi-asset put options, which, despite not being a requirement, would have been interesting. We also could have study tree-based pricing techniques as presented in [Lamberton and Lapeyre, 2008].

In conclusion, while each method exhibits particular strengths, they are invariably coupled with their limitations. Depending on the context, each of the technics have their advantages, taking into consideration the trade-off between computational efficiency and sensitivity to underlying parameters.

References

[Claisse, 2021] Claisse, J. (2021). *Monte Carlo and Finite Difference Methods with Applications in Finance*. Lecture notes for M2 M.A.TH./MASEF.

[Glasserman, 2004] Glasserman, P. (2004). *Monte Carlo methods in financial engineering*. Springer, New York.

[Lamberton and Lapeyre, 2008] Lamberton, D. and Lapeyre, B. (2008). *Introduction to stochastic calculus applied in finance*. Chapman & Hall/CRC Financial Mathematics Series. Chapman & Hall/CRC, Boca Raton, FL, second edition.

[Rogers, 2002] Rogers, L. (2002). Monte carlo valuing of american options. *Mathematical Finance*, 12:271–286.