

# Pricing American Put Options Using Spectral Collocation Method

WANG Zhiyuan\*, ZHANG Yiqing, QIU Cong, ZENG Zihua, WANG Yuqian

## Abstract

In this paper, we presents a study on the application of the Spectral Collocation Method for pricing American put options. Our primary objective is to implement this method and evaluate its performance under various conditions. We begin by implementing this algorithm and replicating Table 2 from the influential paper "High Performance American Option Pricing,".

The core of our study involves an in-depth analysis of the results obtained from the Spectral Collocation Method. We meticulously evaluate the accuracy, numerical stability, and convergence speed of this method. These factors are examined in the context of varying model parameters: the degree of the polynomial approximation ( $l, m, n$ ), the spot price ( $S$ ), interest rate ( $r$ ), dividend yield ( $q$ ), and time to maturity ( $\tau$ ). Such an analysis provides a nuanced understanding of the method's effectiveness and reliability across different market conditions. We also measured the stability using the smoothness of the greeks.

Furthermore, we extend our research by implementing the Crank-Nicolson method, a well-established technique in option pricing. This implementation allows us to perform a comparative analysis between the Crank-Nicolson and Spectral Collocation Methods. By juxtaposing these two methods, we shed light on their respective strengths and limitations, offering valuable insights into their applicability in various option pricing scenarios.

This project offers practical insights for financial practitioners interested in applying these techniques in real-world contexts.

## Contents

<b>1</b>	<b><i>Spectral Collocation Method</i></b>	<b>2</b>
1.1	<i>Review of the algorithm</i>	2
1.2	<i>Implementation of the algorithm</i>	5
1.3	<i>Other consideration and clarification</i>	5
1.3.1	First and last terms in Chebyshev Interpolation	6
1.3.2	Interpolation Remark	6
1.3.3	Adjusting negative values to zero	6
<b>2</b>	<b><i>Replicate Table 2 in the paper High Performance American Option Pricing</i></b>	<b>6</b>
<b>3</b>	<b><i>Performance Evaluation</i></b>	<b>7</b>
3.1	<i>Accuracy Tests</i>	8
3.2	<i>Numerical Stability</i>	10
3.3	<i>Convergence</i>	12
3.4	<i>Option Greeks</i>	12
<b>4</b>	<b><i>Crank Nicolson Method</i></b>	<b>13</b>
4.1	<i>Implementation of the method</i>	14
4.2	<i>Performance Evaluation and Comparison with Spectral Collocation</i>	16

---

\*Student IDs are listed as: Wang Zhiyuan: A0285411J, Zhang Yiqing: A0285527U, Qiu Cong: A0285372X, Zeng Zihua: A0250540U, Wang Yuqian: A0250497X

4.2.1	Accuracy . . . . .	17
4.2.2	Convergence . . . . .	18
4.2.3	Stability . . . . .	19

# 1 Spectral Collocation Method

In this section, we first give a review of this method (without proofs and deductions). And then we introduce some details we need to pay attention when implementing this method using python.

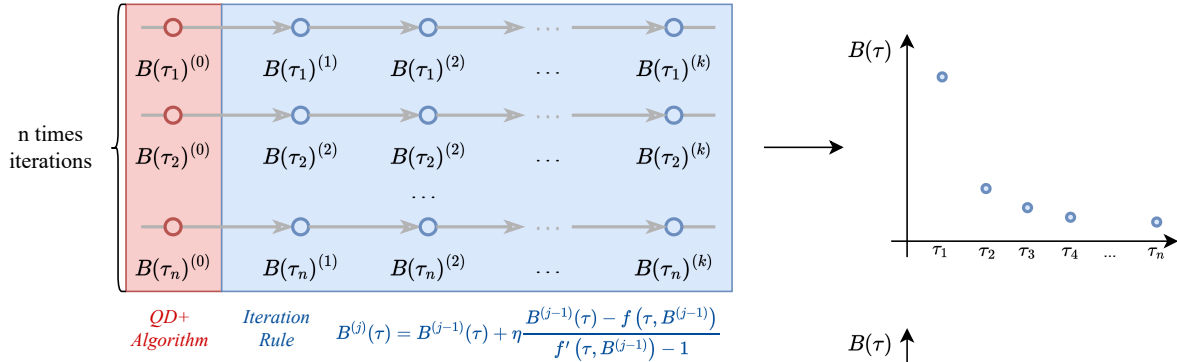
## 1.1 Review of the algorithm

The motivation and major steps of this algorithm can be illustrated by the following figure,

### Integration Representation

$$V(\tau, S) = v(\tau, S) + \underbrace{\int_0^\tau r K e^{-r(\tau-u)} \Phi(-d_-(\tau-u, S/B(u))) du}_{\text{Gauss Quadrature}} \underbrace{- \int_0^\tau q S e^{-q(\tau-u)} \Phi(-d_+(\tau-u, S/B(u))) du}_{\text{Fix point Iteration}}$$

### Fix point Iteration System



### Chebyshev Interpolation

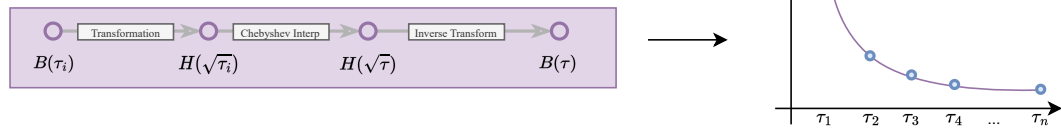


Figure 1.1: Illustration of the Algorithm

The spectral collocation method starts from the integral representation of the American put option,

$$V(\tau, S) = v(\tau, S) + \int_0^\tau r K e^{-r(\tau-u)} \Phi(-d_-(\tau-u, S/B(u))) du - \int_0^\tau q S e^{-q(\tau-u)} \Phi(-d_+(\tau-u, S/B(u))) du$$

In this expression, in order to calculate the price numerically we need two components: **a numerical integration method** and **the exercise boudary**  $B(\tau)$  .

For the numerical integration, we use **Gaussian-Quadrature** rule.

$$\int_{-1}^1 f(x)dx \approx \sum_{i=1}^n w_i f(x_i)$$

where the nodes and weights are fixed. For the computation of  $B(\tau)$ , it can be represented as a **fixed point problem**. Thus it can be calculated through the following iteration rule. However, to achieve this, we need 3 more things: **Collocation points**, **Initial Value** , and **Iteration rule**.

- **Collocation points**  $\{\tau_i\}_{i=1,\dots,n} = \{x_i^2\}_{i=1,\dots,n}$ , where  $x_i$  are chebyshev nodes.(will explain later)
- **Initial Value**  $B(\tau_i)_{i=1,\dots,n}^{(0)}$ : Set by using **QD+ Algorithm** given below:

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

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

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

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  $\Theta$  is the theta (time derivative) of the European put price:

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

- **Iteration rule:**

$$B^{(j)}(\tau) = B^{(j-1)}(\tau) + \eta \frac{B^{(j-1)}(\tau) - f(\tau, B^{(j-1)})}{f'(\tau, B^{(j-1)}) - 1}$$

where, where

$$\begin{aligned} f(\tau, B) &= K^*(\tau) \frac{N(\tau, B)}{D(\tau, B)} \\ f'(\tau, Q) &\triangleq \frac{\partial f}{\partial Q(\tau)} = K^*(\tau) \left( \frac{N'(\tau, Q)}{D(\tau, Q)} - \frac{D'(\tau, Q)N(\tau, Q)}{D(\tau, Q)^2} \right). \end{aligned}$$

with  $K^*(\tau) \triangleq K e^{-(r-q)\tau}$  and

$$\begin{aligned} N(\tau, B) &= \frac{\phi(d_-(\tau, B(\tau)/K))}{\sigma\sqrt{\tau}} + r\mathcal{K}_3(\tau) \\ D(\tau, B) &= \frac{\phi(d_+(\tau, B(\tau)/K))}{\sigma\sqrt{\tau}} + \Phi(d_+(\tau, B(\tau)/K)) + q(\mathcal{K}_1(\tau) + \mathcal{K}_2(\tau)) \\ \mathcal{K}_1(\tau) &= \int_0^\tau e^{qu} \Phi(d_+(\tau - u, B(\tau)/B(u))) du, \\ \mathcal{K}_2(\tau) &= \int_0^\tau \frac{e^{qu}}{\sigma\sqrt{\tau - u}} \phi(d_+(\tau - u, B(\tau)/B(u))) du, \\ \mathcal{K}_3(\tau) &= \int_0^\tau \frac{e^{ru}}{\sigma\sqrt{\tau - u}} \phi(d_-(\tau - u, B(\tau)/B(u))) du, \end{aligned}$$

$N'$  and  $D'$  indicate partial derivatives with respect to  $B(\tau)$ . Thus,

$$N'(\tau, Q) = -d_-(\tau, Q(\tau)/K) \frac{\phi(d_-(\tau, Q(\tau)/K))}{Q(\tau)\sigma^2\tau} \\ - r \int_0^\tau \frac{e^{ru} d_-(\tau, Q(\tau)/Q(u))}{Q(\tau)\sigma^2(\tau-u)} \phi(d_-(\tau-u, Q(\tau)/Q(u))) du$$

and

$$D'(\tau, Q) = -\frac{K^*(\tau)}{Q(\tau)} d_-(\tau, Q(\tau)/K) \frac{\phi(d_-(\tau, Q(\tau)/K))}{Q(\tau)\sigma^2\tau} \\ - q \frac{K^*(\tau)}{Q(\tau)} \int_0^\tau \frac{Q(u)}{K} \frac{e^{ru} d_-(\tau-u, Q(\tau)/Q(u))}{\sigma^2(\tau-u)} \phi(d_-(\tau-u, Q(\tau)/Q(u))) du$$

After doing the iteration, we have  $B(\tau_i)$ ,  $i = 1, \dots, n$ , which is the exercise boundary at the collocation points. However, in order to use the Gauss-Quadrature rule, we need to know the value of the exercise boundary at the quadrature points. Since we collocation points and quadrature points do not usually coincide, we need an interpolation method, for which we use **Chebyshev Interpolation**.

Since nodes in chebyshev interpolation are fixed, we choose the collocation points to be the chebyshev nodes. But according to the paper, here we do not interpolate on  $B(\tau)$  directly, instead we do a transformation from  $B(\tau)$  to  $H(\tau)$ ,

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

And,

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

Then, we perform a chebyshev interpolation to  $H(x_i)$  and then use the transformation given above to do a reverse transformation to get  $B(\tau)$ .

Since the chebyshev nodes are fixed, if we want to do a chebyshev interpolation we need to set  $\{\tau_i\}$  to be the chebyshev nodes initially, which **sets the collocation points**.

$$\{\tau_i\}_{i=1, \dots, n} = \{x_i^2\}_{i=1, \dots, n}$$

, where  $x_i$  are chebyshev nodes.

Also, notice that in the fixed point iteration, the iteration rules contain integration, which means that we need to do the Gauss Quadrature within each iteration. Thus, to get the values of  $B(\tau_i)$  at the Gauss Quadrature points, we need to conduct an interpolation **within each iteration**.

So, the procedures of pricing American put is actually the following steps,

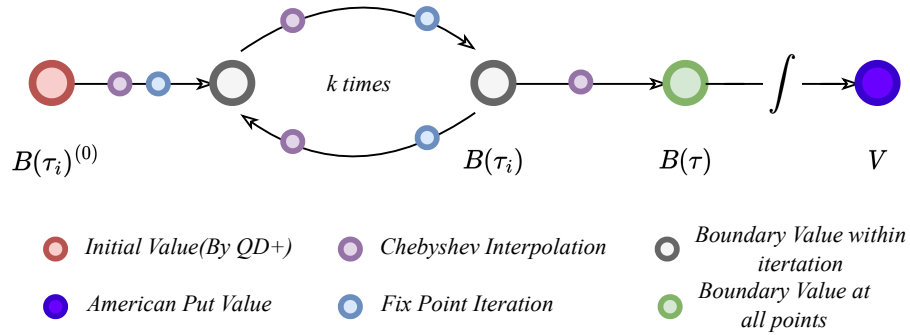


Figure 1.2: Major Steps of Spectral Collocation Method

For clarity, we summarize the algorithm as follows,

---

**Algorithm** Spectral Collocation Method

---

Input:  $S, K, r, q, \tau, \sigma, n, l, m, p$

Step 1: Set collocation points:  $\{\tau_i\}_{i=0}^n = \{x_i^2\}_{i=0}^n$

Step 2: Compute the quadrature nodes  $y_k$  and weights  $w_k$ , for  $k = 1, \dots, l$ .

Step 3: Use **QD+** to set initial guesses  $B(\tau_i)_{i=1, \dots, n}^{(0)}$

Step 4: **for**  $j \leftarrow 1$  to  $m$  **do**

    Set  $H(\sqrt{\tau}) = \ln(B^{(j-1)}(\tau)/X)^2$  and initialize the Chebyshev interpolation;

**for**  $i \leftarrow 1, \dots, n$  **do**:

        Use clenshaw algorithm to get  $B^{(j-1)}(\tau_i - \tau_i(1 + y_k)^2/4)$ ,  $k = 1, \dots, l$

        Calculate  $N(\tau_i, B^{(j-1)})$  and  $D(\tau_i, B^{(j-1)})$ .

        Compute  $f(\tau_i, B^{(j-1)})$ ,  $f'(\tau_i, B^{(j-1)})$

        Compute  $B^{(j)}(\tau_i)$  using fix point iteration;

Step 5: Use chebyshev interpolation to get  $B^{(m)}(\tau_i - \tau_i(1 + y_k)^2/4)$

Step 6: Calculate  $V(\tau, S)$  using integral representation formula.

Output: American put option price  $V(\tau, S)$

---

## 1.2 Implementation of the algorithm

For the implementation, we use python as our programming language. In order to make the program more readable and easier to use, we have written 2 separate scripts, one for the Spectral Collocation implementation([spectral\\_collocation.py](#)), the other for defining some helper functions and classes([util.py](#)) We illustrate their main structures in the following figure.

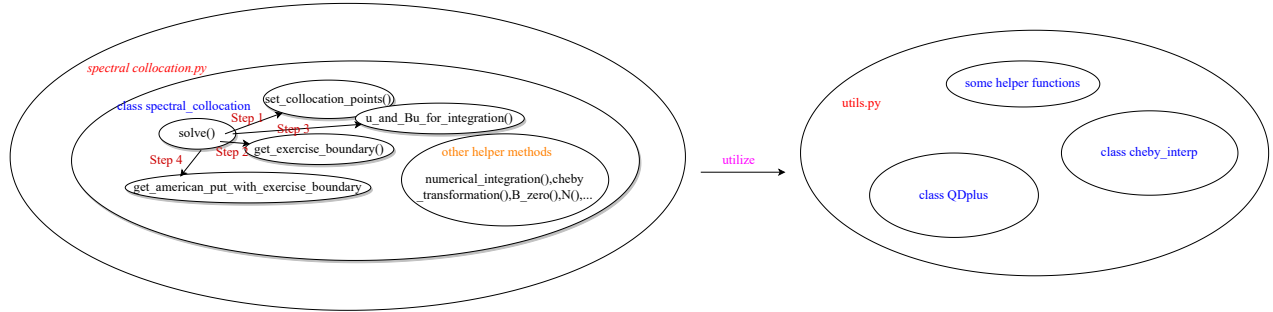


Figure 1.3: Python Scripts Illustration

As we have shown in the above figure. [spectral\\_collocation.py](#) defines a class named `class spectral_collocation` in which we use the `solve()` as the pricing function and it mainly depends on four methods: `set_collocation_points()` which sets the collocation points as the square of chebyshev nodes, `get_exercise_boundary()` which calculates  $B(\tau)$  by using fixed point iteration and numerical integration rules, `u_and_Bu_for_integration` which performs the last interpolation and gets the final exercise boundary, and `get_american_put_with_exercise_boundary()` which computes the price of American puts. Also, within this class, we have defined other small helper functions that are useful during the calculation. In [utils.py](#) we mainly define 2 classes addressing the QD+ and chebyshev interpolation problems.

## 1.3 Other consideration and clarification

In this section, we mainly introduce the issues we encounter and the solution we use during the implementation.

### 1.3.1 First and last terms in Chebyshev Interpolation

In equation (51) of the paper, it gives,

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

However, for  $a_k$ , it should be:

$$a_k = \frac{1}{n} [q_0 + (-1)^n q_n] + \frac{2}{n} \sum_{i=1}^{n-1} q_i \cos \frac{ik\pi}{n}$$

We believe this is a typo made by the author, thus in our implementation we use the correct formula (divide first and last terms by 2 instead of 4).

### 1.3.2 Interpolation Remark

We know that to do the interpolation we first transform  $B(\tau)$  to  $H(\sqrt{\tau})$  using,

$$H(\sqrt{\tau}) = \left( \ln \left( \frac{B(\tau)}{X} \right) \right)^2, \quad X = K \min(1, r/k) \quad (*)$$

To get  $B(\tau)$  from  $H(\sqrt{\tau})$  after doing the interpolation we need to do an inverse transformation from the above equation, notice that for American put options,

$$B(\tau) = X e^{-\sqrt{H(\sqrt{\tau})}}$$

And for American calls,

$$B(\tau) = X e^{+\sqrt{H(\sqrt{\tau})}}$$

This is because when taking square root to both sides of (\*). It can be positive and negative, if we take the positive for American put, the exercise boundary would be greater than strike price, which would be wrong.

### 1.3.3 Adjusting negative values to zero

During calculation, as we use Chebyshev interpolation there are chances that the value of interpolated  $B(u)$  might take negative values, which would be wrong if we use this negative value. For this problem, we fix these negative values by taking  $\max(0, B(u))$ .

## 2 Replicate Table 2 in the paper High Performance American Option Pricing

Here we conduct the pricing using the parameters given in table 2. The results are given as follows,

As we can see that from our implementation, the American Premium, Relative Error and CPU seconds were slightly different from the paper. Three reasons mainly contribute to this difference,

(1, m, n)	p	American Premium	Relative Error	CPU Seconds
(5, 1, 4)	15	0.10823642991146265	$1.2\text{E} - 02$	$4.5\text{E} - 01$
(7, 2, 5)	20	0. 10726530614443774	$2.9\text{E} - 03$	$7.9\text{E} - 05$
(11, 2, 5)	31	0. 1072674878475599	$2.9\text{E} - 03$	$1.4\text{E} - 00$
(15, 2, 6)	41	0. 10728595334718705	$3.1\text{E} - 03$	$1.5\text{E} - 01$
(15, 3, 7)	41	0.10704519601925888	$8.6\text{E} - 04$	$1.5\text{E} - 01$
(25, 4, 9)	51	0.10697876774680637	$2.4\text{E} - 04$	$3.4\text{E} - 01$
(25, 5, 12)	61	0.10697876774680637	$6.9\text{E} - 05$	$5.0\text{E} - 01$
(25, 6, 15)	61	0.10696015316986873	$2.0\text{E} - 05$	$8.0\text{E} - 01$
(35, 8, 16)	81	0. 10695484577771275	$1.6\text{E} - 06$	$1.6\text{E} - 00$
(51, 8, 24)	101	0.10695288271972636	$1.7\text{E} - 06$	$4.9\text{E} - 00$
(65, 8, 32)	101	0.10695288606740938	$1.7\text{E} - 06$	$2.6\text{E} + 01$

Table 1: 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  $p = 201$ . All results were computed using Gauss-Quadrature

- (a) **Chebyshev Interpolation:** As we indicated before, the first and last terms of  $a_k$  are different from the ones we actually use (difference between divide by 2 or 4 as stated in **Section 1.3**). This contributes to the difference of the American Premium and Relative Error.
- (b) **Quadrature Rules:** In the paper, tanh-sinh and Gauss quadrature rules are both used to enhance the accuracy. However, in our implementation we only used Gauss-Quadrature. This also cause the differences in results.
- (c) **Environment and Programming Language:** The paper used C++ as programming language to implement this method, but in our implementation we used python. This difference would cause the differences in CPU seconds, because python were usually slower as being a interpreted language.

### 3 Performance Evaluation

In numerical analysis and computer science, the terms "accuracy," "convergence speed," and "stability" are crucial for evaluating algorithms. Accuracy refers to how close the result of an algorithm is to the true or exact solution of the problem it's trying to solve. Convergence Speed refers to the rate at which the sequence of approximations generated by an algorithm approaches the exact solution.

Stability in numerical algorithms refers to their behavior in the presence of small **perturbations or errors**. An algorithm is stable if small changes in the input or intermediate steps do not lead to significantly large changes in the output. As in real world calculation, the input data are sometimes collected with small errors. A stable algorithm should shrink or at least do not amplify this small error in the input data so that the results would not differ significantly. If the algorithm is not stable, meaning that a slight perturbation in the input would lead to a significant change in the output, then it would be problematic if we use this algorithm to trading.

The FP-A method demonstrates high accuracy by approximately one order of magnitude, particularly noticeable when  $r \neq q$ . Despite this advantage, the FP-A method exhibits a susceptibility to generating oscillations, occasionally leading to significant numerical instability, particularly prevalent in scenarios with strongly drift-dominated dynamics, where the disparity between  $r$  and  $q$  is considerable relative to the volatility. This behavior is reminiscent of certain finite difference methods.

Previous versions of this approach explored strategies to address FP-A oscillations by introducing dampening factors within the fixed-point iterations. However, while these remedies aimed to mitigate oscillations, they sometimes resulted in rendering the FP-A method slower and more complex.

### 3.1 Accuracy Tests

In the first test, we use the following parameter ranges for a bulk setup (Table 2), where  $K = 100$  in all cases.

Parameter	Range
$r$	$\{0.02, 0.06, 0.10\}$
$q$	$\{0.04, 0.08, 0.12\}$
$S$	$\{25, 80, 100, 150, 200\}$
$T$	$\{1/12, 0.25, 0.5, 1\}$
$\sigma$	$\{0.1, 0.3, 0.5\}$

Table 2: Model and contract parameter ranges for timing and precision tests

(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	1.18E − 04	6.68E − 05	2.76E − 06	2.55E − 06	1.63E − 06
RRMSE	1.23E − 05	6.98E − 06	2.88E − 07	2.66E − 07	1.70E − 07
MAE	2.03E − 04	1.12E − 04	4.77E − 06	3.98E − 06	2.48E − 06
MRE	2.16E − 05	1.19E − 05	5.07E − 07	4.23E − 07	2.63E − 07

Table 3: Error measures on American put prices using Spectral Collocation Method

Our initial accuracy test results concerning the hyperparameters (l, m, n), and p indicate a general trend: higher hyperparameter values correspond to reduced error metrics. However, **we observe a threshold effect, notably when (l, m, n), p = (21, 6, 10), 41**. Beyond these values, further increments in hyperparameters exhibit diminishing improvements in error measures, leading to a plateau, while computational time escalates significantly. We report errors and timing results in Table 3 below.

Subsequently, our focus shifts to evaluating hyperparameter performance across three scenarios: in-the-money (ITM), at-the-money (ATM), and out-of-the-money (OTM). We endeavor to compare the discrepancy between the benchmark price and prices obtained under these three scenarios. The benchmark price of the American option is derived using benchmark hyperparameters (l, m, n), p = (131, 16, 64), 131. Introducing variations in the hyperparameters (l, m, n) for ITM, ATM, and OTM scenarios, we observe and tabulate the price differences in Table 5 and illustrate them in Figure 1.1.

(l, m, n)	ITM	ATM	OTM
(5, 2, 4)	1.51E − 04	3.73E − 04	1.63E − 05
(8, 4, 6)	1.71E − 05	2.70E − 05	7.21E − 07
(21, 6, 10)	1.40E − 06	2.17E − 06	3.71E − 08
(25, 8, 12)	1.20E − 07	1.91E − 07	3.29E − 09
(31, 16, 16)	3.09E − 09	4.47E − 09	1.55E − 10

Table 4: Relative Errors: The disparity in option prices under three scenarios: In-The-Money (ITM), At-The-Money (ATM), and Out-Of-The-Money (OTM), considering different hyperparameters (l, m, n). The respective spot prices for ITM, ATM, and OTM scenarios are designated as 50, 100, and 150.



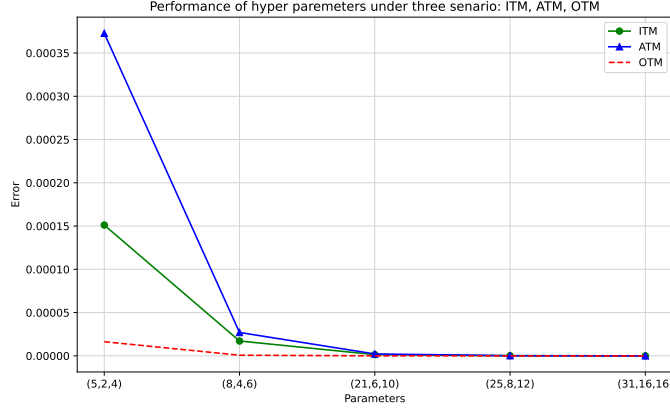


Figure 3.1: Accuracy under different situations-Spectral Collocation

This comprehensive analysis sheds light on how alterations in hyperparameters impact option pricing across different market situations. Based on the observations from Table 3 and Figure 1.1, a significant turning point is evident in the price difference concerning variations in different hyperparameters  $(l, m, n)$ . Broadly, larger values for  $(l, m, n)$  tend to yield smaller price differences. Notably, when  $(l, m, n) = (21, 6, 10)$ , the differences reach a minimum level, dropping to less than  $1E - 05$ . However, in practical financial markets, this level of precision, smaller than  $1E - 05$ , might not be fully significant, as the basic unit of price in financial markets typically operates around  $1E - 04$ . Hence, while the model exhibits an exceptional level of accuracy at these hyperparameter values, this ultra-high precision might not align with the practical nuances of real-world financial pricing.

In our final accuracy assessment, we delved into the accuracy concerning the time to maturity, differentiating between long-term and short-term scenarios. Specifically, we designated the longer time to maturity as one year and the shorter duration as one month (equivalent to  $1/12$  of a year). Subsequently, we computed the relative error for each option price across various hyperparameters  $(l, m, n)$ . The outcomes, detailed in Table 5, elucidate that as  $(l, m, n)$  approaches  $(21, 6, 10)$ , the relative error for longer timeframes falls below  $1E-05$ . Conversely, for shorter timeframes, the inflection point occurs at  $(l, m, n) = (8, 4, 6)$ . Remarkably, these findings underscore that option pricing tends to be more accurate for shorter time to maturity intervals, likely due to heightened market confidence and more reliable forecasting in these temporal scopes.

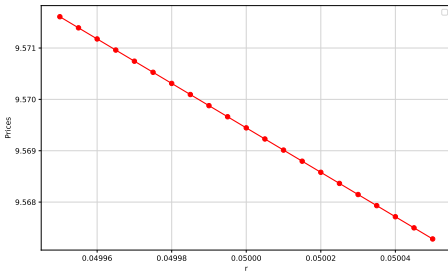
$(l, m, n)$	1 month	1 year
(5, 2, 4)	4.34E - 06	3.73E - 04
(8, 4, 6)	5.91E - 08	2.70E - 05
(21, 6, 10)	7.44E - 09	2.17E - 06
(25, 8, 12)	3.93E - 10	1.91E - 07
(31, 16, 16)	5.46E - 12	4.47E - 09

Table 5: Relative Errors in At-The-Money Option Prices Across Various Hyperparameters  $(l, m, n)$  for Different Time Horizons.

### 3.2 Numerical Stability

In this part, we undertake numerical stability tests involving variations in the spot price ( $S$ ), strike ( $K$ ), interest rate ( $r$ ), dividend ( $q$ ), volatility ( $\sigma$ ), and time to maturity ( $T$ ). Numerical stability is gauged based on whether the option price changes continuously with the gradual variation of these parameters. A stable computational method would exhibit a smooth transition in option price despite minor changes in these parameters.

Our benchmark parameters are set as follows:  $S = 100, K = 100, r = 0.05, q = 0.05, \sigma = 0.25, T = 1.0$ . We conduct calculations using our method with slight alterations in these parameters and present the outcomes in Figure 1.2. This graphical representation illuminates how the option price behaves in response to marginal adjustments in the specified parameters, providing insights into the stability of our computational approach for evaluating American option prices.



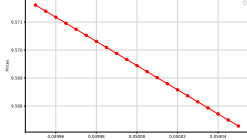
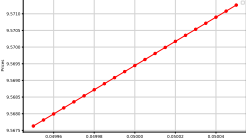
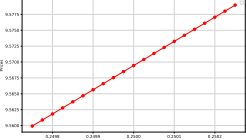
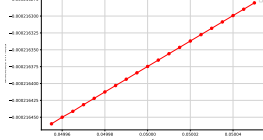
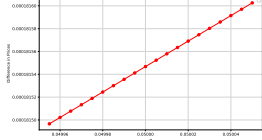
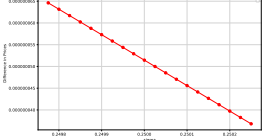
	$r$	$q$	$\sigma$
Prices			
Price Changes			
European Greeks	$-K\tau e^{-r\tau}\Phi(-d_2)$	$S\tau e^{-q\tau}\Phi(-d_1)$	$Ke^{-r\tau}\varphi(d_2)\sqrt{\tau}$
Signs Check	Correct	Correct	Correct
Stability	Stable	Stable	Stable

Table 6: Stability and Greeks Sign check w.r.t  $r, q, \sigma$

evident when minor increments of 1 base point are added to the input values, generating these observable patterns.

While the figures may seem to depict linear relationships due to the small increments and precision issues in the plotted values, the examination of first differences and the consistency of trends indicate that the apparent linearity is not representative of the underlying relationships. This stability test highlights the model's ability to yield continuous and coherent changes in option prices despite slight variations in the input parameters, further reinforcing its stability and reliability.

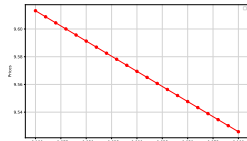
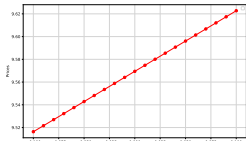
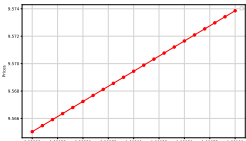
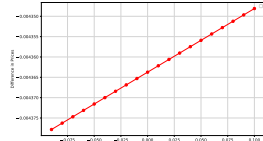
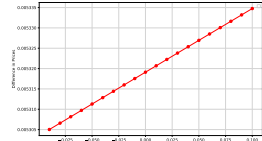
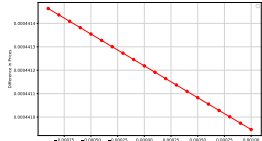
	$S$	$K$	$T$
Prices			
Price Changes			
European Greeks	$-e^{-q\tau}\Phi(-d_1)$	-	-Theta
Signs Check	Correct	Correct	Correct
Stability	Stable	Stable	Stable

Table 7: Stability and Greeks Sign check w.r.t  $S, K, T$

The outcomes from our tests indicate a consistent pattern: when each parameter undergoes minor variations within a small range, the option price exhibits a continuous change. This behavior underscores the stability of our computational method, signifying that slight alterations in the input parameters result in smooth and continuous adjustments in the calculated option price.

### 3.3 Convergence

In the third section, we delve into the convergence speed analysis of hyperparameters (l, m, n) while keeping p constant at 131. As inferred from the accuracy test results in the initial part, it's noticeable that the error measure regarding the difference in option price diminishes as the hyperparameters grow larger. Consequently, the option price tends to converge with larger hyperparameters.

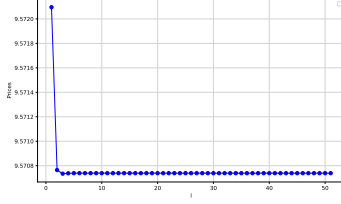


Figure 3.4: Convergence w.r.t l

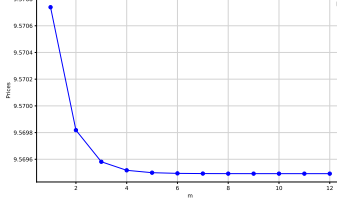


Figure 3.5: Convergence w.r.t m

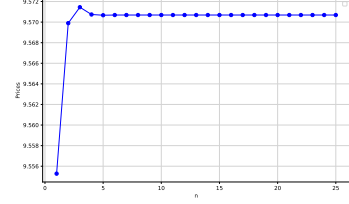


Figure 3.6: Convergence w.r.t n

To pinpoint when this convergence occurs, we conduct a specific test: when the difference between the option prices derived from the benchmark and the test falls below  $1e-05$ , we determine that convergence has been achieved. Figure 1.3 visually portrays these convergence trends. According to the  $1e-05$  criteria, we observe that the option price converges at  $(l, m, n) = (10, 8, 10)$ , correspondingly. Intriguingly, this highlights that among the hyperparameters, 'm' emerges as the most influential factor governing the convergence of the option price.

This finding emphasizes the pivotal role of the 'm' hyperparameter in influencing the convergence behavior of the option price concerning the chosen criteria. Additionally, it underscores the importance of carefully selecting and optimizing these hyperparameters to achieve accurate and stable results in pricing American options through numerical methods. Further exploration and fine-tuning of hyperparameters, particularly 'm', could potentially optimize convergence and enhance the computational efficiency of the pricing model.

### 3.4 Option Greeks

Greeks are vital in the realm of financial derivatives trading and risk management as they measure the sensitivity of the option's price to various factors, including changes in the underlying asset price, volatility, time decay, and interest rates.

For American options, the possibility of early exercise adds complexity to their valuation and risk profile. Greeks help in determining the scenarios where early exercise might be beneficial.

As we have discussed in the last section, the signs of greeks have been checked to be all correct. In this section, we calculated the greeks of American put option by using the spectral collocation methods in order to see how **smooth** the greeks are, as also this measures the stability of the algorithm if we use greeks for trading. Here due to high computational cost, the step in Stock Price is chosen to be 3. And each greek is calculated by using 1 bp change w.r.t the respective parameter.

We have calculated Delta( $\Delta = \frac{\partial V}{\partial S}$ ), Gamma( $\Gamma = \frac{\partial \Delta}{\partial S}$ ), Theta( $\Theta = \frac{\partial V}{\partial t}$ ), Rho( $\rho = \frac{\partial V}{\partial r}$ ) and Vega( $\mathcal{V} = \frac{\partial V}{\partial \sigma}$ ) and show the results in the following figure along with theoretical greeks for European puts

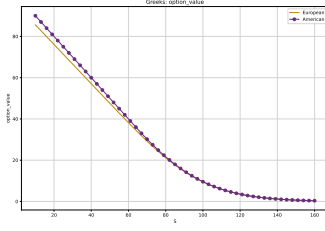


Figure 3.7: Option Value

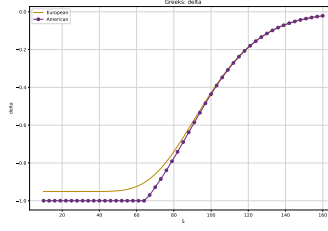


Figure 3.8: Delta

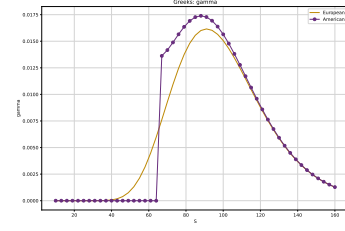


Figure 3.9: Gamma

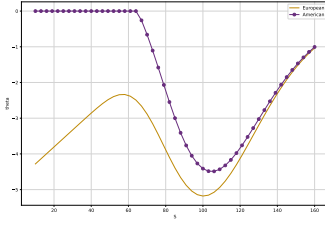


Figure 3.10: Theta

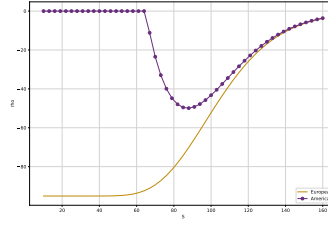


Figure 3.11: Rho

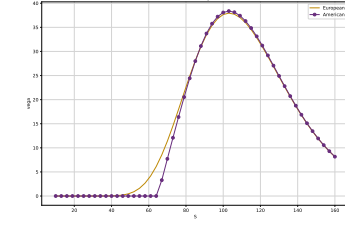


Figure 3.12: Vega

Figure 3.13: Greeks using Spectral Collocation: with parameter  $r = 0.05, q = 0.05, K = 100, \sigma = 0.25, T = 1$  with change of 1 base point in each calculation and  $S$  varying from 10 to 160 with step 3.

We analyze the figures in the following points,

- **Stability of the Algorithm** : From the above figure, we can observe that the Greeks of American option are relatively smooth, which indicates the **stability** of spectral collocation methods in some sense.
- **Exercise Boundary at initial time**: We can also observe that the exercise boundary is around 60 at initial time. As the option value for  $S < 60$  is a straight line ( $K - S$ ) for exercising the option immediately. Also, for  $S$  below 60, all Greeks shown are taking constant values either -1 or 0. Since we immediately exercise the option for a small  $S$  (below around 60). The price would be  $K - S$ . For Delta, it would be -1 and for other Greeks it would be 0.
- **Discontinuity of Gamma**: We can clearly see that there is discontinuity in the plot of Gamma, this is due to the immediate exercise when  $S$  are small. As in the plot of Delta, it performs a hocky stick shape by being -1 for  $S$  below the exercise boundary and other non -1 values. Since Gamma is the derivative of Delta, it would take zero values for  $S$  below the exercise boundary and a jump at the exercise boundary point. This is a result of American options' early exercise property and does not mean that the algorithm is not stable.
- **Different shapes from European for Rho**: Similar to our analysis with Gamma, when  $S$  are below the exercise boundary, Rho would be zero as well. And for European Rho, we know that it presents a normal c.d.f shape, and when  $S \rightarrow \infty$ , Rho approaches 0. Thus, for American put rho, it would be zero for both small and quite large  $S$ , thus it must go down first and then go up when  $S$  is large, which gives the shape we see above.

## 4 Crank Nicolson Method

In this section, we implement the Crank Nicolson Method to pricing American put options. We first provide an overview of this method and then we analyze and compare the accuracy, stability and convergence speed

of this algorithm under a similar framework as we did before with spectral collocation method.

## 4.1 Implementation of the method

Black-Scholes-Merton equation can be written as:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0$$

Finite Difference methods involve discretizing equations and boundary value problems by substituting difference for derivatives. For a time range set to  $(0, T)$ , set the range of the underlying asset price to  $(0, 4\max(S_0, K))$ , and then design a grid in the region  $\{0 \leq t \leq T, 0 \leq S \leq 4\max(S_0, K)\}$ . Divide the semi-infinite line  $0 \leq S \leq 4\max(S_0, K)$  into equal parts with a spacing of  $\Delta S$ , and divide the segment  $0 \leq t \leq T$  with a spacing of  $\Delta t$ . Denote the grid points as  $(S_m, t_n)$ :

In the spatial dimension, we use central differences, expressed as follows:

$$\begin{aligned} \frac{\partial V_j(t_j)}{\partial S} &\approx \frac{V_{j+1}(t_j) - V_{j-1}(t_j)}{2\Delta S} \\ \frac{\partial^2 V_j(t_j)}{\partial S^2} &\approx \frac{V_{j+1}(t_j) - 2V_j(t_j) + V_{j-1}(t_j)}{\Delta S^2} \end{aligned}$$

By substituting the **backward difference** in time into the PDE, we obtain the following **explicit difference** format:

$$\frac{V_j(t_{j+1}) - V_j(t_j)}{\Delta t} + rS_j \frac{V_{j+1}(t_{j+1}) - V_{j-1}(t_{j+1})}{2\Delta S} + \frac{1}{2}\sigma^2 S_j^2 \frac{V_{j+1}(t_{j+1}) - 2V_j(t_{j+1}) + V_{j-1}(t_{j+1})}{\Delta S^2} - rV_j(t_{j+1}) = 0$$

which simplifies to

$$V_j(t_j) = \left[ \left( \frac{\sigma^2 S_j^2}{2\Delta S^2} - \frac{rS_j}{2\Delta S} \right) \Delta t \right] V_{j-1}(t_{j+1}) + \left[ 1 + \left( -2\frac{\sigma^2 S_j^2}{2\Delta S^2} - r \right) \Delta t \right] V_j(t_{j+1}) + \left[ \left( \frac{\sigma^2 S_j^2}{2\Delta S^2} + \frac{rS_j}{2\Delta S} \right) \Delta t \right] V_{j+1}(t_{j+1})$$

This can be viewed as,

$$\underbrace{V_j(t_j)}_{\text{Value at } t_j} = l_j \Delta t \times \underbrace{V_{j-1}(t_{j+1})}_{\text{Value at } t_{j+1}} + [1 + c_j \Delta t] \times \underbrace{V_j(t_{j+1})}_{\text{Value at } t_{j+1}} + u_j \Delta t \times \underbrace{V_{j+1}(t_{j+1})}_{\text{Value at } t_{j+1}}$$

which means that the value at a previous time point is the weighted sum of the values of three later times. Since B-S equation is a Backward Equation, this method gives a direct way of calculating, thus explicit.

By substituting the **forward difference** in time into the PDE, we obtain the following **implicit difference** format:

$$\frac{V_j(t_{j+1}) - V_j(t_j)}{\Delta t} + rS_j \frac{V_{j+1}(t_j) - V_{j-1}(t_j)}{2\Delta S} + \frac{1}{2}\sigma^2 S_j^2 \frac{V_{j+1}(t_j) - 2V_j(t_j) + V_{j-1}(t_j)}{\Delta S^2} - rV_j(t_j) = 0$$

which simplifies to:

$$V_j(t_{j+1}) = \left[ \left( \frac{rS_j}{2\Delta S} - \frac{\sigma^2 S_j^2}{2\Delta S^2} \right) \Delta t \right] V_{j-1}(t_j) + \left[ 1 + \left( 2\frac{\sigma^2 S_j^2}{2\Delta S^2} + r \right) \Delta t \right] V_j(t_j) + \left[ \left( -\frac{rS_j}{2\Delta S} - \frac{\sigma^2 S_j^2}{2\Delta S^2} \right) \Delta t \right] V_{j+1}(t_j)$$

This can be viewed as,

$$\underbrace{V_j(t_{j+1})}_{\text{Value at } t_{j+1}} = -l_j \Delta t \times \underbrace{V_{j-1}(t_j)}_{\text{Value at } t_j} + [1 - c_j \Delta t] \times \underbrace{V_j(t_j)}_{\text{Value at } t_j} - u_j \Delta t \times \underbrace{V_{j+1}(t_j)}_{\text{Value at } t_j}$$

which means that the value at a later time point is the weighted sum of the values of three previous times. Since B-S equation is a Backward Equation, this method implies that we want to use the value of one point to get the value of three points, which cannot be explicitly calculated and need to solve a linear equation system, thus implicit.

In the explicit and implicit difference methods, we use  $(t_j, S_j)$  as an example to explain the discretization process, which represents an **actual** point in the grid. In the  $\theta$ -method, we discretize on a point that does not exist in the grid, denoted as  $(t_{j,j+1}^\theta, S_j)$ , where  $t_{j,j+1}^\theta = \theta t_j + (1 - \theta)t_{j+1}$ .  $\theta$ -method is a more general method as:

- When  $\theta = 0$ ,  $t_{j,j+1}^\theta = t_{j+1}$ , in this case, the  $\theta$ -method is the explicit difference method.
- When  $\theta = 1$ ,  $t_{j,j+1}^\theta = t_j$ , in this case, the  $\theta$ -method is the implicit difference method.
- When  $\theta = 0.5$ ,  $t_{j,j+1}^\theta$  is the midpoint of  $t_j$  and  $t_{j+1}$ , in this case, the  $\theta$ -method is also known as the Crank-Nicolson (CN) method.

To shorten the formula length,  $\theta_a$  and  $\theta_b$  represent  $\theta$  and  $1 - \theta$ , respectively.

$$\begin{aligned} & \frac{V_j(t_{j+1}) - V_j(t_j)}{\Delta_t} + rS_j \left[ \theta_a \frac{V_{j+1}(t_j) - V_{j-1}(t_j)}{2\Delta_S} + \theta_b \frac{V_{j+1}(t_{j+1}) - V_{j-1}(t_{j+1})}{2\Delta_S} \right] \\ & + \frac{1}{2}\sigma^2 S_j^2 \left[ \theta_a \frac{V_{j+1}(t_j) - 2V_j(t_j) + V_{j-1}(t_j)}{\Delta_S^2} + \theta_b \frac{V_{j+1}(t_{j+1}) - 2V_j(t_{j+1}) + V_{j-1}(t_{j+1})}{\Delta_S^2} \right] \\ & - r [\theta_a V_j(t_j) + \theta_b V_j(t_{j+1})] = 0 \end{aligned}$$

Upon rearrangement, we obtain:

$$\begin{aligned} & \left[ \left( \frac{rS_j}{2\Delta_S} - \frac{\sigma^2 S_j^2}{2\Delta_S^2} \right) \theta_a \Delta_t \right] V_{j-1}(t_j) + \left[ 1 + \left( 2 \frac{\sigma^2 S_j^2}{2\Delta_S^2} + r \right) \theta_a \Delta_t \right] V_j(t_j) + \left[ \left( \frac{rS_j}{2\Delta_S} + \frac{\sigma^2 S_j^2}{2\Delta_S^2} \right) \theta_a \Delta_t \right] V_{j+1}(t_j) = \\ & \left[ \left( \frac{\sigma^2 S_j^2}{2\Delta_S^2} - \frac{rS_j}{2\Delta_S} \right) \theta_b \Delta_t \right] V_{j-1}(t_{j+1}) + \left[ 1 + \left( -2 \frac{\sigma^2 S_j^2}{2\Delta_S^2} - r \right) \theta_b \Delta_t \right] V_j(t_{j+1}) + \left[ \left( -\frac{\sigma^2 S_j^2}{2\Delta_S^2} - \frac{rS_j}{2\Delta_S} \right) \theta_b \Delta_t \right] V_{j+1}(t_{j+1}) \end{aligned}$$

Thus,

$$\begin{aligned} & -l_j \theta_a \Delta_t \times \underbrace{V_{j-1}(t_j)}_{\text{Value at } t_j} + [1 - c_j \theta_a \Delta_t] \times \underbrace{V_j(t_j)}_{\text{Value at } t_j} - u_j \theta_a \Delta_t \times \underbrace{V_{j+1}(t_j)}_{\text{Value at } t_j} \\ & = l_j A_b \Delta_t \times \underbrace{V_{j-1}(t_{j+1})}_{\text{Value at } t_j} + [1 + c_j \theta_b \Delta_t] \times \underbrace{V_j(t_{j+1})}_{\text{Value at } t_{j+1}} + u_j A_b \Delta_t \times \underbrace{V_{j+1}(t_{j+1})}_{\text{Value at } t_{j+1}} \end{aligned}$$

In explicit difference methods, each equation derives one unknown from three known values, so an arithmetic equation can be used to find one value. Unlike explicit difference methods, implicit difference methods derive three unknowns from one known value in each equation and must be solved by integrating into a matrix. Explicit and implicit differences are special forms of the Crank-Nicolson method, as shown in the following figure,

As for the implementation, we use the following BSM equation,

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + \delta S \frac{\partial V}{\partial S} - rV$$

where  $\tau = T - t$ ,  $\delta = r - q$ . From Crank-Nicolson method, we could derive:

$$-\frac{1}{4}\Delta\tau \left( \left( \frac{\sigma S}{\Delta S} \right)^2 - \frac{(r-q)S}{\Delta S} \right) V_{i+1,j-1} + \left( 1 + \frac{1}{2}\Delta\tau \left[ \left( \frac{\sigma S}{\Delta S} \right)^2 + r \right] \right) V_{i+1,j} - \frac{1}{4}\Delta\tau \left( \left( \frac{\sigma S}{\Delta S} \right)^2 + (r-q) \frac{S}{\Delta S} \right) V_{i+1,j+1}$$

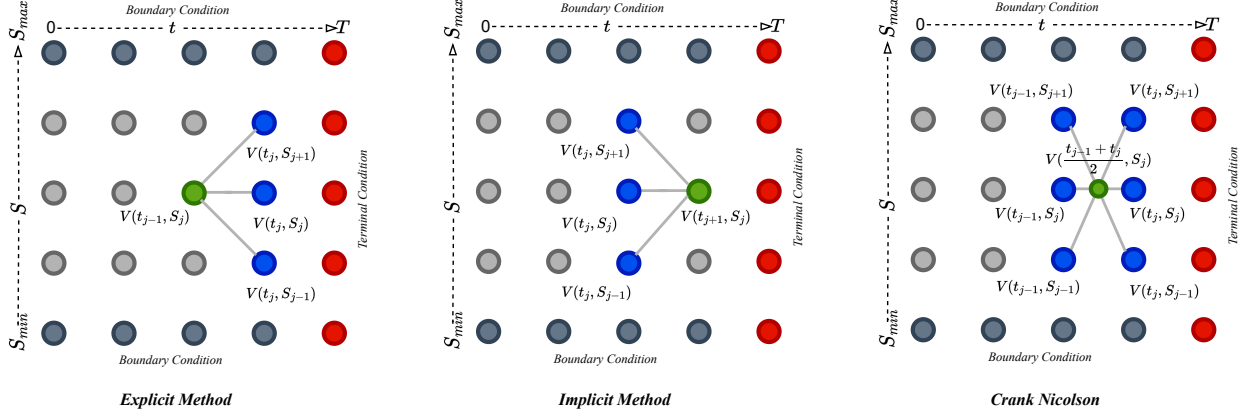


Figure 4.1: Finite Difference Methods

$$= \frac{1}{4} \Delta \tau \left( \left( \frac{\sigma S}{\Delta S} \right)^2 - \frac{(r-q)S}{\Delta S} \right) V_{i,j-1} + \left( 1 - \frac{1}{2} \Delta \tau \left[ \left( \frac{\sigma S}{\Delta S} \right)^2 + r \right] \right) V_{i,j} + \frac{1}{4} \Delta \tau \left( \left( \frac{\sigma S}{\Delta S} \right)^2 + (r-q) \frac{S}{\Delta S} \right) V_{i,j+1}$$

To simplify it, let  $\alpha = \frac{1}{4} \Delta \tau \left( \left( \frac{\sigma S}{\Delta S} \right)^2 - \frac{(r-q)S}{\Delta S} \right)$ ,  $\beta = \frac{1}{2} \Delta \tau \left( \left( \frac{\sigma S}{\Delta S} \right)^2 + r \right)$ ,  $\gamma = \frac{1}{4} \Delta \tau \left( \left( \frac{\sigma S}{\Delta S} \right)^2 + (r-q) \frac{S}{\Delta S} \right)$  and  $b_{i,j} = \alpha V_{i,j-1} + (1 - \beta) V_{i,j} + \gamma V_{i,j+1}$ , we have

$$-\alpha V_{i+1,j-1} + (1 + \beta) V_{i+1,j} - \gamma V_{i+1,j+1} = b_{i,j}$$

for  $0 \leq i \leq M - 2$  and  $1 \leq j \leq N - 2$ .

From lecture 3, the SOR for Crank-Nicolson is

$$V_i^{(k+1)} = (1 - \omega) V_i^k + \frac{\omega}{a_{i,i}} \left( b - \sum_{j < i} a_{i,j} V_j^{(k+1)} - \sum_{j > i} a_{i,j} V_j^k \right)$$

And for American put options, the initial condition is

$$V(0, S) = \max\{K - S, 0\}$$

Boundary conditions

$$V(\tau, S) = 0, \quad S \rightarrow \infty$$

$$V(\tau, 0) = e^{-r\tau}$$

Projected SOR

$$V_i^{(k+1)} = \max\left\{ (1 - \omega) V_i^k + \frac{\omega}{a_{i,i}} \left( b - \sum_{j < i} a_{i,j} V_j^{(k+1)} - \sum_{j > i} a_{i,j} V_j^k \right), K - S \right\}$$

for  $i = 1, \dots, N - 2$ .

We implement this method using python, and analyze the results compared with spectral collocation methods in the following sections.

## 4.2 Performance Evaluation and Comparison with Spectral Collocation

In this section, we discuss the performance of the Crank-Nicolson algorithm from three aspects: accuracy, convergence, and stability, and compare its performance with the performance of the Spectral Collocation algorithm introduced earlier.



### 4.2.1 Accuracy

The accuracy of the finite difference methods are mainly affected by the truncation errors arising from the finite difference approximation in the Taylor series expansions. The Crank-Nicolson approximation is more accurate than implicit or explicit finite difference approximations. Theoretically it is accurate up to  $O((\Delta t)^2, (\Delta S)^2)$ , which is not bad. As mentioned in the previous section,  $N$  and  $M$  are two important parameters for the Crank-Nicolson algorithm.  $N$  represents the number of divisions in space, that is, how many intervals  $S$  is divided into, and  $M$  represents the number of divisions in time, that is, how many intervals  $T$  is divided into. As  $N$  and  $M$  increase, the accuracy of the CN algorithm increases. In order to better measure the accuracy of the algorithm and compare it with the Spectral Collocation algorithm, we perform bulk test on the algorithm. We use the following parameter ranges:

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

By implementing bulk test, rather than reporting individual pricing errors, we report the RMSE, relative RMSE(RRMSE), MAE, MRE across all option prices. The benchmark of the price is the high-precision "Exact" estimate (from Spectral Collocation algorithm with  $(l, m, n) = (131, 16, 64)$ ). The result are as below:??

	(100, 100)	(250, 250)	(500, 500)
RMSE	0.029746	0.007065	0.005655
RRMSE	0.003111	0.000739	0.000591
MAE	0.197737	0.025961	0.024781
MRE	0.082757	0.010865	0.010371

Table 8: Accuracy of Crank-Nicolson

From the figure, we can see that as the parameters  $(N, M)$  increase from  $(100, 100)$  to  $(500, 500)$ , the accuracy of the Crank-Nicolson algorithm is indeed constantly improving. Each of the four measures of error decreases monotonically. When  $(N, M)$  equals  $(500, 500)$ , the absolute value of the algorithm error is approximately  $5e^{-3}$ . However, compared with the accuracy results of the Spectral Collocation algorithm in the previous section, it can be seen that the accuracy of the Crank-Nicolson algorithm is far less than that of the Spectral Collocation algorithm. When the parameters of the Spectral Collocation algorithm are only set to  $(l, m, n) = (5, 2, 4)$ , the accuracy of the algorithm has reached  $1e^{-5}$ .

Afterwards, we study the accuracy of the Crank-Nicolson algorithm under different initial underlying prices. We habitually classify it into three situations: In the moment (ITM), At the moment (ATM), and Out the moment (OTM). In these three situations, we respectively observe the decrease of the error as the  $(N, M)$  parameters increase. The results are shown in the figure below:

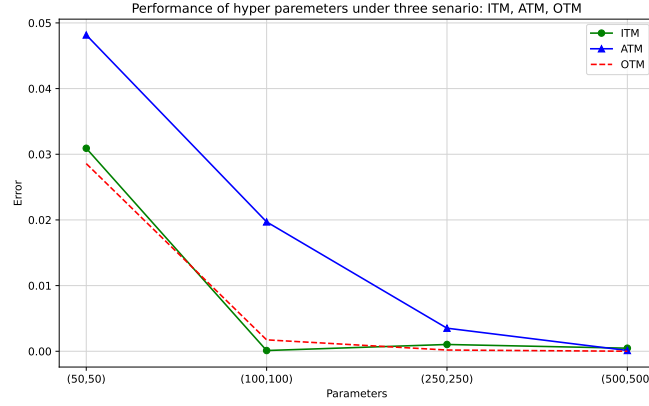


Figure 4.2: Accuracy Under different situations-Crank Nicolson

	ITM	ATM	OTM
(50, 50)	0.030913	0.048167	0.028577
(100, 100)	0.000113	0.019701	0.001737
(250, 250)	0.001034	0.003515	0.000187
(500, 500)	0.000469	0.000121	0.000005

Table 9: Error under different situation

It can be seen from the figure that when the initial underlying price is at ATM, that is,  $S = K$ , the calculation error of the CN algorithm is the largest. However, as the  $(N, M)$  parameters of the algorithm increase, the errors gradually decrease to about  $1e^{-4}$ , and the errors in different situations are not much different. Therefore, the initial price of the underlying has almost no impact on the accuracy of the algorithm when the  $(N, M)$  parameters are large enough. Similarly, the results of the Spectral Collocation algorithm also show that the initial underlying price has little impact on the accuracy of the algorithm. However, from the perspective of absolute error, the Spectral Collocation algorithm is still significantly better than the Crank-Nicolson algorithm.

#### 4.2.2 Convergence

For convergence, one measure is the speed of convergence under similar parameters and conditions. At this point, Crank-Nicolson has advantages over the explicit difference method and the implicit difference method, which are both finite difference methods. Another measure focuses on the impact of parameter changes on model pricing. In the Crank-Nicolson algorithm, as the parameters  $(N, M)$  increase, the accuracy of the algorithm is improved, but when a certain threshold is reached, continuing to increase the parameters  $(N, M)$  will become very limited in improving the algorithm. For example, suppose we set an error of  $1e^{-5}$ . When  $N$  and  $M$  reach a certain threshold, the improvement in accuracy of the algorithm by continuing to increase  $(N, M)$  will be less than  $1e^{-5}$ . We call the model convergence at this time. The convergence speed is expressed as the degree to which the parameters need to be improved compared to the baseline in order to achieve the set error. We discuss  $N$  and  $M$  separately.

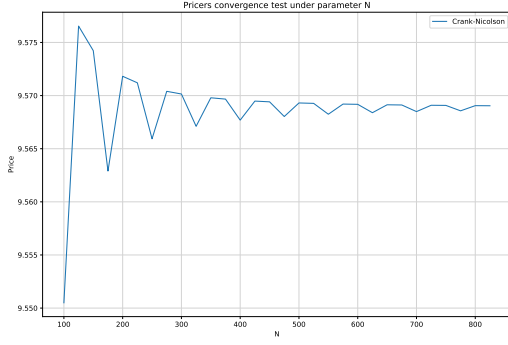


Figure 4.3: Convergence under N

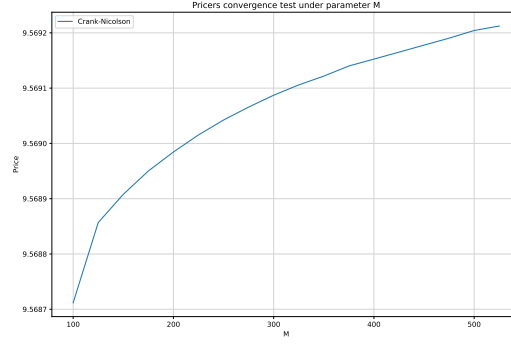


Figure 4.4: Convergence under M

Convergence Speed	
When $M = 500$ , Model Converge at	$N = 825$
When $N = 500$ , Model Converge at	$M = 525$

Table 10: Convergence Speed Summary

We can see from the figures and tables that when the error is set to  $1e^{-5}$ , increasing the parameter M will make the algorithm converge faster. When N is equal to 500, M only needs to be increased to 525 for the algorithm to converge. For parameter N, when M is equal to 500, N needs to reach 825 before the algorithm converges. It can be seen that the **change of parameter M has a greater impact** on the accuracy and convergence of the algorithm. In addition, compared with the discussion on the convergence of the Spectral Collocation algorithm in the previous section, it can be seen that when the error is set to  $1e^{-10}$ , the parameters  $l$ ,  $m$ , and  $n$  only need to reach 51, 12, and 25 respectively to make the algorithm converge, it can be seen that the **convergence speed of the Spectral Collocation algorithm is also better than the Crank-Nicolson algorithm**.

### 4.2.3 Stability

**Pricing Stability** We first examine the pricing stability of the algorithm. Factors affecting the price of American options include  $K$ ,  $r$ ,  $q$ ,  $T$  (time to maturity),  $\sigma$ ,  $S$  (underlying price), etc. Therefore, an important criterion for measuring the stability of the algorithm is that if a small bias is applied to one of these financial variables (we set it from -10bp to 10bp), how much influence will it have on the pricing results given by the model. If the impact is too large, then the algorithm lacks stability, because it means that a small error in the input will have a large impact on the result.

Similar to our previous discussion, the seemingly linear relation is deceptive, we take  $r$  as an example,

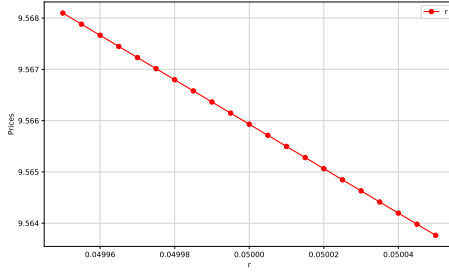


Figure 4.5: Put prices under different  $r$

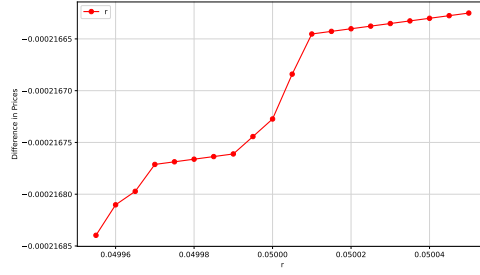


Figure 4.6: 1st Difference of the put prices

From the figure we see that the first difference in price does not show a constant value, which means that the prices are not perfectly linearly related with  $r$ . Also, the difference plot presents a similar shape to greeks, we know a in a small range for  $r$  a stable algorithm would give a linear-look plot (as spectral collocation). Thus, Crank-Nicolson method is less stable in this sense. In the following tables we present the comparison of Crank-Nicolson method and spectral collocation method.

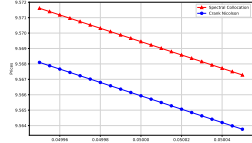
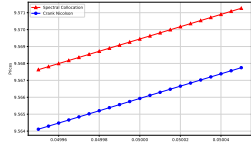
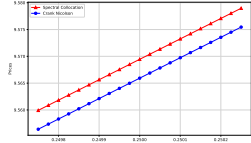
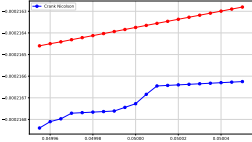
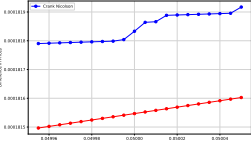
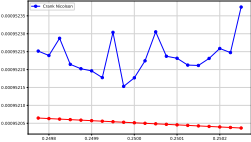
	$r$	$q$	$\sigma$
Prices			
Price Changes			
European Greeks	$-K\tau e^{-r\tau}\Phi(-d_2)$	$S\tau e^{-q\tau}\Phi(-d_1)$	$Ke^{-r\tau}\varphi(d_2)\sqrt{\tau}$
Signs Check	Correct	Correct	Correct
Stability	Less Stable	Less Stable	Less Stable

Table 11: Stability and Greeks Sign check w.r.t  $r, q, \sigma$

From the figure, we can see that no matter it is the Crank-Nicolson algorithm or the Spectral Collocation algorithm, small changes in any of the variables  $r$ ,  $q$ ,  $T$  (time to maturity),  $\sigma$ ,  $S$  (underlying price) will not have much impact on the pricing results. Among these variables, the most influential one is  $\sigma$ , which changes from -10bp to 10bp, causing a change of about  $2e^{-3}$  in the price. Therefore, both algorithms perform well in terms of price stability. In addition, comparing the Crank-Nicolson algorithm and the Spectral Collocation algorithm, we can see that for any variable, the stability performance of the two algorithms is not much different. Therefore, the two algorithms are relatively similar in terms of price stability, but with **spectral collocation algorithm more stable** as it shows a more stable pattern in the first difference.

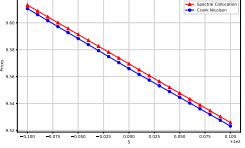
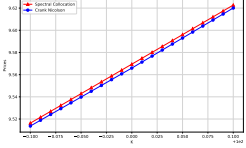
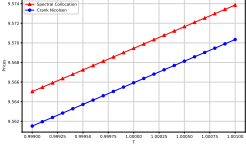
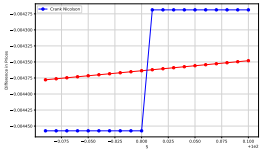
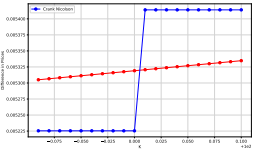
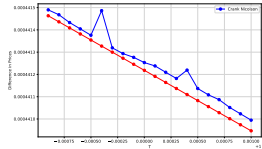
	$S$	$K$	$T$
Prices			
Price Changes			
European Greeks	$-e^{-q\tau}\Phi(-d_1)$	-	-Theta
Signs Check	Correct	Correct	Correct
Stability	Less Stable	Less Stable	Less Stable

Table 12: Stability and Greeks Sign check w.r.t  $S, K, T$

	$r$	$S$	$q$	$\sigma$	$T$
Crank-Nicolson	-0.000453	-0.009080	0.000380	0.001993	0.000923
Spectral Collocation	-0.000451	-0.009078	0.000378	0.001992	0.000923

Table 13: Stability Test

**Greeks** As we have discussed before, a good pricing model should obviously guarantee the stability of Greek value calculations.

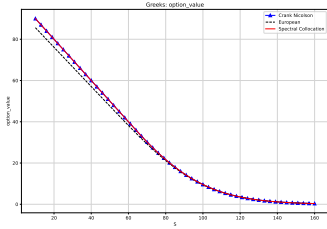


Figure 4.7: Option Value

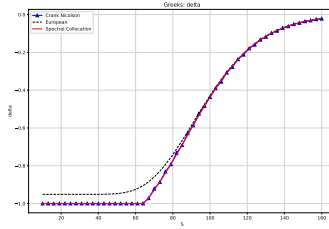


Figure 4.8: Delta

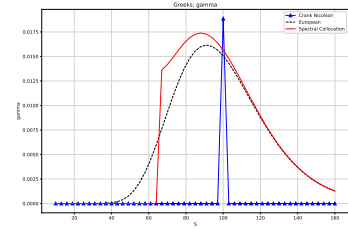


Figure 4.9: Gamma

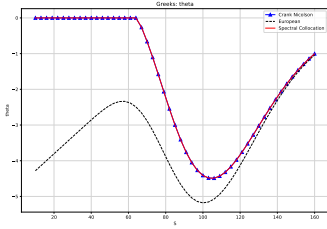


Figure 4.10: Theta

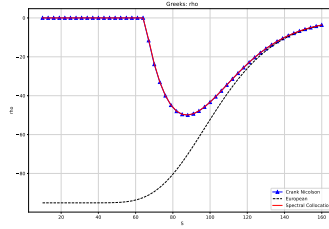


Figure 4.11: Rho

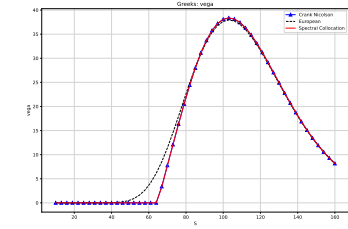


Figure 4.12: Vega

Figure 4.13: Greeks using Crank Nicolson: with parameter  $r = 0.05, q = 0.05, K = 100, \sigma = 0.25, T = 1$  with change of 1 base point in each calculation and  $S$  varying from 10 to 160 with step 3.

We calculated four Greek values using the Crank-Nicolson algorithm: Delta ( $\Delta$ ), Theta ( $\Theta$ ), Rho ( $\rho$ ), Vega( $\mathcal{V}$ ). As can be seen from the above figure, all greeks except for gamma are quite close to the results from spectral collocation methods. But still, gamma under Crank Nicolson is relatively unstable, thus underperforms Spectral Collocation Methods.

## References

- [1] Andersen, Leif BG, Mark Lake, and Dimitri Offengenden. "High performance American option pricing." Available at SSRN 2547027 (2015).
- [2] FastAmericanOptionPricing: <https://github.com/antddvid/FastAmericanOptionPricing>