FOURIER TRANSFORM METHODS FOR OPTION PRICING: AN APPLICATION TO EXTENDED HESTON-TYPE MODELS

Master Thesis in Quantitative Finance and Banking

Gorka Koldo González Sáez

Universidad del País Vasco

Academic advisors:
Federico Platanía[1], Manuel Moreno F.[2]

[1]Dept. of Quantitative Economics
Economic Faculty
Univ. Complutense of Madrid
Spain

[2]Dept. of Economic Analysis
Social & Legal Sciences Faculty
Univ. of Castilla-La Mancha
Spain

Submitted: 10/07/2014

FOURIER TRANSFORM METHODS FOR OPTION PRICING: AN APPLICATION TO EXTENDED HESTON-TYPE MODELS

Short abstract:
The main purpose of this master thesis it to show that Fourier transform methods can be applied to Option Pricing theory to reduce the computational time compared with other methodologies like Monte Carlo, when we price European vanilla options considering the following models: Heston, Bates, SVJJ, Double Heston and Time Dependent Heston.

Keywords: Fourier transform, FFT, FRFT, Heston, Bates, SVJJ

Contents

List of Figures vii

List of Tables ix

Acronyms xi

1 Overview of Fourier Transform in Finance 1
- 1.1 Introduction . . . 2
- 1.2 The Fourier Transform . . . 2
- 1.3 Gil-Peláez (1951) Inversion Theorem . . . 3
- 1.4 Carr and Madan (1999) Formulation . . . 5
  - 1.4.1 The Fourier Transform of an Option Price . . . 5
  - 1.4.2 Fourier Transform of Out-of-the-Money Option Prices . 8

2 Pricing Methods 11
- 2.1 Introduction . . . 12
- 2.2 Direct Integration Method . . . 12
- 2.3 Euler Monte Carlo Method . . . 13
- 2.4 Fast Fourier Transform Method . . . 14
- 2.5 Fractional Fast Fourier Transform Method . . . 17

3 The Models 19
- 3.1 Introduction . . . 20
- 3.2 The Heston (1993) Model . . . 20
  - 3.2.1 Characteristic Function . . . 21
  - 3.2.2 Numerical Results . . . 22
- 3.3 The Bates (1996) Model . . . 27
  - 3.3.1 Characteristic Function . . . 27
  - 3.3.2 Numerical Results . . . 28
- 3.4 The SVJJ (2000) Model . . . 31
  - 3.4.1 Characteristic Function . . . 32
  - 3.4.2 Numerical Results . . . 33
- 3.5 The Double Heston (2009) Model . . . 36
  - 3.5.1 Characteristic Function . . . 37
  - 3.5.2 Numerical Results . . . 38
- 3.6 The Mikhailov and Nögel (2004) Model . . . 41
  - 3.6.1 Characteristic Function . . . 42
  - 3.6.2 Numerical Results . . . 43

4 Greeks and other Sensitivities 47
- 4.1 Introduction . . . 48

4.2 The Heston (1993) Model 48
4.2.1 Greeks and other Sensitivities 49
4.2.2 Numerical Results 50
4.3 The Bates (1996) Model 56
4.3.1 Greeks and other Sensitivities 56
4.3.2 Numerical Results 58
4.4 The SVJJ (2000) Model 63
4.4.1 Greeks and other Sensitivities 63
4.4.2 Numerical Results 65
4.5 The Double Heston (2009) Model 72
4.5.1 Greeks and other Sensitivities 72
4.5.2 Numerical Results 73
4.6 The Mikhailov and Nögel (2004) Model 82
4.6.1 Numerical Results 82

5 Conclusions and Outlook **89**

A Mean Errors and Times for the Heston Model **91**

B Mean Errors and Times for the Bates Model **93**

C Mean Errors and Times for the SVJJ Model **95**

D Mean Errors and Times for the Double Heston Model **97**

E Mean Errors and Times for the Mikhailov and Nögel Model **99**

F Alternative Methodology for Greeks and other sensitivities **101**

G Final Presentation **105**

Bibliography **107**

List of Figures

2.1 Huge differences between $O(N^2)$ and $O(N \log_2 N)$ 15

3.1 Adjustments, errors and CPU times for Fourier Methods in the Heston model 23
3.2 Adjustments, errors and CPU times for Fourier Methods in the Bates model 28
3.3 Adjustments, errors and CPU times for Fourier Methods in the SVJJ model 33
3.4 Adjustments, errors and CPU times for Fourier Methods in the Double Heston model 38
3.5 Adjustments, errors and CPU times for Fourier Methods in the Mikhailov and Nögel model 44

4.1 3D Visualization, adjustments and errors for Heston Delta . . . 51
4.2 3D Visualization, adjustments and errors for Heston Gamma . 52
4.3 3D Visualization, adjustments and errors for Heston Vega 1 . . 53
4.4 3D Visualization, adjustments and errors for Heston Rho . . . 54
4.5 3D Visualization, adjustments and errors for Heston Theta . . 54
4.6 3D Visualization, adjustments and errors for Heston Kappa . . 55
4.7 3D Visualization, adjustments and errors for Heston Sigma . . 55
4.8 3D Visualization, adjustments and errors for Heston Vega 2 . . 56
4.9 3D Visualization, adjustments and errors for Bates Gamma . . 58
4.10 3D Visualization, adjustments and errors for Bates Theta . . . 59
4.11 3D Visualization, adjustments and errors for Bates Delta . . . 60
4.12 3D Visualization, adjustments and errors for Bates Rho 61
4.13 3D Visualization, adjustments and errors for Bates Vega 1 . . . 61
4.14 3D Visualization, adjustments and errors for Bates Kappa . . . 61
4.15 3D Visualization, adjustments and errors for Bates Sigma . . . 62
4.16 3D Visualization, adjustments and errors for Bates Vega 2 . . . 63
4.17 3D Visualization, adjustments and errors for SVJJ Delta 66
4.18 3D Visualization, adjustments and errors for SVJJ Rho 67
4.19 3D Visualization, adjustments and errors for SVJJ Gamma . . 68
4.20 3D Visualization, adjustments and errors for SVJJ Theta . . . 69
4.21 3D Visualization, adjustments and errors for SVJJ Vega 1 . . . 69
4.22 3D Visualization, adjustments and errors for SVJJ Sigma . . . 69
4.23 3D Visualization, adjustments and errors for SVJJ Kappa . . . 71
4.24 3D Visualization, adjustments and errors for SVJJ Vega 2 . . . 71
4.25 3D Visualization, adjustments and errors for Double Heston Theta 74
4.26 3D Visualization, adjustments and errors for Double Heston Vega 11 75

4.27 3D Visualization, adjustments and errors for Double Heston Delta 76
4.28 3D Visualization, adjustments and errors for Double Heston Gamma 77
4.29 3D Visualization, adjustments and errors for Double Heston Rho 77
4.30 3D Visualization, adjustments and errors for Double Heston Vega 12 78
4.31 3D Visualization, adjustments and errors for Double Heston Vega 22 78
4.32 3D Visualization, adjustments and errors for Double Heston Kappa 1 79
4.33 3D Visualization, adjustments and errors for Double Heston Kappa 2 80
4.34 3D Visualization, adjustments and errors for Double Heston Sigma 1 80
4.35 3D Visualization, adjustments and errors for Double Heston Sigma 2 81
4.36 3D Visualization, adjustments and errors for Double Heston Vega 21 81
4.37 CPU times, adjustments and errors for Mikhailov and Nögel Delta 83
4.38 CPU times, adjustments and errors for Mikhailov and Nögel Theta 84
4.39 CPU times, adjustments and errors for Mikhailov and Nögel Gamma 85
4.40 CPU times, adjustments and errors for Mikhailov and Nögel Rho 86
4.41 CPU times, adjustments and errors for Mikhailov and Nögel Vega 1 86
4.42 CPU times, adjustments and errors for Mikhailov and Nögel Sigma 87
4.43 CPU times, adjustments and errors Mikhailov and Nögel Kappa 88
4.44 CPU times, adjustments and errors for Mikhailov and Nögel Vega 2 88

List of Tables

3.1 ITM results for the Heston model. 24
3.2 ATM results for the Heston model. 25
3.3 OTM results for the Heston model. 26
3.4 ITM results for the Bates model. 29
3.5 ATM results for the Bates model. 30
3.6 OTM results for the Bates model. 31
3.7 ITM results for the SVJJ model. 34
3.8 ATM results for the SVJJ model. 35
3.9 OTM results for the SVJJ model. 36
3.10 ITM results for the Double Heston model. 39
3.11 ATM results for the Double Heston model. 40
3.12 OTM results for the Double Heston model. 41
3.13 ITM results for the Mikhailov and Nögel model. 44
3.14 ATM results for the Mikhailov and Nögel model. 45
3.15 OTM results for the Mikhailov and Nögel model. 46

4.1 Results for Heston Delta. 52
4.2 Results for Heston Gamma. 53
4.3 Results for Heston Vega 1. 54
4.4 Results for Bates Gamma. 59
4.5 Results for Bates Theta. 60
4.6 Results for Bates Kappa. 62
4.7 Results for SVJJ Delta. 67
4.8 Results for SVJJ Rho. 68
4.9 Results for SVJJ Sigma. 70
4.10 Results for Double Heston Theta. 75
4.11 Results for Double Heston Vega 11. 76
4.12 Results for Double Heston Vega 22. 79
4.13 Results for Mikhailov and Nögel Delta. 84
4.14 Results for Mikhailov and Nögel Theta. 85
4.15 Results for Mikhailov and Nögel Sigma. 87

Acronyms

ATM At the Money

CF Characteristic Function

DI Direct Integration

EMC Euler Monte-Carlo

FFT Fast Fourier Transform

FRFT Fractional Fast Fourier Transform

ITM In the Money

OTM Out of the Money

PDE Partial Differential Equation

SPDE Stochastic Partial Differential Equation

SPDJE Stochastic Partial Differential Jump Equation

SR Simpson's Rule

SVJJ Stochastic Volatility with double jump

TD Time-Dependent

TDHM Time-Dependent Heston Model

TR Trapezoidal Rule

Resumé:

The goal of this master thesis is to prove the computational efficiency achieved to pricing options through the use of Fourier transform theory, instead of traditional valuation methods, like Monte Carlo or finite differences. Through this master thesis, an European call option shall be considered, for which we know the semi-closed solutions for different models and whose results shall serve us to further check the values obtained by Fourier techniques, finite differences and Monte Carlo.

The master thesis is divided into four chapters that shall seek to provide the necessary information for the proper monitoring of the work. In the first, the basics of Fourier theory shall be presented. Then, in the second chapter, a brief and concise overview about the methods to be used to option pricing shall be made. Shall not be until the third chapter, when we offer the first results obtained using all the above methodology to price an European call option for Heston model and four of its variants, such as: the Heston model considering a jump in the stock equation (Bates model), the Bates model allowing jumps in variance equation (SVJJ model), the double Heston model, which consider two variance equations and finally the Heston model with time dependent parameters. The relevant features of these models are also discussed in this third chapter, previously to the presentation of the results obtained. Finally, in the fourth chapter it shall be showed that it is possible to use these algorithms efficiently for calculating greeks and other sensitivities. This fourth chapter with the third one, are what offer us relevant results regarding the advantages and disadvantages of using the FFT and FRFT algorithms for option pricing and parameter sensitivities and they make up the core of this work.

Due to the completion of this work, it has been checked that algorithms based on Fourier transform are methods more accurate and faster when assessing options compared with use of any method based on Monte Carlo simulation, coming to simultaneously provide prices for 2^{11} strikes for an order of magnitude time similar to a single Monte Carlo simulation, but the problem arises, however, when we try to evaluate exotic options, for which the Fourier methods can be much more difficult to perform and even in the worst case, impossible to implement.

Resumé:

El objetivo de este trabajo de fin de máster, es dejar constancia de la eficiencia computacional lograda al valorar opciones mediante el empleo de la teoría de transformadas de Fourier, con respecto de los métodos de valoración tradicionales, como son Monte Carlo o diferencias finitas. En este trabajo se ha optado por valorar una opción call europea, para la que conocemos soluciones semicerradas para los diferentes modelos que estudiaremos y cuyos resultados nos servirán para comprobar además el grado de ajuste obtenido para cada modelo mediante los métodos de Fourier y Monte Carlo.

El trabajo está dividido en cuatro capítulos en los que se tratará de proporcionar la información necesaria para el correcto seguimiento del trabajo. En el primero de ellos, se expondrán los conceptos básicos sobre la teoría de Fourier. A continuación, en el segundo capitulo, se hará un breve y conciso repaso acerca de los métodos que serán empleados en la valoración de opciones. No será hasta en el capitulo tercero, cuando se ofrecerán los primeros resultados obtenidos empleando toda la metodología anteriormente descrita para valorar una opción call para el modelo de Heston y cuatro de sus variantes: el modelo de Heston considerando un salto en el subyacente (modelo de Bates), el modelo de Bates incluyendo un salto en la parte de la volatilidad, el modelo doble de Heston y finalmente el modelo de Heston dependiente del tiempo. Las características relevantes de todos estos modelos serán también comentadas en este tercer capitulo, previamente a la exposición de los resultados obtenidos. Finalmente, el cuarto capitulo mostrará que también es posible emplear estos algoritmos de forma eficiente para el cálculo de griegas y otras sensibilidades. Este cuarto capitulo junto con el tercero, son los que ofrecen resultados relevantes en cuanto a las ventajas e inconvenientes de emplear los algoritmos FFT y FRFT para la valoración de opciones y cálculo de sensibilidades y forman por tanto, el núcleo del presente trabajo.

Debido a la realización de este trabajo, se ha podido comprobar que la aplicación de algoritmos basados en transformadas de Fourier son una metodología mucho más precisa y rápida a la hora de valorar opciones que cualquier otro método basado en simulaciones de Monte Carlo, llegando a proporcionar de forma simultánea los precios para 2^{11} strikes para un orden de magnitud temporal similar al de una sóla simulación Monte Carlo, aunque el problema surge, sin embargo, cuando tratamos de valorar opciones exóticas, para los cuales la metodología de Fourier puede ser mucho más complicada de aplicar e incluso en el peor de los casos, imposible de implementar.

Preface

This master thesis was submitted to the Faculty of Economic Science, University Complutense of Madrid, as a partial fulfillment of the requirements to obtain the master degree in Quantitative Banking and Finance. The work presented was carried out from February to July in the year 2014, with the collaboration of Prof. Federico Platanía from Department of Quantitative Economics, University Complutense of Madrid and Prof. Manuel Moreno from Department of Economic Analysis, Univerisity of Castilla-La Mancha.

Acknowledgements

First of all, I am grateful to my supervisors in Madrid and Toledo, Federico Platanía and Manuel Moreno, respectively, for supporting and trust in me during all the way of my work to pursuing the master degree. Although circumstances hindered an explicit supervision during most of the work, I am incredibly thankful to them for the time and full confidence in me in these months and for the encouragement to follow my own ideas. I have always been free to pursue the projects that I had in my mind yet still been guided me in the right direction, and I am grateful for that.

Finally, I am forever thankful to my aunt María for make possible that I may have done this master degree. I am equally forever thankful to Miriam for all of her love and understanding and to my parents and sister, for all their support.

Gorka Koldo González

Madrid, Spain, July 2014

Dedicated to María, Miriam and my parents.

1

Overview of Fourier Transform in Finance

This chapter provides a brief, but complete discussion about the concepts of continuous Fourier transforms

Contents

1.1 Introduction **2**
1.2 The Fourier Transform **2**
1.3 Gil-Peláez (1951) Inversion Theorem **3**
1.4 Carr and Madan (1999) Formulation **5**
1.4.1 The Fourier Transform of an Option Price 5
1.4.2 Fourier Transform of Out-of-the-Money Option Prices 8

1.1 Introduction

The outline of this chapter is as follows. At first, we will present some useful results of Fourier analysis and after that Gil-Peláez inversion formula will be presented. These are all the prerequisites needed to face Carr and Madan [CM99] inversion formula for European options, which it will be presented in the last section of this chapter.

1.2 The Fourier Transform

Let W be a random variable defined on some probability space $(\Omega, \mathcal{F}, \mathcal{P})$. The Fourier transform of the continuous function f is defined by

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} e^{i\omega t} f(t)\, dt < \infty \tag{1.1}$$

where $\omega \in \mathbb{R}$. The original f can be recovered as the Fourier transform by inversion of f and for this reason as much f as $\hat{f}$ satisfies the same above conditions

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\omega t} \hat{f}(\omega)\, d\omega < \infty \tag{1.2}$$

The sufficient (but not necessary) condition for the existence of Fourier transform and its inverse is that if $f : \mathbb{R} \to \mathbb{R}$ is in L^1, i.e, the space of integrable functions, then:

$$\int_{-\infty}^{\infty} \mid f(t) \mid \, dt < \infty$$

Characteristic functions (CF) are closely related to Fourier transforms. Then, a characteristic function $\phi(\omega)$, with $\omega \in \mathbb{R}$, is defined as the Fourier transform of the probability density function $\mathbb{P}(x)$

$$\phi(\omega) \equiv \mathcal{F}\left[\mathbb{P}(x)\right] \equiv \int_{-\infty}^{\infty} e^{i\omega x} \mathbb{P}(x)\, dx = E\left[e^{i\omega x}\right] \tag{1.3}$$

Probability density function $\mathbb{P}(x)$ can be obtained by inverse Fourier transform of the characteristic function using the equation (1.2)

$$\mathbb{P}(x) = \mathcal{F}^{-1}\left[\phi(\omega)\right] = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\omega x} \phi(\omega)\, d\omega \tag{1.4}$$

These are the basics concepts about continuous Fourier transforms, so we can already focus on to explain some important applications of these methods to Finance.

1.3 Gil-Peláez (1951) Inversion Theorem

Gil-Peláez [GP51] published his famous inversion formula[1] in 1951. The following proposition states this inversion formula.

Proposition 1. *Gil-Peláez Inversion Formula. Let $F(x)$ be the cumulative distribution function of some variable X. Furthermore, let*

$$\phi(x) = \int_{-\infty}^{\infty} e^{i\omega x}\, dF(x)$$

be the associated characteristic function. Then we have

$$F(x) = \frac{1}{2} - \frac{1}{\pi}\int_0^{\infty} Re\left[\frac{e^{-iux}\phi(u)}{iu}\right] du$$

The proof of this proposition can be found in [GP51] and [Ng05].

Next, let $c(K)$ denotes the price of an European call on a non-dividend paying stock with spot price S_t, strike K and time to maturity $\tau = T - t$. Under the risk neutral measure $\mathbb{Q}$ we have

$$\begin{aligned} c(K) &= e^{-r\tau}\mathbb{E}^{\mathbb{Q}}\left[(S_T - K)^+\right] \\ &= e^{-r\tau}\mathbb{E}^{\mathbb{Q}}\left[(S_T - K)\mathbb{1}_{(S_T>K)}\right] \\ &= e^{-r\tau}\mathbb{E}^{\mathbb{Q}}\left[S_T\mathbb{1}_{(S_T>K)}\right] - Ke^{-r\tau}\mathbb{E}^{\mathbb{Q}}\left[\mathbb{1}_{(S_T>K)}\right] \end{aligned} \tag{1.5}$$

where $\mathbb{1}$ is the indicator function. These probabilities are obtained under different probability measures. We can write

$$\mathbb{E}^{\mathbb{Q}}\left[\mathbb{1}_{(S_T>K)}\right] = \mathbb{Q}(S_T > K) = P_2.$$

On the other hand, evaluating $e^{-r\tau}\mathbb{E}^{\mathbb{Q}}\left[\mathbb{1}_{(S_T>K)}\right]$ requires changing the original measure $\mathbb{Q}$ to another measure $\mathbb{Q}^S$. We employ the Radon-Nykodym derivative

$$\frac{d\mathbb{Q}}{d\mathbb{Q}^S} = \frac{B_T/B_t}{S_T/S_t} = \frac{E^{\mathbb{Q}}[e^{x_T}]}{e^{x_T}} \tag{1.6}$$

[1] This formula was used by Heston in [Hes93] to derive its model.

where we define $B(t)$ to be the value of a bank account at time $t \geq 0$. We assume $B(0) = 1$ and that the bank account evolves according to the following differential equation:

$$dB(t) = rB(t)\,dt, \qquad B(0) = 1$$

where r is the risk-free rate. As a consequence,

$$B_t = exp\left(\int_0^t r\,du\right) = e^{r_t}$$

Then, we can write $e^{-r\tau}\mathbb{E}^{\mathbb{Q}}\left[\mathbb{1}_{(S_T>K)}\right]$ as

$$\begin{aligned} e^{-r\tau}\mathbb{E}^{\mathbb{Q}}\left[\mathbb{1}_{(S_T>K)}\right] &= S_t E^{\mathbb{Q}}\left[\frac{S_T/S_t}{B_T/B_t}\mathbb{1}_{(S_T>K)}\right] \\ &= S_t E^{\mathbb{Q}^S}\left[\frac{S_T/S_t}{B_T/B_t}\mathbb{1}_{(S_T>K)}\frac{d\mathbb{Q}}{d\mathbb{Q}^S}\right] \\ &= S_t E^{\mathbb{Q}^S}\left[\mathbb{1}_{(S_T>K)}\right] \\ &= S_t\mathbb{Q}^S(S_T > K) \\ &= S_t P_1 \end{aligned} \tag{1.7}$$

with these results, European call options prices can be written as

$$c(K) = S_t P_1 - Ke^{-r\tau}P_2 \tag{1.8}$$

The quantities P_1 and P_2 represent the probability of the option expiring in-the-money, conditioned to the value of the stock $S_t = e^{x_t}$, where $x_t = \log S_t$ and on the value v_t of the variance of the stock price at time t. Hence

$$P_1 = \mathbb{Q}^S(S_T > K) \qquad and \qquad P_2 = \mathbb{Q}(S_T > K)$$

Where the measure $\mathbb{Q}$ uses the bank account as numeraire, whereas the measure $\mathbb{Q}^S$ uses the stock price S_t.

The next proposition can be find in [CM99].

Proposition 2. *The probabilities P_j, for $j = 1, 2$, obtained under different measures can be written as*

$$P_j = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\phi\ln k}\phi_j(\varphi; x, v)}{i\varphi}\right]d\varphi \tag{1.9}$$

where $\phi_j(\varphi; x, v)$ represents the characteristics functions ϕ_1 and ϕ_2 for the logarithm of the terminal stock price, $x_T = \ln S_T$.

It makes sense that two CF ϕ_1 and ϕ_2 be associated with the Heston model, due to P_1 and P_2 are obtained under different measures. However, only a single CF ought to exist, because there is only one underlying stock price in the model, so we can write the probabilities P_1 and P_2 in terms of a single CF $\phi(\varphi; x, v)$ as: $\phi_2(\varphi) = \phi(\varphi)$ and $\phi_1(\varphi) = \phi(\varphi - i)/\phi(-i)$.

1.4 Carr and Madan (1999) Formulation

The Fourier technique illustrated in this section was proposed by Carr and Madan [CM99] in (1999). It offers advantages in terms of reduced computation time and an integrand that decays faster than the integrand of the original Heston [Hes93] formulation and shows that Fourier transform of an European option exists once singularities are removed by the inclusion of a damping factor.

1.4.1 The Fourier Transform of an Option Price

Let S_T denote the price at maturity of the underlying asset of an European call with strike K. Define also, $x \equiv \log S_T$, whose associated risk neutral density is given by $q_T(x)$. Then, the Fourier transform of $q_T(x)$, or equivalently the characteristic function of S, can be written as

$$\phi_T(u) = \int_{-\infty}^{\infty} e^{iux} q_T(x)\, dx$$

Now, let $k \equiv \log(K)$, then risk neutral valuation yields

$$\begin{aligned} c_T(K) &= e^{-r\tau}\mathbb{E}\left[(S_T - K)^+\right] \\ &= e^{-r\tau}\mathbb{E}\left[(e^x - e^k)^+\right] \\ &= e^{-r\tau}\int_{-\infty}^{\infty}\left[(e^x - e^k)^+\right] q_T(x)\, dx \\ &= e^{-r\tau}\int_{k}^{\infty}\left[(e^x - e^k)^+\right] q_T(x)\, dx \end{aligned}$$

Since

$$\begin{aligned}\lim_{K\to 0)} c_T(K) &= \lim_{k\to -\infty)} c_T(e^k) \\ &= e^{-r\tau}\int_k^\infty \left[(e^x - e^k)^+\right] q_T(x)\,dx \\ &= e^{-r\tau}\mathbb{E}^{\mathbb{Q}}\left[e^x\right] - 0 \\ &= S_0\end{aligned}$$

we have now that $c_T(e^k)$ does not tend to zero for $k \to -\infty$. Thus $c_T(e^k)$ is not in L^1, the space of integrable functions. For this reason, the Fourier transform will not exist. Carr & Madan [CM99] rectify this by defining the modified call price $\widetilde{c}_T(k)$ as

$$\widetilde{c}_T(k) \equiv e^{\alpha k} c_T(e^k)$$

where $\alpha > 0$. For now, we assume that the Fourier transform of $\widetilde{c}_T(k)$ is well-defined[2], so we have $\widetilde{c}_T(k) \in L^1$.

$$\psi_T(v) \equiv \int_{-\infty}^{\infty} e^{ivk}\widetilde{c}_T(k)\,dk \tag{1.10}$$

Inverting this expression gives

$$\widetilde{c}_T(k) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-ivk}\psi_T(v)\,dv$$

or

$$\begin{aligned}c_T(K) &= \frac{e^{-\alpha\ln(K)}}{2\pi}\int_{-\infty}^{\infty} e^{-iv\ln(K)}\psi_T(v)\,dv \\ &= \frac{e^{-\alpha\ln(K)}}{\pi} Re\left[\int_0^{\infty} e^{-iv\ln(K)}\psi_T(v)\,dv\right]\end{aligned} \tag{1.11}$$

where the last equality follows from the observation that

$$\int_{-\infty}^{\infty} e^{-iv\ln(K)}\psi_T(v)\,dv = \int_0^{\infty} e^{-iv\ln(K)}\psi_T(v)\,dv + \int_{-\infty}^{0} e^{-iv\ln(K)}\psi_T(v)\,dv$$

[2] A complete study of the dampening factor can be found in [LK06]

and where the second term on the right hand side can be rewritten as

$$\begin{aligned}\int_{-\infty}^{0} e^{-iv\ln(K)}\psi_T(v)\,dv &= \int_0^{\infty} e^{iu\ln(K)}\psi_T(-u)\,du \\ &= \int_0^{\infty}\left[e^{-iu\ln(K)}\psi_T(u)\right]^{\dagger} du \\ &= \left[\int_0^{\infty} e^{-iv\ln(K)}\psi_T(v)\right]^{\dagger} dv\end{aligned}$$

yield the claim. Note that we have a nice closed form for the Fourier transform of $\widetilde{c}_T(k)$:

$$\begin{aligned}\psi_T(v) &= \int_{-\infty}^{\infty} e^{ivk}\widetilde{c}_T(k)\,dk \\ &= \int_{-\infty}^{\infty} e^{ivk}e^{\alpha k}c_T(e^k)\,dk \\ &= \int_{-\infty}^{\infty} e^{ivk}e^{\alpha k}\left[e^{-r\tau}\int_k^{\infty}(e^x-e^k)q_T(x)\,dx\right] dk \\ &= e^{-r\tau}\int_{-\infty}^{\infty} q_T(x)\left[\int_{-\infty}^{x} e^{(iv+\alpha)k}(e^x-e^k)\,dk\right] dx \\ &= e^{-r\tau}\int_{-\infty}^{\infty} q_T(x)\left[e^x\int_{-\infty}^{x} e^{(iv+\alpha)k}\,dk - \int_{-\infty}^{x} e^{(iv+\alpha+1)k}\,dk\right] dx \\ &= e^{-r\tau}\int_{-\infty}^{\infty} q_T(x)\left\{e^x\left[\frac{e^{(iv+\alpha)k}}{iv+\alpha}\right]_{-\infty}^{s} - \left[\frac{e^{(iv+\alpha+1)k}}{iv+\alpha+1}\right]_{-\infty}^{s}\right\} dx\end{aligned}$$

since for $\alpha > 0$

$$\lim_{k\to-\infty}\left|e^{(iv+\alpha)k}\right| = \lim_{k\to-\infty}\left|e^{(iv+\alpha+1)k}\right| = \lim_{k\to-\infty} e^{(iv+\alpha)k} = 0$$

the last expression reduces to

$$\begin{aligned}\psi_T(v) &= e^{-r\tau}\int_{-\infty}^{\infty} q_T(x)\left[\frac{e^{(iv+\alpha+1)x}}{iv+\alpha} - \frac{e^{(iv+\alpha+1)x}}{iv+\alpha+1}\right] dx \\ &= e^{-r\tau}\int_{-\infty}^{\infty} q_T(x)\left[\frac{e^{(iv+\alpha+1)x}}{(iv+\alpha)(\alpha+iv+1)}\right] dx\end{aligned}$$

Taking now the Fourier transform for

$$\int_{-\infty}^{\infty} q_T(x)e^{(iv+\alpha+1)x}\,dx = \int_{-\infty}^{\infty} q_T(x)e^{i[v-(\alpha+1)i]x}\,dx$$

we get the characteristic function for the risk neutral price process $\phi_T\,[v-(\alpha+1)i]$.

Finally, we have

$$\psi_T(v)=\frac{e^{-r\tau}\phi_T\,[v-(\alpha+1)i]}{\alpha^2+\alpha-v^2+i(2\alpha+1)v} \tag{1.12}$$

The call price is found through the inverse Fourier transform of $\psi_T(v)$

$$\begin{aligned} c_T(k) &= e^{-\alpha k}\widetilde{c}_T(e^k) \\ &= \frac{e^{-\alpha k}}{2\pi}\int_{-\infty}^{\infty} e^{-ivk}\psi_T(v)\,dv \\ &= \frac{e^{-\alpha k}}{\pi}\int_{0}^{\infty} Re\left[e^{-ivk}\psi_T(v)\right]\,dv \end{aligned} \tag{1.13}$$

1.4.2 Fourier Transform of Out-of-the-Money Option Prices

As it was explained by Carr & Madan [CM99], the equation (1.13) is valid only for pricing ATM and ITM options. However, for very short maturities, the call value approaches its intrinsic value $(S_T-K)^+$, and this forces to the integrand in the Fourier inversion equation (1.13) to be highly oscillatory, and therefore, difficult to integrate numerically. In this section, following the steps given in [CM99], it will be developed an analytic expression in terms of the characteristic function of the ln of the terminal stock price for the Fourier transform of $z_T(k)$, which represents the time T maturity price of OTM call or put option with strike $K=e^k$.

Defining $\zeta_T(v)$ as the Fourier transform of $z_T(k)$

$$\zeta_T(v)=\int_{-\infty}^{\infty} e^{ivk}z_T(k)\,dk \tag{1.14}$$

The prices of out-of-the-money options are obtained by inverting this transform:

$$z_T(k)=\frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-ivk}\zeta_T(v)\,dk \tag{1.15}$$

Assuming that $S_0 = 1$, the price is given by

$$\begin{aligned} z_T(k) = & e^{-rT} \int_{-\infty}^{\infty} \left[\left(e^k - e^x\right) \mathbb{1}_{\{x<k, k<0\}} \right] q_T(x)\, dx \\ & + e^{-rT} \int_{-\infty}^{\infty} \left[\left(e^x - e^k\right) \mathbb{1}_{\{x>k, k>0\}} \right] q_T(x)\, dx \end{aligned} \tag{1.16}$$

Now, as it is described in [PK12], applying the Fourier transform to $z_T(k)$, we obtain

$$\begin{aligned} \zeta_T(v) = & \int_{-\infty}^{\infty} e^{ivk} z_T(k)\, dk \\ = & \int_{-\infty}^{\infty} e^{ivk} e^{-rT} \int_{-\infty}^{\infty} \left[\left(e^k - e^x\right) \mathbb{1}_{\{x<k, k<0\}} \right] q_T(x)\, dxdk \\ & + \int_{-\infty}^{\infty} e^{ivk} e^{-rT} \int_{-\infty}^{\infty} \left[\left(e^x - e^k\right) \mathbb{1}_{\{x>k, k>0\}} \right] q_T(x)\, dxdk \\ = & \int_{-\infty}^{0} e^{ivk} e^{-rT} \int_{-\infty}^{k} \left(e^k - e^x\right) q_T(x)\, dxdk \\ & + \int_{0}^{\infty} e^{ivk} e^{-rT} \int_{k}^{\infty} \left(e^x - e^k\right) q_T(x)\, dxdk \\ = & \int_{-\infty}^{0} e^{-rT} \int_{x}^{0} e^{ivk} \left(e^k - e^x\right) q_T(x)\, dxdk \\ & + \int_{0}^{\infty} e^{-rT} \int_{0}^{x} e^{ivk} \left(e^x - e^k\right) q_T(x)\, dxdk \\ = & e^{-rT} \left[\frac{1}{1+iv} - \frac{e^{rT}}{iv} - \frac{\varphi(v-i)}{v(v-i)} \right] \end{aligned} \tag{1.17}$$

It is important to point out that when $k = 0$ and $T \to 0$, $z_T(k)$ is wide and oscillatory, as can be checked in [CM99]. For this reason it is useful to include a dampening factor[3] and consider the transform of $\sinh(\alpha k) z_T(k)$ as this function vanishes at $k = 0$. Then

$$\begin{aligned} \gamma_T(v) &= \int_{-\infty}^{\infty} e^{ivk} \sinh(\alpha k) z_T(k)\, dk \\ &= \frac{\zeta_T(v - i\alpha) - \zeta_T(v + i\alpha)}{2} \end{aligned} \tag{1.18}$$

and the price of an OTM option is given by

$$z_T(k) = \frac{1}{2\pi \sinh(\alpha k)} \int_0^{\infty} Re\left[e^{-ivk} \gamma_T(v)\right] dv \tag{1.19}$$

[3] A complete study of dampening factor can be found in [LK06]

2

Pricing Methods

This chapter introduces four theoretical pricing methods and shows how they can be applied to option pricing

Contents

2.1 Introduction **12**
2.2 Direct Integration Method **12**
2.3 Euler Monte Carlo Method **13**
2.4 Fast Fourier Transform Method **14**
2.5 Fractional Fast Fourier Transform Method **17**

2.1 Introduction

Many techniques have been suggested to pricing European options under different assumptions of the underlying asset's evolution. For example, one can attempt to find a solution of a pricing partial differential equation (PDE) using numerical methods. One can also resort to Monte Carlo techniques to simulate sample paths of the asset. Averaging a sufficiently large number of realized payoffs then yields the required price. Another methods are based on Fourier analysis, which presents the advantage of pricing options for a huge number of strikes very quickly.

The first method that we will study is based on a direct integration (DI) of the semi-closed formulas for a Call option, once we have solved the model PDE. The second one is based on Euler Monte Carlo simulation scheme (EMC) and the last ones are based in Fourier analysis, being these the Fast Fourier Transform (FFT) and the Fractional Fast Fourier Transform method (FRFT). We will present these last two methods using the adaptive Simpson's and Trapezoidal rules.

2.2 Direct Integration Method

Pricing European options in each one of the following models usually requires the evaluation of an integral, for which we have chosen Gauss-Laguerre quadrature as approximate numerical method, as it is explained in [Rou13]. The goal is to approximate an integral defined on $[a, b]$ as the (weighted) sum of functional values evaluated at several discrete points along the integration domain

$$\int_a^b f(x)\,dx \approx \sum_{j=1}^{N} w_j f(x_j)\,.$$

where the points $(x_1, \ldots, x_N)$ presesent the abscissas and the points $(w_1, \ldots, w_N)$ are the weights.

Gauss-Laguerre quadrature is really relevant for evaluating the integrals for the studied models, because it is designed for integrals over the integration domain $(0, \inf)$. If we consider N points to apply the Gauss-Laguerre quadrature, we

have that abscissas $(x_1, \ldots, x_N)$ are the roots of the Laguerre polynomial $L_N(x)$ of order N, defined as:

$$L_N(x) = \sum_{k=0}^{N} \frac{(-1)^k}{k!} \binom{N}{k} x^k \tag{2.1}$$

where $\binom{N}{k}$ is the binomial coefficient. There are N roots in all and the weights are obtained with the derivative of $L_N(x)$ evaluated at each abscissa

$$L'_N(x_j) = \sum_{k=1}^{N} \frac{(-1)^k}{(k-1)!} \binom{N}{k} x_j^{k-1} \qquad \text{for j = 1,. . . ,N} \tag{2.2}$$

Then, we can define each weight as follows:

$$w_j = \frac{(n!)^2 e^{x_j}}{x_j \left[L'_N(x_j)\right]^2} \qquad \text{for j = 1,. . . ,N}$$

It is important to notice that although the Laguerre polynomial in equation (2.1) has $N + 1$ terms, its derivative (2.2) has N terms.

2.3 Euler Monte Carlo Method

We will examine the stochastic Euler scheme by the simulation of approximating discrete-time trajectories. In addition, general definitions for discrete-time approximations will be given, and the strong and weak convergence criteria for discrete-time approximations introduced. These concepts will all be developed more extensively in the Monte Carlo simulations of the models.

One of the simplest discrete-time approximations of an Ito process is the Euler approximation, or the Euler-Mamyama approximation as it is sometimes called. Explanation of this method can be find in [KP92] and [FR08].

We will consider an Ito process $X = \{X_t, t_0 \leq t \leq T\}$ satisfying the scalar stochastic differential equation

$$dX_t = a(t, X_t)dt + b(t, X_t)dW_t$$

on $t_0 \leq t \leq T$ with the initial value

$$X_{t_0} = X_0.$$

For a given discretization $t_0 = \tau_0 < \tau_1 < \cdots < \tau_n < \cdots < \tau_N = T$ of the time interval $[t_0, T]$, an Euler approximation is a continuous time stochastic process $Y = \{Y(t), t_0 \leq t \leq T\}$ satisfying the iterative scheme

$$Y_{n+1} = Y_n + a(\tau_n, Y_n)(\tau_{n+1} - \tau_n) + b(\tau_n, Y_n)(W_{\tau_{n+1}} - W_{\tau_n}),$$

for $n = 0, 1, 2, \ldots, N-1$ with initial value

$$Y_0 = X_0,$$

where we have written

$$Y_n = Y(\tau_n)$$

for the value of the approximation at the discretization time τ_n. We will also write

$$\Delta_n = \tau_{n+1} - \tau_n$$

for the nth time increment and call

$$\delta = \max_n \Delta_n$$

the maximum time step. For much of this chapter we will consider equidistant discretization times

$$\tau_n = t_0 + n\delta$$

with $\delta = \Delta_n \equiv (T - t_0)/N$ for some integer N large enough so that $\delta \in (0, 1)$.

In this work, we have models driven by a two or three SPDE and for this reason we will consider Cholesky decomposition to enforce correlation between the Brownian motions.

2.4 Fast Fourier Transform Method

Carr & Madan [CM99] in 1999, applied this method to speed up the computation of option prices. In order to illustrate the algorith, it is important to keep in mind that the Discrete Fourier Transform maps a vector of points ($\mathbf{x} = x_1, \ldots, x_N$) to another vector of points ($\hat{\mathbf{x}} = \hat{x}_1, \ldots, \hat{x}_N$) via the relationship

$$\hat{\mathbf{x}} = \sum_{j=1}^{N} e^{-i\frac{2\pi}{N}(j-1)(k-1)} x_j \qquad \text{for k = 1,...,N} \tag{2.3}$$

In DFT we computes these sums independently one of another, hence the number of arithmetic operations is of order N^2, i.e. $O(N^2)$. It was 1965 when Cooley and Tukey [CT65] showed that it was possible to have the DFT evaluated with $O(N \log_2 N)$ arithmetics operations and computed these sums simultaneously.

Figure (2.1) illustrates the huge differences between $O(N^2)$ and $O(N \log_2 N)$

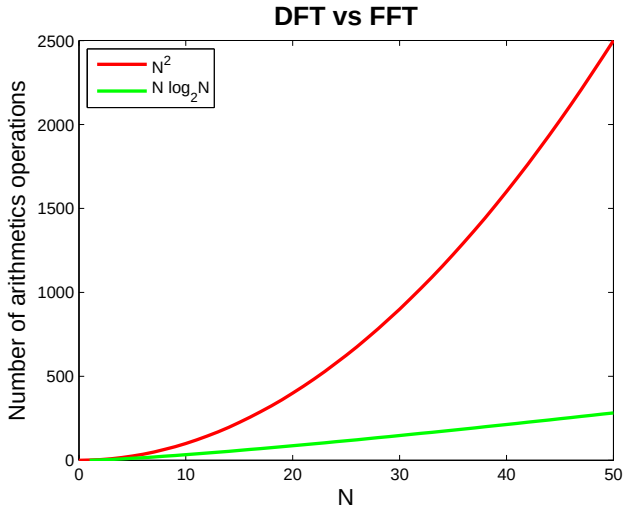


Figure 2.1: Huge differences between $O(N^2)$ and $O(N \log_2 N)$

This method is designed for evaluating integrals approximating them using an integration rule as follows

$$\int_0^\infty e^{-ixu}\psi(u)\,du \approx \sum_{j=0}^{N-1} e^{-ixu_j}\hat{\psi}_j\eta \tag{2.4}$$

Two examples of possible approximations are given by the trapezoidal rule

$$\int_a^b f(x)\,dx \approx \frac{h}{2}f(x_1) + h\sum_{j=2}^{N-1} f(x_j) + \frac{h}{2}f(x_N) \tag{2.5}$$

or by the Simpson's rule

$$\int_a^b f(x)\,dx \approx \frac{h}{3}f(x_1) + \frac{4h}{3}\sum_{j=1}^{N/2-1} f(x_{2j}) + \frac{2h}{3}\sum_{j=1}^{N/2} f(x_{2j-1}) + \frac{h}{3}f(x_N) \tag{2.6}$$

We saw in equation (1.13), that using the Carr and Madan representation, the

call price is given by

$$c_T(k) = \frac{e^{-\alpha k}}{\pi} \int_0^\infty Re \left\{ e^{-ivk} \frac{e^{-r\tau}\phi_T \left[v - (\alpha+1)i\right]}{\alpha^2 + \alpha - v^2 + i(2\alpha+1)v} \right\} dv$$

To implement the FFT algorithm, we must discretize equation (2.4) in strikes and integration domains. Then, as Carr and Madan explain in [CM99], if we approximate the call price by the trapezoidal rule over the truncated domain $[a, b]$ for v and using N discretization points

$$v_j = (j-1)\eta \qquad \text{for j = 1,...,N} \tag{2.7}$$

where $b = N\eta$, being η the increment, then

$$c(k) \approx \frac{\eta e^{-\alpha k}}{\pi} \sum_{j=1}^{N} Re \left[e^{-iv_j k} \psi(v_j) \right] w_j$$

where the weight are determined according the integration rule chosen before. As Carr & Madan point out, we are mainly interested in values c(k), of at-the-money calls, which correspond to k near 0. The FFT returns N values of k and we employ a regular spacing of size λ, so that our values for the strike range, k are given by

$$k_u = -\delta + (u-1)\lambda + \ln S_t \qquad \text{for u = 1,...,N} \tag{2.8}$$

This gives us log strike levels ranging from $\ln S_t - \delta$ to $\ln S_t + \delta - \lambda$, where $\delta = N\lambda/2$. Substituting (2.7) and (2.8) into (2.4), we obtain that the call price is given by

$$c(k_u) \approx \frac{\eta e^{-\alpha k_u}}{\pi} \sum_{j=1}^{N} Re \left[e^{-i\lambda\eta(j-1)(u-1)} e^{i(\delta - \ln S_t)v_j} \psi(v_j) \right] w_j \tag{2.9}$$

To apply the FFT, we note from equation (2.3) that we have the following constraint on the increments η and λ

$$\eta\lambda = \frac{2\pi}{N}$$

being this is an important limitation of the FFT algorithm, since it entails a trade-off between the grid sizes.

2.5 Fractional Fast Fourier Transform Method

The Fractional Fast Fourier Transform (FRFT), was applied in Finance by first time by K. Chourdakis [Cho04] in 2005. Compared with FFT, this method relaxes the constraint $\lambda\eta = 2\pi/N$ on the grid size parameters, so that the term $1/N$ in the exponent of FFT is replaced with a general term β. The FRFT algorithm has the advantage of using the characteristic function information in a more efficient way than the straight FFT. Therefore less function evaluations are typically needed and substantial savings in computational time can be made.

$$\hat{x}_u = \frac{\eta e^{-\alpha k_u}}{\pi} \sum_{j=1}^{N} Re[e^{-i2\pi\beta(j-1)(u-1)} x_j] \qquad \text{for u = 1,. . . ,N} \qquad (2.10)$$

On the other side, the relationship between λ and η becomes $\lambda\eta = 2\pi\beta$. Hence, we can choose the grid size parameters freely, and set

$$\beta = \frac{\lambda\eta}{2\pi}$$

To implement the FRFT on a set of points $(x_1, \ldots, x_N)$, we first define the vectors $\mathbf{y}$ and $\mathbf{z}$, each of dimension 2N.

$$\mathbf{y} = \left(\left[e^{-i\pi(j-1)^2\beta} x_j \right]_{j=1}^{N}, [0]_{j=1}^{N} \right)$$
$$\mathbf{z} = \left(\left[e^{i\pi(j-1)^2\beta} \right]_{j=1}^{N}, \left[e^{i\pi(N-j+1)^2\beta} \right]_{j=1}^{N} \right)$$

The next step is take the FFT of $\mathbf{y}$ and $\mathbf{z}$ to obtain $\hat{\mathbf{y}} = D(\mathbf{y})$ and $\hat{\mathbf{z}} = D(\mathbf{z})$, taking their product element by element, which produces the vector $\hat{\mathbf{h}}$ of dimension 2N defined as:

$$\hat{\mathbf{h}} = \hat{\mathbf{y}} \odot \hat{\mathbf{z}} = \{y_j z_j\}_{j=1}^{2N}$$

Now, take the inverse FFT of $\hat{\mathbf{h}}$ to produce the vector $\mathbf{h} = D^{-1}(\hat{\mathbf{h}})$ of dimension 2N. Finally, multiply element by element the resulting vector with the vector $\mathbf{e}$ defined as

$$\mathbf{e} = \left(\left[e^{-i\pi(k-1)^2\beta} \right]_{k=1}^{N}, [0]_{k=1}^{N} \right)$$

Therefore, we can write the FRFT in compact form as:

$$\begin{aligned}\hat{\mathbf{x}} &= \mathbf{e} \odot D^{-1}(\hat{\mathbf{h}}) = \mathbf{e} \odot D^{-1}(\hat{\mathbf{y}} \odot \hat{\mathbf{z}}) \\ &= \mathbf{e} \odot D^{-1}\left[D(\mathbf{y}) \odot D(\mathbf{z})\right]\end{aligned}$$

We have take only the first N terms of $\hat{\mathbf{x}}$, whereas the next N terms are discarded, as all of them are zeros. If we compare FRFT with FFT, the first method takes the N-vector $\mathbf{x}$ and maps it to the N-vector $\hat{\mathbf{x}}$. However, the FRFT uses the intermediate 2N-vectors $\mathbf{y}$ and $\mathbf{z}$, and requires the computation of two FFTs in the intermediate steps. Nevertheless, the increase in computational time required by the two intermediate FFTs is usually offset by the increase in accuracy due to being able to chosen the strike and integration grid independently and as small as we wish.

3

The Models

This chapter presents the Heston model and four variants, as well as their characteristic functions and numerical options prices

Contents

3.1 Introduction 20
3.2 The Heston (1993) Model 20
3.2.1 Characteristic Function 21
3.2.2 Numerical Results 22
3.3 The Bates (1996) Model 27
3.3.1 Characteristic Function 27
3.3.2 Numerical Results 28
3.4 The SVJJ (2000) Model 31
3.4.1 Characteristic Function 32
3.4.2 Numerical Results 33
3.5 The Double Heston (2009) Model 36
3.5.1 Characteristic Function 37
3.5.2 Numerical Results 38
3.6 The Mikhailov and Nögel (2004) Model 41
3.6.1 Characteristic Function 42
3.6.2 Numerical Results 43

3.1 Introduction

This chapter presents five models, namely, the Heston proposed in [Hes93] (1993) model and four variants of this model that were presented in Bates [Bat96] (1996), the SVJJ model proposed in Duffie et al. (2000), the Double Heston model introduced by Christoffersen et al. [CHJ09] (2009) and a Time-Dependent Heston model proposed by Mikhailov and Nögel [MN04] (2004). Each subsection starts showing the SDPEs that define each model and describes the corresponding parameters. The second part of each section presents the analytical formula of the characteristic function of the corresponding model and applies it to pricing options by different methods, direct integration of semi-closed solution via Gauss-Laguerre quadrature, Fourier algorithms via Simpson's and trapezoidal rules, and Monte Carlo simulation.

Finally, it is important to indicate that for a similar temporal magnitude order, Fourier methods provide at the same time prices for about 2^{11} strikes, while Monte Carlo simulation, provides only a single strike price. Here is the great advantage of Fourier methods over Monte Carlo simulations. So the following analyses represent a valid comparison in the only case that we are interested in knowing the price for a given strike, since otherwise, Fourier methods are much more powerful.

3.2 The Heston (1993) Model

The Heston Model [Hes93] is based on two differential stochastic equations for, respectively, the evolution of the underlying asset price S_t and its variance v_t:

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{v_t} S_t dW_{1,t} \\ dv_t &= \kappa(\theta - v_t)dt + \sigma\sqrt{v_t} dW_{2,t} \\ E^{\mathbb{P}}\left[dW_{1,t} dW_{2,t}\right] &= \rho dt \end{aligned} \tag{3.1}$$

The parameters of the model are:

μ : the drift of the process for the stock
$\kappa > 0$: the mean reversion speed for the variance
$\theta > 0$: the mean reversion level for the variance
$\sigma > 0$: the volatility of the variance
$v_0 > 0$: the initial level of the variance
ρ : the correlation between the two Brownian motions $\mathrm{W}_{1,t}$ and $\mathrm{W}_{2,t}$

We have here a pure diffusion model, which does not allow for jumps in the stock price or the variance processes. Unlike the Black-Scholes model, the volatility in the Heston model is stochastic and follows a mean-reverting square root process, a process originally proposed by Cox, Ingersoll and Ross [CIJR85] to model the spot interest rate.

3.2.1 Characteristic Function

Heston [Hes93] postulates that the characteristic function for the logarithm of the stock price, $x_T = \ln S_T$, has the following log linear form

$$\begin{aligned} \phi_j(\varphi; x_t, v_t) &= \mathbb{E}\left[\phi_j(\varphi; x_T, v_T)|\mathfrak{F}_t\right] \\ &= \mathbb{E}\left[e^{i\varphi \ln S_T}|(x_t, v_t)\right] \\ &= exp\left[C_j(\tau, \varphi) + D_j(\tau, \varphi)v_t + i\varphi x_t\right] \end{aligned} \tag{3.2}$$

where $i = \sqrt{-1}$ and:

$$C_j = ri\varphi\tau + \frac{a}{\sigma^2}\left[(b_j - \rho\sigma i\varphi + d_j)\tau - 2\ln\left(\frac{1 - g_j e^{d_j\tau}}{1 - g_j}\right)\right] \tag{3.3}$$

$$D_j = \frac{b_j - \rho\sigma i\varphi + d_j}{\sigma^2}\left(\frac{1 - e^{d_j\tau}}{1 - g_j e^{d_j\tau}}\right) \tag{3.4}$$

with:

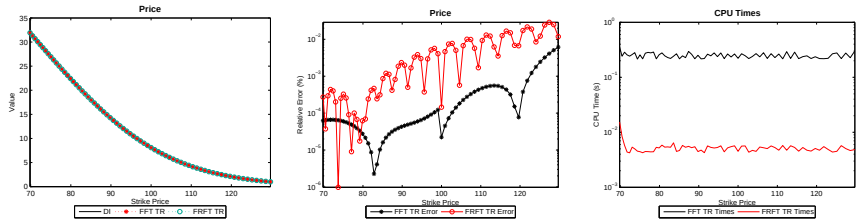
$$
\begin{aligned}
g_j &= \frac{b_j - \rho\sigma i\varphi + d_j}{b_j - \rho\sigma i\varphi - d_j} \\
d_j &= \sqrt{(\rho\sigma i\varphi - b_j)^2 - \sigma^2 (2u_j i\varphi - \varphi^2)} \\
u_j &= \begin{cases} \frac{1}{2}, & \text{if } j = 1; \\ -\frac{1}{2}, & \text{if } j = 2; \end{cases} \\
b_j &= \begin{cases} \kappa + \lambda - \rho\sigma, & \text{if } j = 1; \\ \kappa + \lambda, & \text{if } j = 2; \end{cases} \\
a &= \kappa\theta
\end{aligned}
$$

3.2.2 Numerical Results

We present here the results obtained for the Heston model when we employ the four methods presented before; direct integration of the semi-closed solution by means of Gauss-Laguerre quadrature, Monte Carlo simulation, Fast Fourier Transform and Fractional Fast Fourier Transform. The latter two Fourier methods will be implemented via Trapezoidal and Simpson's rules.

It is important to point out that all the numerical results have been obtained by means of laptop with an Intel Core $i5$ processor of four cores running at 2.27 GHz and with 4.00 GB of RAM memory. Increasing the number of cores and speed of the CPU would probably allow us to get better results.

For an European Call option, we consider a strikes range of $K \in [70, 130]$ and the values $S_0 = 100,\ \kappa = 2,\ \theta = 0.06,\ \sigma = 0.1,\ v_0 = 0.06, \rho = 0.9,\ \tau = 0.5,\ r = 0.05$ and $q = 0$, where q denotes the dividend payment as a continuous yield. Figure (3.1) shows three graphs that include the adjustments comparing the Fourier algorithms with the closed solution, the errors in the previous adjustment and CPU times.



(a) Adjustment **(b)** Errors **(c)** CPU Times

Figure 3.1: Adjustments, errors and CPU times for Fourier Methods in the Heston model

The figure (3.1a)[1] shows that FFT and FRFT are in accordance to the solution provided by DI method for the strikes range, as it can be seen in figure (3.1b). In this last figure we can see that accuracy order for the whole strike range is approximately $4.8 \times 10^{-4}\%$ in arithmetic mean for FFT under trapezoidal rule and approximately $5.4 \times 10^{-3}\%$ in arithmetic mean for FRFT under trapezoidal rule also[2]. The most relevant aspect in these errors are that they show an increasing slope in FFT, which can be explained due to the different algorithm applied in each case depending on we price ITM or OTM options. On the other hand, CPU times are showed in figure (3.1c) where it can be seen that FRFT method is more faster than FFT but losing accuracy, so we necessarily need make a trade-off between accuracy and CPU time.

Before presenting the tables, we need to point out that we have decided to prioritize accuracy rather than CPU time to show how the time that Fourier methods required to reach a fourth order of accuracy in comparison with the CPU time requires by Monte Carlo methods. For these purposes, we have chosen the parameters $\alpha = 1.75$, $N = 2^{11}$ and $uplimit = 700$ for both Fourier methods and the exclusive parameters $\eta = 0.1$, $\lambda = 0.005$ for FRFT whereas, on the other side, all the Monte Carlo simulations have been implemented out with 50 time steps.

The next tables provide the results for an European Call option under the Heston pricing model by applying direct integration of semi-closed solution, Monte Carlo, and FFT and FRFT methods in their two alternative approximations under the following conditions aforementioned. We consider three cases depending on the moneyness of the option.

[1] This graphs only provide the results for the Fourier techniques implemented via trapezoidal rule for reason of readability.

[2] Results under the Simpson's rule are very similar to those obtained with the trapezoidal rule and are not presented for the sake of brevity.

ITM options

We begin with the ITM options.

ITM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	31.9150	0.0000	0.0020
Monte Carlo 10000 paths	32.1107	0.6134	0.0900
Monte Carlo 50000 paths	31.8269	-0.2761	0.2770
Monte Carlo 100000 paths	31.9387	0.0745	0.6320
Monte Carlo 150000 paths	31.9241	0.0287	0.9230
FFT Trapezoidal Rule	31.9150	0.0001	0.2640
FFT Simpson's Rule	31.9150	0.0001	0.2450
FRFT Trapezoidal Rule	31.9149	-0.0003	0.0100
FRFT Simpson's Rule	31.9149	-0.0003	0.0090
$S_0 = 100$, $K = 70.46$, $\kappa = 2$, $\theta = 0.06$, $\sigma = 0.1$, $v_0 = 0.06$, $\rho = 0.9$			

Table 3.1: ITM results for the Heston model.

At first sight, we can see that, with the configuration chosen, Fourier methods are more accurate than either Monte Carlo method and even less time consuming than the simulations with at least $50,000$ paths.

It can be seen that effectively FRFT is the fastest method and almost as accuracy as FFT, being their accuracies of the same order ($\sim 10^{-4}\%$). For this reason, we should be aware of this limitation and make a trade-off between CPU time and accuracy required when we use this method for option pricing.

Furthermore, the FFT is the most accurate method, which is due to the parameters chosen before implementing the model, N and *uplimit* are the same that the FRFT method. Different set ups allow us to control the accuracy and CPU times. However, by far, the FRFT is the most versatile method in the sense that, modifying its parameters, we can achieve a great adjustment for CPU time or the accuracy error.

On the other hand, focusing on Fourier methods, there are not any noticeable differences in accuracy between the trapezoidal or Simpson's rule but, regarding times, we can appreciate that CPU times are higher for the trapezoidal rule than for the Simpson's rule. A possible explanation for this can be related to the code vectorization in MATLAB.

ATM options

The results for ATM options are provided in the next Table.

ATM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	8.0902	0.0000	0.0010
Monte Carlo 10000 paths	8.0589	-0.3866	0.0610
Monte Carlo 50000 paths	8.0247	-0.8090	0.2700
Monte Carlo 100000 paths	8.1616	0.8829	0.5920
Monte Carlo 150000 paths	8.0305	-0.7380	0.9260
FFT Trapezoidal Rule	8.0902	-0.0000	0.3340
FFT Simpson's Rule	8.0902	-0.0001	0.2820
FRFT Trapezoidal Rule	8.0901	-0.0001	0.0050
FRFT Simpson's Rule	8.0901	-0.0001	0.0040
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma = 0.1$, $v_0 = 0.06, \rho = 0.9$			

Table 3.2: ATM results for the Heston model.

In this case, the most relevant result is that the FRFT method almost achieves the same accuracy order than FFT methods, but with a significantly shorter CPU time (up to eight time less). It can be explained due to the special circumstances of this table as it shows the ATM options results and, in this case, the Fourier methods can reach an extraordinary accuracy modifying the parameters. Again, the FRFT shows the best results as it has been implemented with the same parameters N and *uplimit* than the FFT method and has been enhanced with the choice of parameters η and λ.

OTM options

Finally, we present the results for OTM options under this model.

OTM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	0.9904	0.0000	0.0010
Monte Carlo 10000 paths	0.9490	-4.1764	0.0720
Monte Carlo 50000 paths	1.0125	2.2300	0.2420
Monte Carlo 100000 paths	0.9880	-0.2401	0.7640
Monte Carlo 150000 paths	0.9779	-1.2569	0.9270
FFT Trapezoidal Rule	0.9904	0.0061	0.2680
FFT Simpson's Rule	0.9904	0.0058	0.2610
FRFT Trapezoidal Rule	0.9905	0.0118	0.0050
FRFT Simpson's Rule	0.9905	0.0118	0.0050
$S_0 = 100$, $K = 129.73$, $\kappa = 2$, $\theta = 0.06$, $\sigma = 0.1$, $v_0 = 0.06, \rho = 0.9$			

Table 3.3: OTM results for the Heston model.

Here, we can see that the pricing errors for all the methods increase with respect to Table (3.2) and for both trapezoidal or Simpson's rules of integration, being this the most relevant result for OTM options. Needless to say, the FRFT is again the fastest method, by construction of its algorithm.

Until now, we have presented the results for the Heston model, for which we can appreciate some advantages of the Fourier methods for European option pricing, being these the accuracy order reached and the CPU time required to compute option prices. We have also seen that the FFT and FRFT algorithms present a different behavior regarding both aspects, being FFT the most accurate and FRFT the fastest one. For the Heston model, it may seem that these algorithms do not provide a great advantage compared with the Monte Carlo method, but we will see that, for more complicated models as the SVJJ or the double Heston models, FFT and FRFT are a serious alternative to be taken into account.

It is also important to keep in mind that Fourier prices for OTM calls in (3.3) have been calculated by using alternative algorithm proposed by Carr and Madan [CM99] instead of the algorithm applied for ATM or ITM options. For this reason we appreciate some significant increments of accuracy losses in these results.

3.3 The Bates (1996) Model

This model was proposed in [Bat96] and, compared with the Heston (1993) model, it considers jumps that are independently and identically distributed and modeled by a compound Poisson process in the asset price evolution. The model has the following risk-neutral dynamics:

$$\begin{aligned} dS_t &= (r - \Lambda\mu_J)S_t dt + \sqrt{v_t}S_t dW_{1,t} + JS_t d\tilde{N}_t \\ dv_t &= \kappa(\theta - v_t)dt + \sigma_v\sqrt{v_t}dW_{2,t} \\ E^{\mathbb{P}}\left[dW_{1,t}dW_{2,t}\right] &= \rho dt \end{aligned} \tag{3.5}$$

where the new terms included are:

Λ : annual frequency of jumps

J : random percentage jump conditional on a jump ocurring

$\tilde{N}$: Poisson counter with intensity lambda

with

$$1 + J \sim \log N\left(\mu_S, \sigma_S^2\right)$$

and where the relationship between μ_S and μ_J is the following:

$$\mu_J = exp\left(\mu_S + \frac{\sigma_S^2}{2}\right) - 1$$

3.3.1 Characteristic Function

The characteristic function of Bates model [Bat96] has the same appearance as that in the Heston model, with the only difference of a jump part.

$$\begin{aligned} \phi_j(\varphi; x_t, v_t) &= \mathbb{E}\left[\phi_j(\varphi; x_T, v_T)|\mathfrak{F}_t\right] \\ &= \mathbb{E}\left[e^{i\varphi \ln S_T}|(x_t, v_t)\right] \\ &= exp\left[C_j(\tau,\varphi) + D_j(\tau,\varphi)v_t + P(\varphi)\Lambda\tau + i\varphi x_t\right] \end{aligned} \tag{3.6}$$

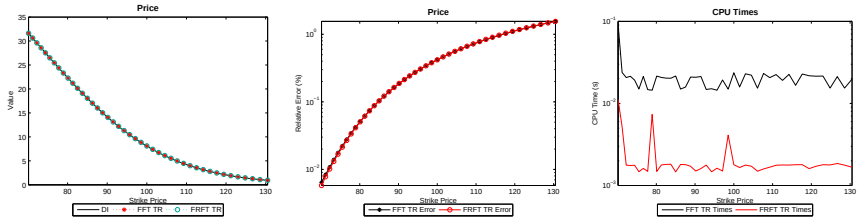
where the functions $C_j(\tau, \varphi)$ and $D_j(\tau, \varphi)$ are the same as for the Heston model. On the other side, $P(\varphi)$ is defined by:

$$P(\varphi) = -\mu_J i\varphi + \left[(1+\mu_J)^{i\varphi}\, e^{\sigma_S^2\left(\frac{i\varphi}{2}\right)(i\varphi-1)} - 1\right] \tag{3.7}$$

3.3.2 Numerical Results

We present now the results for the Bates (1996) model. The four numerical methods mentioned before are used again to price an European call option with the following parameters: $\kappa = 2,\ \theta = 0.06,\ \sigma_V = 0.1,\ v_0 = 0.06,\ \rho = 0.9,\ \tau = 0.5,\ r = 0.05,\ q = 0$. The additional parameters needed to implement this model are $\Lambda = 3,\ \mu_S = -0.05,\ \sigma = 10^{-4}$. The parameters chosen to implement both Fourier methods are $\alpha = 1.75,\ N = 2^9$ and $uplimit = 425$, whereas the FRFT exclusive parameters have been $\eta = 0.1$ and $\lambda = 0.005$. As it can be observed, the values of N and $uplimit$ are smaller than those used to implement the Heston (1993) model. The reason is that it can be checked numerically that any increase of these values does not improve the accuracy and implies a drastic increment of CPU time.

Figure (3.2) shows the adjustment, errors and CPU times.



(a) Adjustment **(b)** Errors **(c)** CPU Times

Figure 3.2: Adjustments, errors and CPU times for Fourier Methods in the Bates model

Figure (3.2a) shows that both Fourier algorithms follow very close the prices provided by the integration of semi-closed solution via the Gauss-Laguerre quadrature, despite of the existence of jumps in this model, although they are small. As can be seen in Figure (3.2b), because of the jumps, the errors in the Bates model are higher than in the Heston model. In this case, we have that the behavior of both methods is very close each other, with a remarkable increased slope along the strikes range. These errors are of the same magnitude order with a mean value of $10^{-4}\%$, which it is more accurate than in the Monte

Carlo method, whit a mean error around 0.4% when we consider $n = 50$ time steps and 1.5×10^6 paths. CPU times are presented in Figure (3.2c) where we can see that the FRFT is more than one magnitude order faster than the FFT algorithm. Looking at Monte Carlo simulations with the fastest simulation (10^4 paths), we note that it is approximately seventeen times slower than the FFT algorithm.

As in the Heston (1993) model, we present now the results for call option prices, distinguishing by its moneyness.

ITM options

Table (3.4) shows the results at maturity for ITM options.

ITM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	31.6284	0.0000	0.0030
Monte Carlo 10000 paths	32.0012	1.1785	0.4300
Monte Carlo 50000 paths	31.6742	0.1447	1.8040
Monte Carlo 100000 paths	31.6597	0.0987	3.8020
Monte Carlo 150000 paths	31.6780	0.1568	5.6010
FFT Trapezoidal Rule	31.6305	0.0064	0.0940
FFT Simpson's Rule	31.5869	-0.1315	0.0290
FRFT Trapezoidal Rule	31.6303	0.0058	0.0090
FRFT Simpson's Rule	31.6303	0.0058	0.0040
$S_0 = 100$, $K = 70.46$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06$, $\rho = 0.9$			
$\Lambda = 3$, $\mu_S = -0.05$, $\sigma_S = 10^{-4}$			

Table 3.4: ITM results for the Bates model.

This table reflects all the issues discussed previously. For example, in general, Monte Carlo simulations are less accurate than any other method based on the Fourier algorithm, except the FFT implemented via Simpson's rule that provides the worst accuracy. Focusing on the FFT technique, a quite counterintuitive result is that the implementation via SR does not fit very accurately the price compared with TR, although it is more than three times faster than the alternative based on TR. The reason is that exists a minimum number of points, N, to calculate the integral via the Simpson's rule to implementing the FFT algorithm correctly, and we have calculated the integral under this limit.

The following tables will not provide this conclusion. In this case, it is clear that for ITM options is more convenient to price using the FRFT instead of

any other method, because the best results are obtained implementing this algorithm. Other relevant aspect is the CPU time, which is abnormally large in relation to its mean value, as can be seen in figure (3.2c).

ATM options

Next table, represents the ATM options case.

ATM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	8.0733	0.0000	0.0010
Monte Carlo 10000 paths	8.1732	1.2375	0.3690
Monte Carlo 50000 paths	8.1906	1.4523	1.7740
Monte Carlo 100000 paths	8.0925	0.2377	3.5890
Monte Carlo 150000 paths	8.0805	0.0892	5.5430
FFT Trapezoidal Rule	8.1073	0.4209	0.0230
FFT Simpson's Rule	8.0640	-0.1161	0.0230
FRFT Trapezoidal Rule	8.1071	0.4186	0.0010
FRFT Simpson's Rule	8.1071	0.4186	0.0020
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06, \rho = 0.9$			
$\Lambda = 3$, $\mu_S = -0.05$, $\sigma_S = 10^{-4}$			

Table 3.5: ATM results for the Bates model.

We can see that the FFT algorithm implemented via SR is more accurate than the alternative based on the TR. We have analyzed again whether Fourier algorithms are faster than the Monte Carlo technique but, now, while Monte Carlo prices present a similar error size to that for ITM options, we find that the price errors from the Fourier algorithms have increased in two magnitude orders, which was observed when discussing Figure (3.2b). Interestingly, CPU times have decreased in a significant amount with a much smaller variance.

OTM options

We finish this section providing the Table for OTM options.

OTM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	0.9268	0.0000	0.0010
Monte Carlo 10000 paths	0.9280	0.1310	0.3760
Monte Carlo 50000 paths	0.9305	0.4008	1.7710
Monte Carlo 100000 paths	0.9501	2.5197	3.5840
Monte Carlo 150000 paths	0.9259	-0.0902	5.4710
FFT Trapezoidal Rule	0.9412	1.5555	0.0220
FFT Simpson's Rule	0.8981	-3.0889	0.0240
FRFT Trapezoidal Rule	0.9412	1.5553	0.0020
FRFT Simpson's Rule	0.9412	1.5553	0.0010
$S_0 = 100$, $K = 129.73$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06$, $\rho = 0.9$			
$\Lambda = 3$, $\mu_S = -0.05$, $\sigma_S = 10^{-4}$			

Table 3.6: OTM results for the Bates model.

As shown in Figure (3.2b), pricing errors from any Fourier algorithm are increasing as the option goes deep OTM. In the extreme situation represented in Table 3.6, the accuracy of the Fourier method corresponds to an error in the units, so it is necessary to adjust the parameters N and *uplimit* to improve the accuracy or if the CPU time is not too important, consider Monte Carlo method as an alternative. In any case, it is worthy to take into account that the FRFT is more flexible to solve this drawback as it incorporates the parameters η and λ jointly with the parameters indicated before.

3.4 The SVJJ (2000) Model

This model was proposed by Duffie et al. [DPS00] in (2000) and extends the Bates (1996) model adding jumps in the variance process. As a result, the model is based on the following risk-neutral dynamics:

$$\begin{aligned} dS_t &= (r - \lambda\mu_J)S_t dt + \sqrt{v_t}S_t dW_{1,t} + JS_t d\tilde{N}_t \\ dv_t &= \kappa(\theta - v_t)dt + \sigma_v\sqrt{v_t}dW_{2,t} + Zd\tilde{N}_t \\ E^{\mathbb{P}}\left[dW_{1,t}dW_{2,t}\right] &= \rho dt \end{aligned} \tag{3.8}$$

where the jump terms are defined as:

$$Z \sim exp(\mu_V)$$
$$(1+J) \sim LogN\left(\mu_S + \rho_J Z, \sigma_S^2\right)$$

with:

$$\mu_J = \frac{exp\left(\mu_S + \frac{\sigma_S^2}{2}\right)}{1 - \rho_J \mu_V}$$

3.4.1 Characteristic Function

The characteristic function of the SVJJ model has the same appearance as that in the Bates (1996) model, but with a more complicated jump part. Poklewski [PK12] provides the following closed-form expression:

$$\begin{aligned}\phi_j(\varphi; x_t, v_t) &= \mathbb{E}\left[\phi_j(\varphi; x_T, v_T)|\mathfrak{F}_t\right] \\ &= \mathbb{E}\left[e^{i\varphi \ln S_T}|(x_t, v_t)\right] \\ &= exp\left[C_j(\tau,\varphi) + D_j(\tau,\varphi)v_t + P_j(\tau,\varphi)\lambda + i\varphi x_t\right] \end{aligned} \tag{3.9}$$

where the functions $\mathrm{C}_j(\tau,\varphi)$ and $\mathrm{D}_j(\tau,\varphi)$ are the same as for the Heston model. On the other side, $\mathrm{P}(\tau,\varphi)$ is defined by:

$$P_j(\tau,\varphi) = -\tau\left(1 + i\varphi\mu_J\right) + exp\left[i\varphi\mu_S + \frac{\sigma_S^2(i\varphi)^2}{2}\right]\nu_j \tag{3.10}$$

where

$$\nu_j = \frac{\beta_j + d_j}{(\beta + d_j)c - 2\mu_V\alpha}\tau + \frac{4\mu_V\alpha}{(d_j c)^2 - (2\mu_V\alpha - \beta_j c)^2}\log\left(\vartheta_j\right)$$
$$c = 1 - i\varphi\rho_J\mu_V$$

and

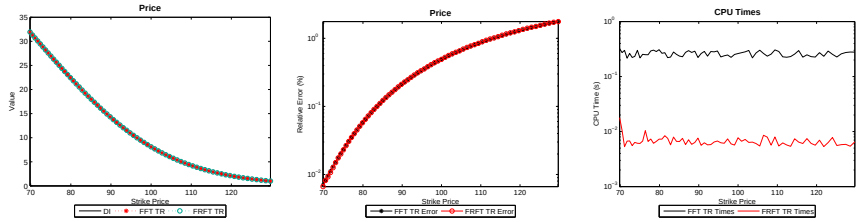
$$\alpha = -\frac{(\varphi^2 + i\varphi)}{2}$$
$$\beta_j = b_j - \rho\sigma_V i\varphi$$
$$\gamma = \frac{\sigma_V^2}{2}$$
$$\vartheta_j = 1 - \frac{(d_j - \beta_j)c + 2\mu_V\alpha}{2d_j c}\left(1 - e^{d_j\tau}\right)$$

where b_j was defined in the Heston model section.

3.4.2 Numerical Results

This Subsection presents the numerical results for the SVJJ model. As this model includes jumps in both SPDEs, it is more difficult to pricing options correctly in this model, so we must consider a lightly jump parameters. As our goal is to price European call options by using the methods considered in the previous sections, we consider the following parameters: $\kappa = 2,\ \theta = 0.06,\ \sigma_V = 0.1,\ v_0 = 0.06,\ \rho = 0.9,\ \Lambda = 3,\ \mu_S = 0.014,\ \sigma_S = 10^{-4},\ \rho_J = -0.4,\ \mu_V = 0.01$ and a strikes range of $K \in [70, 130]$. For both Fourier algorithms, we take $\alpha = 1.75,\ N = 2^{11}$ and $uplimit = 700$, while the FRFT is implemented with its exclusive parameters $\eta = 0.1$ and $\lambda = 0.005$.

Figure (3.3) shows the adjustment, errors and CPU times.



(a) Adjustment **(b)** Errors **(c)** CPU Times

Figure 3.3: Adjustments, errors and CPU times for Fourier Methods in the SVJJ model

Figure (3.3a) represents the adjustment to the exact price of both Fourier methods under the trapezoidal rule[3] to integrate the semi-closed solution via the Gauss-Laguerre quadrature. At first sight, both algorithms seems to fit fairly well the prices, but we need to consider the implementation errors. Figure (3.3b) illustrates that the errors for the SVJJ model show a similar aspect to those in the Bates model as this Figure is graphically identical to Figure (3.2b)). As we will see later, pricing errors for OTM options are greater than those for ITM options. In any case, the errors are around 10^{-2}.

CPU times are different from those in the Bates model although both figures show the same aspect. Now, the mean CPU time is one order higher than

[3]Results from the Fourier methods with the Simpson's rule are very similar, so they are not presented here.

CPU times for the Bates model, an expected result taking into account the complexity of this model. In short, these values for the FFT and for the FRFT are, respectively, of 10^{-1}s and of 10^{-2}s order while they are much higher for the Monte Carlo approach.

As in the previous cases, we present the prices for the different degrees of options moneyness.

ITM options

Table (3.7) summarizes the results for ITM options.

ITM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	31.9142	0.0000	0.0030
Monte Carlo 10000 paths	31.7928	-0.3806	2.2280
Monte Carlo 50000 paths	32.0682	0.4823	10.7900
Monte Carlo 100000 paths	31.9161	0.0058	22.3490
Monte Carlo 150000 paths	32.0178	0.3244	36.3570
FFT Trapezoidal Rule	31.9165	0.0070	0.3440
FFT Simpson's Rule	31.9165	0.0070	0.2380
FRFT Trapezoidal Rule	31.9164	0.0066	0.0170
FRFT Simpson's Rule	31.9164	0.0066	0.0100
$S_0 = 100$, $K = 70.46$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06$, $\rho = 0.9$			
$\Lambda = 3$, $\mu_S = 0.014$, $\sigma_S = 10^{-4}$, $\rho_J = -0.4$, $\mu_V = 0.01$			

Table 3.7: ITM results for the SVJJ model.

This Table shows that CPU times of Monte Carlo methods are much larger than those in Fourier methods and, moreover, they do not guarantee the same grade or accuracy than these algorithms. The last figure shows that both Fourier methods reach the same accuracy order. Clearly, the FRFT is the best alternative to price ITM options in this model.

ATM options

The results for this type of options are summarized in the next Table.

ATM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	8.0706	0.0000	0.0010
Monte Carlo 10000 paths	8.5833	6.3529	2.2600
Monte Carlo 50000 paths	8.4273	4.4208	11.4860
Monte Carlo 100000 paths	8.3981	4.0587	23.2030
Monte Carlo 150000 paths	8.4520	4.7259	33.4450
FFT Trapezoidal Rule	8.1102	0.4910	0.2330
FFT Simpson's Rule	8.1102	0.4909	0.2350
FRFT Trapezoidal Rule	8.1102	0.4909	0.0070
FRFT Simpson's Rule	8.1102	0.4909	0.0060
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06$, $\rho = 0.9$			
$\Lambda = 3$, $\mu_S = 0.014$, $\sigma_S = 10^{-4}$, $\rho_J = -0.4$, $\mu_V = 0.01$			

Table 3.8: ATM results for the SVJJ model.

All the methods provide now much larger errors, being this increase specially large in the Fourier algorithms, where they increase two magnitude orders, whilst the errors in the Monte Carlo method have increased one magnitude order. Now, CPU times for Fourier algorithms are lower than before while, as expected, CPU times for the Monte Carlo are similar to those in the previous Table. Anyway, the FRFT is again the best choice to price European call options under this model.

OTM options

The last case under analysis relates to Deep Out-of-the-Money and is shown in Table (3.9).

OTM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	0.9818	0.0000	0.0010
Monte Carlo 10000 paths	1.0676	8.7409	2.1340
Monte Carlo 50000 paths	1.0943	11.4644	11.1240
Monte Carlo 100000 paths	1.1419	16.3071	22.5510
Monte Carlo 150000 paths	1.1440	16.5278	34.4080
FFT Trapezoidal Rule	0.9990	1.7588	0.2550
FFT Simpson's Rule	0.9990	1.7584	0.2620
FRFT Trapezoidal Rule	0.9991	1.7645	0.0070
FRFT Simpson's Rule	0.9991	1.7645	0.0060
$S_0 = 100$, $K = 129.73$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06, \rho = 0.9$			
$\Lambda = 3$, $\mu_S = 0.014$, $\sigma_S = 10^{-4}$, $\rho_J = -0.4$, $\mu_V = 0.01$			

Table 3.9: OTM results for the SVJJ model.

Corroborating Figure (3.3b), the errors have increased even more in comparison with the last table, but to a lesser extent, being now of one magnitude order. The same is true for the Monte Carlo algorithm. In both cases, CPU times are close to its mean, so one more time, the FRFT is the most accurate and fastest algorithm among all.

3.5 The Double Heston (2009) Model

To match precisely the market implied volatility surface, we can specify a two-factor structure for the volatility instead of a jump component as considered before. This approach was proposed in Christoffersen et al. [CHJ09] leading to a double Heston model. This model considers that the variance of the underlying asset can be split in two components, each following a stochastic process of CIR-type:

For the sake of simplicity, we assume the following correlation structure:

$$\begin{aligned} dS_t &= (r-q)S_t dt + \sqrt{v_{1,t}} S_t dW_{1,t} + \sqrt{v_{2,t}} dW_{2,t} \\ dv_{1,t} &= \kappa_1(\theta_1 - v_{1,t})dt + \sigma_1 \sqrt{v_{1,t}} dZ_{1,t} \\ dv_{2,t} &= \kappa_2(\theta_2 - v_{2,t})dt + \sigma_2 \sqrt{v_{2,t}} dZ_{2,t} \end{aligned} \tag{3.11}$$

For the sake of simplicity, we assume the following stochastic structure:

$$\begin{aligned} \mathbb{E}\left[dW_{1,t} dZ_{1,t}\right] &= \rho_1 dt \\ \mathbb{E}\left[dW_{2,t} dZ_{2,t}\right] &= \rho_2 dt \\ \mathbb{E}\left[dW_{1,t} dW_{2,t}\right] = \mathbb{E}\left[dZ_{1,t} dZ_{2,t}\right] &= \mathbb{E}\left[dW_{1,t} dZ_{2,t}\right] = \mathbb{E}\left[dW_{2,t} dZ_{1,t}\right] = 0. \end{aligned} \tag{3.12}$$

Furthermore, the probabilities P_1 and P_2 for the Double Heston model, obtained under different measures, are different from probabilities given for (1.9), which are valid only for Heston, Bates, SVJJ and Mikhailov and Nögel models. In [Rou13] we can find the following proposition.

Proposition 3. *The probabilities P_1 and P_2 for the Double Heston model can be written as*

$$P_1 = \frac{1}{2} + \frac{1}{\pi} \int_0^\infty Re\left[\frac{e^{-i\varphi \ln k} \phi(\varphi - i; x_t, v_{1t}, v_{2t})}{i\varphi S_t e^{(r-q)\tau}} \right] d\varphi \tag{3.13}$$

$$P_2 = \frac{1}{2} + \frac{1}{\pi} \int_0^\infty Re\left[\frac{e^{-i\varphi \ln k} \phi(\varphi; x_t, v_{1t}, v_{2t})}{i\varphi} \right] d\varphi \tag{3.14}$$

where $\phi(\varphi; x_t, v_{1t}, v_{2t})$ represents the characteristics function for the logarithm of the terminal stock price, $x_T = \ln S_T$.

3.5.1 Characteristic Function

Just as the standard Heston model, the Double Heston model belongs to the larger class of affine models, for which the computation of the characteristic function is rather straightforward. Duffie et al. [DPS00], found that the characteristic function for $\boldsymbol{\varphi} = (\varphi_0, \varphi_1, \varphi_2)$ and $(x_T, v_{1,T}, v_{2,T})$ has the following linear form

$$\begin{aligned} \phi(\boldsymbol{\varphi}; x_t, v_{1,t}, v_{2,t}) &= \mathbb{E}\left[exp(i\varphi_0 x_T + i\varphi_1 v_{1,T} + i\varphi_2 v_{2,T})\right] \\ &= exp\left[A(\tau) + B_0(\tau)x_t + B_1(\tau)v_{1,t} + B_2(\tau)v_{2,t}\right] \end{aligned} \tag{3.15}$$

where

$$A(\tau,\varphi) = (r-q)\varphi i\tau + \sum_{j=1}^{2} \frac{\kappa_j \theta_j}{\sigma_j^2}\left[(\kappa_j - \rho_j\sigma_j\varphi i + d_j)\tau - 2\ln\left(\frac{1-g_j e^{d_j\tau}}{1-g_j}\right)\right]$$

$$B_j(\tau,\varphi) = \frac{\kappa_j - \rho_j\sigma_j\varphi i + d_j}{\sigma_j^2}\left(\frac{1-e^{d_j\tau}}{1-g_j e^{d_j\tau}}\right)$$

and

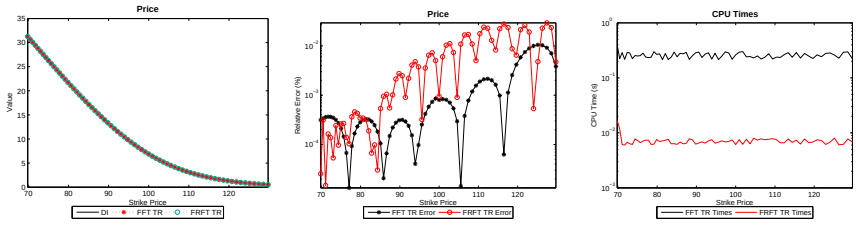
$$g_j = \frac{\kappa_j - \rho_j\sigma_j\varphi i + d_j}{\kappa_j - \rho_j\sigma_j\varphi i - d_j}$$

$$d_j = \sqrt{(\kappa_j - \rho_j\sigma_j\varphi i)^2 + \sigma_j^2\phi\,(\varphi + i)}$$

3.5.2 Numerical Results

We present the numerical results for the Double Heston model. Similarly to the previous models, we price European call options considering a range of strikes in $[70, 130]$ with the following parameters: $S_0 = 100,\ \kappa_1 = 2,\ \theta_1 = 0.005,\ \sigma_1 = 0.2,\ v_{01} = 0.04,\ \rho_1 = 0.6,\ \kappa_2 = 1.5,\ \theta_2 = 0.006,\ \sigma_2 = 0.25,\ v_{02} = 0.03,\ \rho_2 = -0.6,\ \tau = 0.5,\ r = 0.03$ and $q = 0$. Additionally, Fourier algorithms are based on the parameters $\alpha = 1.75,\ N = 2^{11}$ and $uplimit = 700$. Finally, the FRFT is implemented with $\eta = 0.1$ and $\lambda = 0.005$, since this configuration is the most suitable for our purposes.

Figure (3.4) shows the results obtained.



(a) Adjustment **(b)** Errors **(c)** CPU Times

Figure 3.4: Adjustments, errors and CPU times for Fourier Methods in the Double Heston model

Figure (3.4a) illustrates that both Fourier methods implemented via the trapezoidal rule adjust correctly option prices computed by integrating the semi-closed solution via the Gauss-Laguerre quadrature. Figure (3.4b) shows the

pricing errors, with mean values of $1.6 \times 10^{-3}\%$ and $6.8 \times 10^{-3}\%$ for the FFT and FRFT, respectively. For the same values of N and *uplimit*, the FRFT performs worse than the FFT algorithm; then, we should modify these parameters or, alternatively, modify the (FRFT) parameters η and λ, but without taking CPU times away, since although FFT shows lower errors than FRFT, both are of same magnitude order, whereas whereas its mean CPU time are two orders of magnitude less than those for the FRFT times.

ITM options

Table (3.10) shows the results for ITM options.

ITM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	31.8860	0.0000	0.0050
Monte Carlo 5000 paths	32.1318	0.7708	8.7860
Monte Carlo 10000 paths	31.7058	-0.5653	17.6370
Monte Carlo 50000 paths	31.8424	-0.1370	88.3170
Monte Carlo 100000 paths	31.8438	-0.1326	176.2970
FFT Trapezoidal Rule	31.8861	0.0002	0.3740
FFT Simpson's Rule	31.8861	0.0002	0.2880
FRFT Trapezoidal Rule	31.8860	-0.0001	0.0180
FRFT Simpson's Rule	31.8860	-0.0001	0.0100
$S_0 = 100$, $K = 70.46$, $\kappa_1 = 2$, $\theta_1 = 0.005$, $\sigma_1 = 0.2$, $v_{01} = 0.04, \rho_1 = 0.6$			
$\kappa_2 = 1.5$, $\theta_2 = 0.006$, $\sigma_2 = 0.25$, $v_{02} = 0.03, \rho_2 = -0.6$			

Table 3.10: ITM results for the Double Heston model.

Two issues in this Table can be emphasized. First, we consider a smaller number of paths in the Monte Carlo simulations than in the previous models. This choice is motivated as, now, Monte Carlo simulations are very time consuming and considering the same number of paths than before does not make sense as this method never reaches the levels of accuracy and CPU times of the competing methods. Second, CPU times in the Monte Carlo simulation are much higher than those provided by other models and could not be decreased in any of the alternatives under analysis.

We can see that both Fourier algorithms provide a very fine adjustment, whereas the FRFT algorithm is more than thirty times faster than the FFT in some cases. Once again, the Simpson's rule is as accurate as the trapezoidal one but faster that it.

ATM options

Table (3.11) summarizes the results for ATM options.

ATM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	8.0508	0.0000	0.0020
Monte Carlo 5000 paths	7.9245	-1.5691	8.7800
Monte Carlo 10000 paths	7.9322	-1.4736	17.7200
Monte Carlo 50000 paths	7.9636	-1.0841	88.5260
Monte Carlo 100000 paths	7.9988	-0.6464	176.9310
FFT Trapezoidal Rule	8.0508	-0.0004	0.2750
FFT Simpson's Rule	8.0508	-0.0005	0.2390
FRFT Trapezoidal Rule	8.0508	-0.0006	0.0080
FRFT Simpson's Rule	8.0508	-0.0006	0.0070
$S_0 = 100$, $\kappa_1 = 2$, $\theta_1 = 0.005$, $\sigma_1 = 0.2$, $v_{01} = 0.04, \rho_1 = 0.6$			
$\kappa_2 = 1.5$, $\theta_2 = 0.006$, $\sigma_2 = 0.25$, $v_{02} = 0.03, \rho_2 = -0.6$			

Table 3.11: ATM results for the Double Heston model.

The qualitative conclusions equate those for ITM options: the Monte Carlo approach provides larger pricing errors and much higher CPU times than the Fourier methods. Both Fourier methods provide similar errors and the implementation with the Simpson's rule is faster than that with the trapezoidal one.

OTM options

The next Table provides the results for this type of options.

OTM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	0.9553	0.0000	0.0020
Monte Carlo 5000 paths	0.8696	-8.9658	8.8160
Monte Carlo 10000 paths	0.8754	-8.3643	17.6770
Monte Carlo 50000 paths	0.8888	-6.9583	88.2400
Monte Carlo 100000 paths	0.8978	-6.0122	176.0430
FFT Trapezoidal Rule	0.9552	-0.0017	0.2670
FFT Simpson's Rule	0.9552	-0.0021	0.2420
FRFT Trapezoidal Rule	0.9553	0.0039	0.0080
FRFT Simpson's Rule	0.9553	0.0039	0.0080
$S_0 = 100$, $K = 129.73$, $\kappa_1 = 2$, $\theta_1 = 0.005$, $\sigma_1 = 0.2$, $v_{01} = 0.04, \rho_1 = 0.6$			
$\kappa_2 = 1.5$, $\theta_2 = 0.006$, $\sigma_2 = 0.25$, $v_{02} = 0.03, \rho_2 = -0.6$			

Table 3.12: OTM results for the Double Heston model.

In this case, all the methods present a worse adjustment than in the previous moneyness cases. For all the methods, the errors increase one order of magnitude while mean CPU times are practically the same as before. As previously, the Monte Carlo method is the least accurate and with the largest CPU times. The FRFT is the best choice due to its calculation speed, whereas the FFT becomes the most accurate method.

3.6 The Mikhailov and Nögel (2004) Model

Another alternative to adjust the market implied volatilities for short maturities is based on allowing the parameters to be time-dependent, as proposed in Mikhailov and Nögel [MN04]. This time-dependent model has the same appearance as the Heston model presented earlier but with time-dependent parameters in the process of the asset variance and in the correlation between

Brownian motions:

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{v_t} S_t dW_{1,t} \\ dv_t &= \kappa_t(\theta_t - v_t)dt + \sigma_t \sqrt{v_t} dW_{2,t} \\ E^{\mathbb{P}}\left[dW_{1,t} dW_{2,t}\right] &= \rho_t dt \end{aligned} \tag{3.16}$$

3.6.1 Characteristic Function

We start presenting the characteristic function of the time-dependent Heston model. This function is obtained by applying a recursive method, as shown in Rouah [Rou13]. The outline of this method is as follows.

Consider the time interval $[0, T_N]$ and the partition $T_0 = 0 < T_1 < \cdots < T_N < \infty$. The size of this partition is given by the increments $\tau_k = T_k - T_{k-1},\ k = 1, \ldots, N$. We will compute $\widetilde{C}_j(\varphi, \tau_k; \Theta_k)$ and $\widetilde{D}_j(\varphi, \tau_k; \Theta_k)$ recursively, where $\Theta_k = \left[\kappa^{(k-1)}, \theta^{(k-1)}, \sigma^{(k-1)}, v_0^{(k-1)}, \rho^{(k-1)}\right]$ denotes the set of parameter estimates in each stage for $k = 1, \ldots, N$. For the first maturity τ_1, we obtain $\widetilde{C}_j(\varphi, \tau_1; \Theta_1)$ and $\widetilde{D}_j(\varphi, \tau_1; \Theta_1)$ using the initial conditions $C_j^0 = D_j^0 = 0$, exactly as in the Heston model. We then build the characteristic functions, obtain the prices and estimate Θ_1. In the subsequent steps, the estimation is modified since we are using general non-negative values C_j^k and D_j^k. Now, in the second step, replace Θ_1 into the expressions for $\widetilde{C}_j$ and $\widetilde{D}_j$ to produce the second set of initial conditions C_j^1 and D_j^1. Then construct $\widetilde{C}_j(\varphi, \tau_2; \Theta_2)$ and $\widetilde{D}_j(\varphi, \tau_2; \Theta_2)$, obtain the prices and estimates the set Θ_2. And so on.

In summary, the characteristic function is given by the following expressions

$$\phi_j(\varphi; x, v, \Theta_k) = exp\left[\widetilde{C}_j(\varphi, \tau_k; \Theta_k) + \widetilde{D}_j(\varphi, \tau_k; \Theta_k) v_0^k + i\varphi x\right] \tag{3.17}$$

where

$$\begin{aligned} \widetilde{C}_j(\varphi, \tau_k; \Theta_k) &= (r-q)i\varphi\tau_k + \frac{a}{\sigma^2}\left[\varpi_j \tau_k - 2\ln\left(\frac{1 - \widetilde{g}_j e^{d_j \tau_k}}{1 - \widetilde{g}_j}\right)\right] + C_j^{k-1} \\ \widetilde{D}_j(\varphi, \tau_k; \Theta_k) &= \begin{cases} \zeta_j, & \text{if } k = 1; \\ \chi_j, & \text{if } k \geq 2; \end{cases} \end{aligned}$$

with:

$$\begin{aligned}
\varpi_j &= b_j - \rho\sigma i\varphi + d_j \\
\zeta_j &= \frac{b_j - \rho\sigma i\varphi + d_j}{\sigma^2}\left(\frac{1 - e^{d_j\tau}}{1 - g_j e^{d_j\tau}}\right) \\
\chi_j &= \frac{(b_j - \rho\sigma i\varphi + d_j) - (b_j - \rho\sigma i\varphi - d_j)\widetilde{g}_j exp(d_j\tau_k)}{\sigma^2(1 - \widetilde{g}_j exp(d_j\tau_k))}
\end{aligned}$$

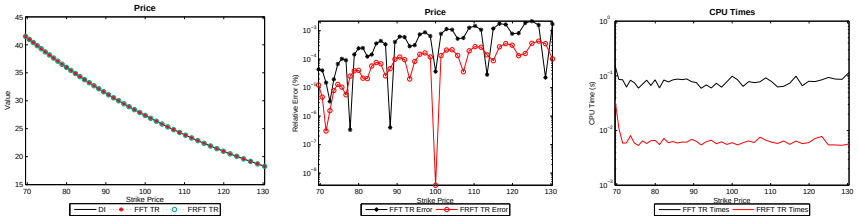
and where

$$\begin{aligned}
g_j &= \frac{b_j - \rho\sigma i\varphi + d_j - D_j^{k-1}\sigma^2}{b_j - \rho\sigma i\varphi + d_j - D_j^{k-1}\sigma^2} \\
d_j &= \sqrt{(\rho\sigma i\varphi - b_j)^2 - \sigma^2\,(2u_j i\varphi - \varphi^2)} \\
u_j &= \begin{cases} \frac{1}{2}, & \text{if } j = 1; \\ -\frac{1}{2}, & \text{if } j = 2; \end{cases} \\
b_j &= \begin{cases} \kappa + \lambda - \rho\sigma, & \text{if } j = 1; \\ \kappa + \lambda, & \text{if } j = 2; \end{cases} \\
a &= \kappa\theta
\end{aligned}$$

3.6.2 Numerical Results

Prices for European call option are computed considering strikes in the interval $[70, 130]$. The remaining parameters are $S_0 = 100,\ \theta = 0.1,\ \sigma = 0.2,\ v_0 = 0.1$, and $\rho = -0.3$. Moreover, $\kappa = 1, 2, 4$ in the three periods, which we assume to have the same length and maturity is $\tau = 5$. Finally, as in Table 1 in Mikhailov and Nögel [MN04], the model is implemented considering $r = 0$.

Both Fourier algorithms have been implemented considering $\alpha = 1.75$ while we consider $N = 2^{10}$ and $uplimit = 550$ for the FFT and $N = 2^9$ and $uplimit = 450$ for the FRFT. Moreover, the FRFT uses $\eta = 0.1$ and $\lambda = 0.005$. As we will see later, this choice allows the FRFT to be much more efficient than the FFT, leading to a decrease in the number of integration points in one magnitude order. Figure (3.5) shows that the numerical results for this model are very accurate.



(a) Adjustment **(b)** Errors **(c)** CPU Times

Figure 3.5: Adjustments, errors and CPU times for Fourier Methods in the Mikhailov and Nögel model

Figure (3.5b) shows that the mean pricing errors in both Fourier methods are around 10^{-4}%, a really small value. It can also be noted that these errors are higher than in the previous ITM and ATM options. The mean CPU times are, respectively, around 10^{-2}s and 10^{3}s for the FFT and FRFT alternatives.

ITM options

Table (3.13) shows the results for ITM options.

ITM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	41.5156	0.0000	0.0170
Monte Carlo 5000 paths	41.8804	0.8786	0.1190
Monte Carlo 10000 paths	40.6857	-1.9991	0.2130
Monte Carlo 50000 paths	41.1471	-0.8878	0.8120
Monte Carlo 100000 paths	41.5893	0.1774	1.8630
FFT Trapezoidal Rule	41.5156	-0.0000	0.1410
FFT Simpson's Rule	41.5144	-0.0029	0.0950
FRFT Trapezoidal Rule	41.5156	-0.0000	0.0140
FRFT Simpson's Rule	41.5156	-0.0000	0.0100
$S_0 = 100$, $K = 70.46$, $\kappa_1 = 2$, $\theta_1 = 0.005$, $\sigma_1 = 0.2$, $v_{01} = 0.04$, $\rho_1 = 0.6$			
$\kappa_2 = 1.5$, $\theta_2 = 0.006$, $\sigma_2 = 0.25$, $v_{02} = 0.03$, $\rho_2 = -0.6$			

Table 3.13: ITM results for the Mikhailov and Nögel model.

We can highlight several interesting results. First, the FRFT offers a price that is accurate up to the fourth decimal, whereas this algorithm is the fastest method and is even faster than the integration of semi-closed solution via the Gauss-Laguerre quadrature. The FFT algorithm is almost as accurate than the FRFT but it much slower, up to ten times in some cases. Monte Carlo

simulations provide accurate results although far from the efficiency achieved by the Fourier algorithms.

ATM options

The next table summarizes the results for ATM options.

ATM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	27.3676	0.0000	0.0140
Monte Carlo 5000 paths	27.7069	1.2399	0.1150
Monte Carlo 10000 paths	27.3054	-0.2271	0.2080
Monte Carlo 50000 paths	27.1933	-0.6369	0.8950
Monte Carlo 100000 paths	27.3169	-0.1852	1.6290
FFT Trapezoidal Rule	27.3676	0.0000	0.0710
FFT Simpson's Rule	27.3664	-0.0043	0.0860
FRFT Trapezoidal Rule	27.3676	0.0000	0.0060
FRFT Simpson's Rule	27.3676	0.0000	0.0060
$S_0 = 100$, $\kappa_1 = 2$, $\theta_1 = 0.005$, $\sigma_1 = 0.2$, $v_{01} = 0.04, \rho_1 = 0.6$			
$\kappa_2 = 1.5$, $\theta_2 = 0.006$, $\sigma_2 = 0.25$, $v_{02} = 0.03, \rho_2 = -0.6$			

Table 3.14: ATM results for the Mikhailov and Nögel model.

The results are similar to the previous ones: the FRFT is the fastest algorithm, including the Gauss-Laguerre quadrature. Once again, the Simpson's rule for the FFT method seems to be less accurate than the other Fourier methods.

OTM options

Finally, table (3.15) shows the results for Deep Out-of-the money options.

OTM			
Method	**Price**	**Error (%)**	**Time (s)**
Closed Form	0.9553	0.0000	0.0020
Monte Carlo 5000 paths	0.8696	-8.9658	8.8160
Monte Carlo 10000 paths	0.8754	-8.3643	17.6770
Monte Carlo 50000 paths	0.8888	-6.9583	88.2400
Monte Carlo 100000 paths	0.8978	-6.0122	176.0430
FFT Trapezoidal Rule	0.9552	-0.0017	0.2670
FFT Simpson's Rule	0.9552	-0.0021	0.2420
FRFT Trapezoidal Rule	0.9553	0.0039	0.0080
FRFT Simpson's Rule	0.9553	0.0039	0.0080
$S_0 = 100$, $K = 129.73$, $\kappa_1 = 2$, $\theta_1 = 0.005$, $\sigma_1 = 0.2$, $v_{01} = 0.04, \rho_1 = 0.6$			
$\kappa_2 = 1.5$, $\theta_2 = 0.006$, $\sigma_2 = 0.25$, $v_{02} = 0.03, \rho_2 = -0.6$			

Table 3.15: OTM results for the Mikhailov and Nögel model.

Now, the FRFT is four times slower than the Gauss-Laguerre quadrature. Moreover, for all the methods, the errors have increased with respect to ATM options. The worst case corresponds to the Monte Carlo method with errors of 10% order, whereas the accuracy of the Fourier methods is around 10^{-3}%. Note that the FFT via the Simpon's rule is even more accurate than any FRFT algorithm.

4

Greeks and other Sensitivities

This chapter computes Greeks and other sensitivities for all the option pricing models presented in the previous chapters

Contents

4.1 Introduction 48
4.2 The Heston (1993) Model 48
4.2.1 Greeks and other Sensitivities 49
4.2.2 Numerical Results 50
4.3 The Bates (1996) Model 56
4.3.1 Greeks and other Sensitivities 56
4.3.2 Numerical Results 58
4.4 The SVJJ (2000) Model 63
4.4.1 Greeks and other Sensitivities 63
4.4.2 Numerical Results 65
4.5 The Double Heston (2009) Model 72
4.5.1 Greeks and other Sensitivities 72
4.5.2 Numerical Results 73
4.6 The Mikhailov and Nögel (2004) Model 82
4.6.1 Numerical Results 82

4.1 Introduction

In this chapter, Greeks and other sensitivities will be computed under the five option pricing models presented previously. These measures will be computed numerically by four alternative techniques: direct integration through Gauss-Laguerre quadrature, finite differences approximation and by FFT and FRT methods. Our main goal is to show that both Fourier methods are valid in this context and analyze their efficiency with respect to the elapsed time involved in the calculation.

Through this chapter, it is important to keep in mind that, under the models presented earlier, the prices of European calls admit a semi-closed form expression and, hence, it is also possible to obtain analytical formulas for the Greeks. Recall that the call price is given as

$$c(K) = S_t e^{-q\tau} P_1 - K e^{-r\tau} P_2 \tag{4.1}$$

where P_1 and P_2 are the probabilities calculated in the previous chapters.

4.2 The Heston (1993) Model

In the first part of this section, Greeks for the Heston model will be showed in their analytic formula. Other sensitivities as regards as parameters like kappa, sigma and theta are also having into account, but only the sensitivity as regards as theta will be calculated. The remaining sensitivities have been calculated symbolically by means of MATLAB.

The goal of the second part of this section is to compare the results obtained when we compute the Greeks and other sensitivities for the Heston model by means of methods as finite differences or Fourier algorithms. We will compare them in relation with the results provided by Direct Integration of semi-closed solution via Gauss-Laguerre quadrature.

4.2.1 Greeks and other Sensitivities

Differentiating equation (4.1) and considering equation (1.9), Delta, Gamma, Rho and Theta are given respectively by

$$\Delta_H = \frac{\partial c}{\partial S} = e^{-q\tau} P_1 \tag{4.2}$$

$$\Gamma_H = \frac{\partial^2 c}{\partial S^2} = e^{-q\tau}\frac{\partial P_1}{\partial S} = \frac{e^{-q\tau}}{\pi S_t}\int_0^\infty Re\left[e^{-i\varphi \ln K}\phi_1(\varphi; x_t, v_t)\right]\, d\varphi \tag{4.3}$$

$$\rho_H = \frac{\partial^2 c}{\partial r} = K\tau e^{-r\tau} P_2 \tag{4.4}$$

$$\Theta_H = -\frac{\partial c}{\partial \tau} = -S_t e^{-q\tau}\left(-qP_1 + \frac{\partial P_1}{\partial \tau}\right) + Ke^{-r\tau}\left(-rP_2 + \frac{\partial P_2}{\partial \tau}\right) \tag{4.5}$$

Being

$$\frac{\partial P_j}{\partial \tau} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{\partial \phi_j}{\partial \tau} \times \frac{e^{-i\varphi \ln K}}{i\varphi}\right] d\varphi$$

where

$$\frac{\partial \phi_j}{\partial \tau} = exp(C_j + D_j v_t + i\varphi x_t)\left(\frac{\partial C_j}{\partial \tau} + \frac{\partial D_j}{\partial \tau} v_t\right)$$

and

$$\frac{\partial C_j}{\partial \tau} = (r-q)\varphi i + \frac{\kappa\theta}{\sigma^2}\left[b_j - \rho\sigma\varphi i + d_j + \frac{2g_j d_j e^{d_j\tau}}{1 - g_j e^{d_j\tau}}\right]$$

$$\frac{\partial D_j}{\partial \tau} = \frac{b_j - \rho\sigma\varphi i + d_j}{\sigma^2}\left[\frac{(g_j - 1)d_j e^{d_j\tau}}{(1 - g_j e^{d_j\tau})^2}\right]$$

On the other hand, taking into account the two measures for the variance explained by Zhu in [Zhu09], we have two Vegas, one based on $v = \sqrt{v_0}$ and the other based on $\omega = \sqrt{\theta}$. With these definitions, we have that the Vegas are

$$\mathcal{V}_{1H} = \frac{\partial c}{\partial v} = \frac{\partial c}{\partial v_0} 2\sqrt{v_0}$$

$$\mathcal{V}_{2H} = \frac{\partial c}{\partial \omega} = \frac{\partial c}{\partial \theta} 2\sqrt{\theta}$$

Being the first Vega

$$\mathcal{V}_{1H} = Se^{-q\tau}\frac{\partial P_1}{\partial v_0} 2\sqrt{v_0} - Ke^{-r\tau}\frac{\partial P_2}{\partial v_0} 2\sqrt{v_0} \tag{4.6}$$

where

$$\frac{\partial P_j}{\partial v_0} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\varphi \ln K}\phi_j(\varphi; x_t, v_t)D_j(\tau,\varphi)}{i\varphi}\right] d\varphi.$$

and being the second Vega

$$\mathcal{V}_{2H} = Se^{-q\tau}\frac{\partial P_1}{\partial \theta}2\sqrt{\theta} - Ke^{-r\tau}\frac{\partial P_2}{\partial \theta}2\sqrt{\theta} \tag{4.7}$$

where

$$\frac{\partial P_j}{\partial \theta} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\varphi \ln K}\phi_j(\varphi; x_t, v_t)\partial C_j/\partial\theta)}{i\varphi}\right] d\varphi.$$

and

$$\frac{\partial C_j}{\partial \theta} = \frac{\kappa}{\sigma^2}\left[(b_j - \rho\sigma\varphi i + d_j)\tau - 2\ln\left(\frac{1-g_j e^{d_j\tau}}{1-g_j}\right)\right]$$

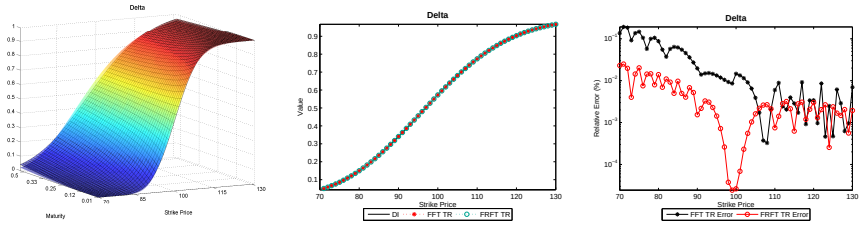
4.2.2 Numerical Results

The goal of this section is to compare the results obtained when we compute the Greeks and other sensitivities for the Heston model by means of methods as finite differences and the Fourier methods. We will compare them in relation with the result provided by Direct Integration of semi-closed solution. In this case, we will see as the Fourier methods continues offering a reduced computation times, although they are not so small as the traditional methods when we compute the most simplest Greeks. They offer also an accurate results.

All the graphics and results that we will present, corresponding to an European call option with the following conditions: $K = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma = 0.1$, $v_0 = 0.06, \rho = 0.9$ $\tau = 0.5$ [1], $r = 0.05$, $q = 0$ and for a spot range of $S_0 \in [70, 130]$. Furthermore, we implemented the Fourier algorithms with $\alpha = 1.75$, $N = 2^9$ integration points and an upper integration limit of $uplimit = 600$ for FFT, while we have chosen $N = 2^9$, $uplimit = 100$, $\eta = 0.1$ and $\lambda = 0.005$ for FRFT. We have chosen these values to show one more time, the flexibility of FRFT method in comparison with FFT and we will see that FRFT will be faster and more accurate than FFT only with a fine choice of η and λ. We want to point out, that all the tridimensional figures showed in this chapter, contain two representation of the Greek or sensibility, corresponding each of them to the cases in which correlation is $\rho = 0.9$ or $\rho = -0.9$.

[1] For 3D graphics, the maturity range is $\tau \in [0.01, 0.5]$, while the value of $\tau = 0.5$ refers only to bidimensional cuts

We begin show the results for Delta Greek, which it can be seen in figure (4.1).



(a) Delta **(b)** Adjust **(c)** Errors

Figure 4.1: 3D Visualization, adjustments and errors for Heston Delta

At first sight, we can see in figure (4.1b), that the adjustment for Fourier methods via trapezoidal integration rule is near close to direct integration of semi-closed solution. but if we see in deep, we can see in figure (4.1c), that the error of the adjustment of both Fourier methods via trapezoidal rule is in arithmetic mean 0.74% for FFT and 4.80×10^{-3}% for FRFT, which it is really a great adjustment in both cases. Times will not be presented here graphically, but it will be discussed in the tables later.

In the next paragraphs, we will comment only three tables relatives to two Greeks and the sensitivity of another parameter of the model, instead of commenting all the tables for all the Greeks and other parameters for reasons of space and readability of the document. Furthermore, all results showed in this chapter corresponding for ATM options, because it has been proved that the other cases do not provide any relevant result.

In the same way in which results for option pricing were presented in chapter 3, we show here the results for Greeks and other sensitivities under the model conditions indicated before. Again, numerical results are obtained with same laptop as we indicated previously.

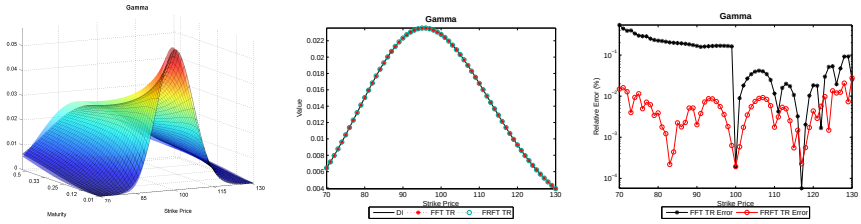
Therefore, we begin with the results for Delta Greek and show them in the next table

Heston Delta			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	0.5726	0.0000	0.0010
Finite Differences	0.5726	0.0102	0.0020
FFT Trapezoidal Rule	0.5727	0.0017	0.0160
FFT Simpson's Rule	0.5727	0.0017	0.0220
FRFT Trapezoidal Rule	0.5726	-0.0000	0.0010
FRFT Simpson's Rule	0.5726	-0.0000	0.0020
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma = 0.1$, $v_0 = 0.06, \rho = 0.9$			

Table 4.1: Results for Heston Delta.

FFT and FRFT algorithms have been executed considering the same integration grid with N^9 points, but with different *uplimit* and it is this, together with the choice of η and λ, which give us a negligible error in the results obtained with FRFT method in comparison with FFT algorithm. Other relevant aspect in table (4.1) is that Trapezoidal rule is faster but equally accurate than Simpson's rule for Fourier algorithms. The speed in the calculation is a reasonable result, because Simpson's rule requires a greater number of allocation of weights. However, for FFT case, it is important notice that the error in accuracy, even though it may seem significant, is an error in the third decimal. Throughout FRFT algorithm, we get a great accurate order, which is greater than Finite Differences methods and less CPU time consuming.

Next, figure (4.2) shows the results for Gamma Greek.



(a) Gamma **(b)** Adjust **(c)** Errors

Figure 4.2: 3D Visualization, adjustments and errors for Heston Gamma

Again, we can see in figure (4.2c), that the adjustment for Fourier methods is in arithmetic mean, around 0.13% for FFT and 5.80×10^{-3}% for FRFT, which are of the same order than the means for Delta Greek and equally both

algorithms offer us an higher reliability. Figures (4.2a) and (4.2b) show the appearance of Gamma Greek.

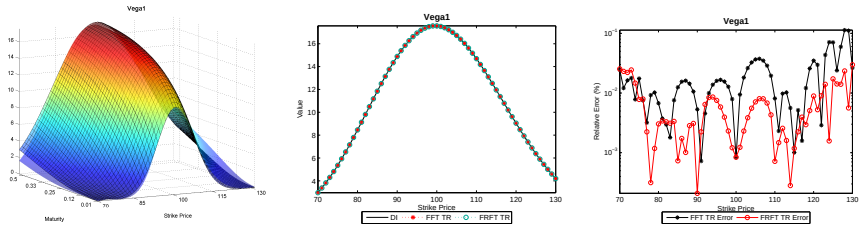
We will analyze now the table of results for ATM options.

Heston Gamma			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	0.0228	0.0000	0.0010
Finite Differences	0.0228	0.0244	0.0030
FFT Trapezoidal Rule	0.0228	-0.0002	0.0220
FFT Simpson's Rule	0.0228	-0.0002	0.0150
FRFT Trapezoidal Rule	0.0228	-0.0002	0.0010
FRFT Simpson's Rule	0.0228	-0.0002	0.0020
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma = 0.1$, $v_0 = 0.06$, $\rho = 0.9$			

Table 4.2: Results for Heston Gamma.

In this table, we can se that results showed are very similar with those of table (4.1) and anew, they are much better in accuracy order than FD method and in the FRFT case, even thought more faster than it.

We continue with other important Greek, Vega 1, for which we present the results obtained in figure (4.3)



(a) Vega 1 **(b)** Adjust **(c)** Errors

Figure 4.3: 3D Visualization, adjustments and errors for Heston Vega 1

Here, we can see in figure (4.3c), that the adjustment for Fourier methods is in arithmetic mean, around 0.02% for FFT and 4.70×10^{-3}% for FRFT, which offer us an higher reliability. Figures (4.3a) and (4.3b) show the appearance of Vega 1 Greek.

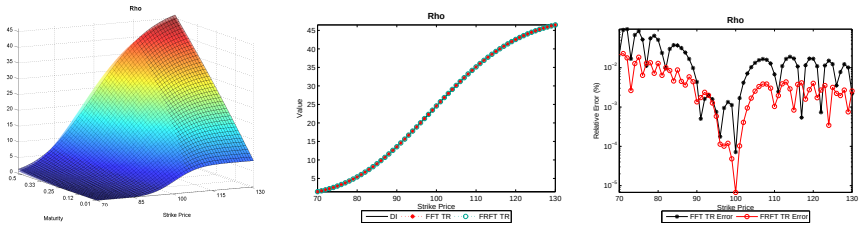
The results for ATM options are showed in the next table

Heston Vega 1			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	17.5660	0.0000	0.0030
Finite Differences	17.5660	-0.0000	0.0020
FFT Trapezoidal Rule	17.5659	-0.0008	0.0210
FFT Simpson's Rule	17.5659	-0.0008	0.0190
FRFT Trapezoidal Rule	17.5659	-0.0008	0.0010
FRFT Simpson's Rule	17.5659	-0.0008	0.0010
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma = 0.1$, $v_0 = 0.06, \rho = 0.9$			

Table 4.3: Results for Heston Vega 1.

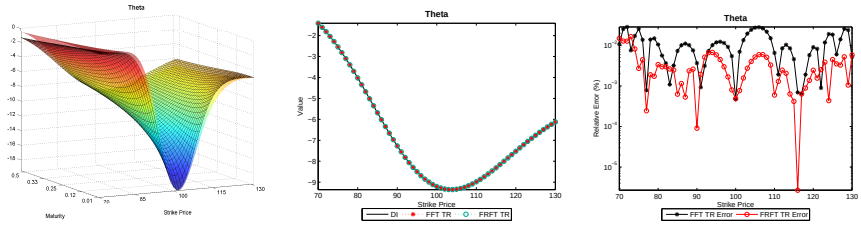
In this table, there are not any remarkable result, being as we can observed here the same situations commented before.

Here, we present here the figures for remaining Greeks Rho and Theta, which no present any relevant aspect.



(a) Rho **(b)** Adjust **(c)** Errors

Figure 4.4: 3D Visualization, adjustments and errors for Heston Rho

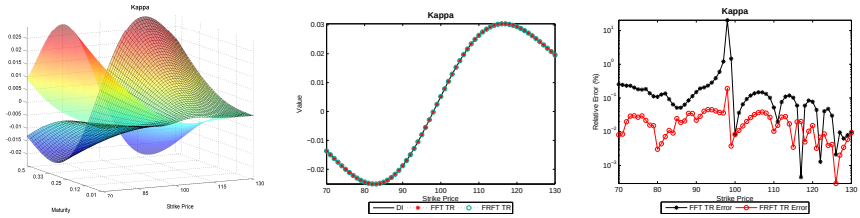


(a) Theta **(b)** Adjust **(c)** Errors

Figure 4.5: 3D Visualization, adjustments and errors for Heston Theta

To finish this section, we present here the figures for parameters kappa, sigma

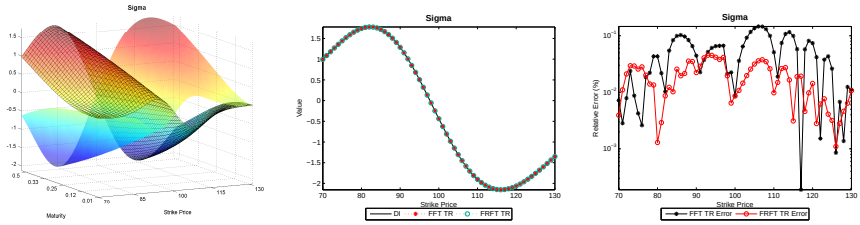
and Vega 2, whose more relevant aspects will be commented in briefly. Kappa will be the first of them



(a) Kappa **(b)** Adjust **(c)** Errors

Figure 4.6: 3D Visualization, adjustments and errors for Heston Kappa

As it can be seen in figure (4.6c), the adjustment error for Kappa is greater than the other Greeks showed before. It can be explained due to the derivatives for Kappa have been calculated symbolically throughout MATLAB due to its complexity, and for this reason the results showed are less accurate and more CPU time consuming in Fourier cases.

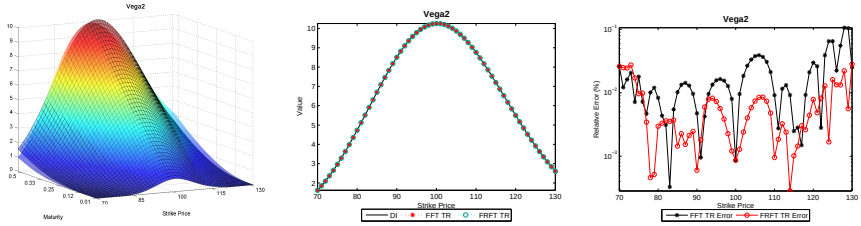


(a) Sigma **(b)** Adjust **(c)** Errors

Figure 4.7: 3D Visualization, adjustments and errors for Heston Sigma

From the same form as we indicated in the last paragraph for Kappa, Sigma has been evaluated symbolically and presents the same inconveniences that we commented previously.

Last, Vega 2 has been calculated in closed form and it does not present the problems previously indicated, as you can see in figure (4.8)



(a) Vega 2 **(b)** Adjust **(c)** Errors

Figure 4.8: 3D Visualization, adjustments and errors for Heston Vega 2

4.3 The Bates (1996) Model

In the first part of this section, Greeks for the Bates model will be derived in their analytic formula. Other sensitivities as regards as parameters like kappa, sigma and theta are also having into account, but only the sensitivity as regards as theta will be derived. The remaining sensitivities have been calculated symbolically by means of MATLAB.

The goal of the second part of this section is to compare the results obtained when we compute the Greeks and other sensitivities for the Bates model by means of methods as finite differences or Fourier algorithms. We will compare them in relation with the results provided by Direct Integration of semi-closed solution via Gauss-Laguerre quadrature.

4.3.1 Greeks and other Sensitivities

Differentiating equation (4.1) and considering equation (1.9), we have that Delta, Gamma, Rho and Vegas are the same that obtained before for the Heston model. However, Theta acquires a different shape, as we will see in the thorough this section.

$$\Delta_B = \frac{\partial c}{\partial S} = e^{-q\tau} P_1 \tag{4.8}$$

$$\Gamma_B = \frac{\partial^2 c}{\partial S^2} = e^{-q\tau} \frac{\partial P_1}{\partial S} = \frac{e^{-q\tau}}{\pi S_t} \int_0^\infty Re\left[e^{-i\varphi \ln K} \phi_1 1(\varphi; x_t, v_t)\right] d\varphi \tag{4.9}$$

$$\rho_B = \frac{\partial^2 c}{\partial r} = K\tau e^{-r\tau} P_2 \tag{4.10}$$

while Theta is given by

$$\Theta_B = -\frac{\partial c}{\partial \tau} = -S_t e^{-q\tau}\left(-qP_1 + \frac{\partial P_1}{\partial \tau}\right) + Ke^{-r\tau}\left(-rP_2 + \frac{\partial P_2}{\partial \tau}\right) \quad (4.11)$$

Being

$$\frac{\partial P_j}{\partial \tau} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{\partial \phi_j}{\partial \tau} \times \frac{e^{-i\varphi \ln K}}{i\varphi}\right] d\varphi$$

where

$$\frac{\partial \phi_j}{\partial \tau} = exp(C_j + D_j v_t + P(\varphi)\Lambda\tau + i\varphi x_t)\left(\frac{\partial C_j}{\partial \tau} + \frac{\partial D_j}{\partial \tau} v_t + P(\varphi)\lambda\right) \quad (4.12)$$

and

$$\frac{\partial C_j}{\partial \tau} = (r-q)\varphi i + \frac{\kappa\theta}{\sigma^2}\left[b_j - \rho\sigma\varphi i + d_j + \frac{2g_j d_j e^{d_j\tau}}{1-g_j e^{d_j\tau}}\right]$$
$$\frac{\partial D_j}{\partial \tau} = \frac{b_j - \rho\sigma\varphi i + d_j}{\sigma^2}\left[\frac{(g_j-1)d_j e^{d_j\tau}}{(1-g_j e^{d_j\tau})^2}\right]$$

Finally, Vegas are given by

$$\mathcal{V}_{1B} = \frac{\partial c}{\partial v} = \frac{\partial c}{\partial v_0} 2\sqrt{v_0}$$
$$\mathcal{V}_{2B} = \frac{\partial c}{\partial \omega} = \frac{\partial c}{\partial \theta} 2\sqrt{\theta}$$

Being the first Vega

$$\mathcal{V}_{1B} = Se^{-q\tau}\frac{\partial P_1}{\partial v_0} 2\sqrt{v_0} - Ke^{-r\tau}\frac{\partial P_2}{\partial v_0} 2\sqrt{v_0} \quad (4.13)$$

where

$$\frac{\partial P_j}{\partial v_0} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\varphi \ln K}\phi_j(\varphi; x_t, v_t) D_j(\tau, \varphi)}{i\varphi}\right] d\varphi.$$

and being the second Vega

$$\mathcal{V}_{2B} = Se^{-q\tau}\frac{\partial P_1}{\partial \theta} 2\sqrt{\theta} - Ke^{-r\tau}\frac{\partial P_2}{\partial \theta} 2\sqrt{\theta} \quad (4.14)$$

where

$$\frac{\partial P_j}{\partial \theta} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\varphi \ln K}\phi_j(\varphi; x_t, v_t)\partial C_j/\partial\theta)}{i\varphi}\right] d\varphi.$$

and

$$\frac{\partial C_j}{\partial \theta} = \frac{\kappa}{\sigma^2}\left[(b_j - \rho\sigma\varphi i + d_j)\tau - 2\ln\left(\frac{1-g_j e^{d_j\tau}}{1-g_j}\right)\right]$$

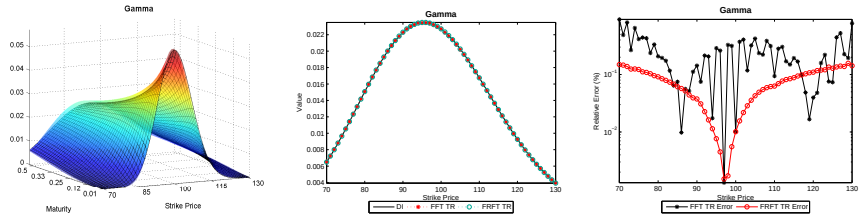
4.3.2 Numerical Results

As we did before for the Heston model, we present here the numerical results for Greeks and other sensitivities for the Bates model employing the same methods. The peculiarity of this model, is that it includes jumps in the stock price and for this reason the results will not be so accurate as what we obtained for the previous model, unless we consider a restrained size of jumps and it is what we did.

In this case, all the graphics and results showed in this section, corresponding to an European call option with the following conditions: $K = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06$, $\rho = 0.9$, $\tau = 0.5$, $r = 0.05$, $q = 0$ and the new parameters needed to implement this model; $\Lambda = 3$, $\mu_S = -0.05$, $\sigma = 10^{-4}$. We consider again, the same spot range of $S_0 \in [70, 130]$. In relation to Fourier algorithms, we have implemented them considering $\alpha = 1.75$, a grid of $N = 2^9$ integration points an upper limit of integration of $uplimit = 200$ for both Fourier methods, while we implemented FRFT with a fine adjustment of its parameters, taking $\eta = 0.1$ and $\lambda = 0.005$.

As novelty for this model, we show the table of results for Delta, Theta and Kappa only, whereas the remainder Greeks and other sensitivities will be showed graphically.

We begin this section, showed the results for Gamma Greek, which it is presented in figure (4.9)



(a) Gamma **(b)** Adjust **(c)** Errors

Figure 4.9: 3D Visualization, adjustments and errors for Bates Gamma

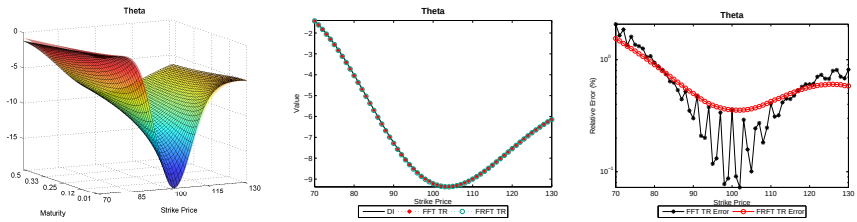
The inclusion of jumps give us a worse adjustment of Gamma than we obtained for the Heston model, as we can see in figure (4.9c). In this case, the error in adjustment is in arithmetic mean 0.25% in FFT case and 0.08% for FRFT algorithm, but in any case, Fourier algorithms continuously being an alternative due to they offer us a minimal error compared with FD method and competitive CPU times, specially in FRFT case, as we see in the next table.

Bates Gamma			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	0.0227	0.0000	0.0010
Finite Differences	0.0228	-0.4922	0.0040
FFT Trapezoidal Rule	0.0227	-0.0101	0.0150
FFT Simpson's Rule	0.0227	-0.0101	0.0200
FRFT Trapezoidal Rule	0.0227	-0.0101	0.0020
FRFT Simpson's Rule	0.0227	-0.0101	0.0010
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06$, $\rho = 0.9$			
$\Lambda = 3$, $\mu_S = -0.05$, $\sigma_S = 10^{-4}$			

Table 4.4: Results for Bates Gamma.

As we commented in the previous paragraph and you can see in table (4.4), FRFT algorithm computes Gamma Greek so faster than the direct integration of semi-closed solution via Gauss-Laguerre quadrature and it can be at least until four times more faster than FD method when we compute this Greek for ATM options, allowing reach an error of $10^{-2}\%$, while FD method is slower than FRFT and presents an error in first decimal place. FFT is so accurate than FRFT, but it is more slower in its two version, trapezoidal and Simpson's rules.

The next Greek that we will comment is Theta. It is showed in figure (4.10).



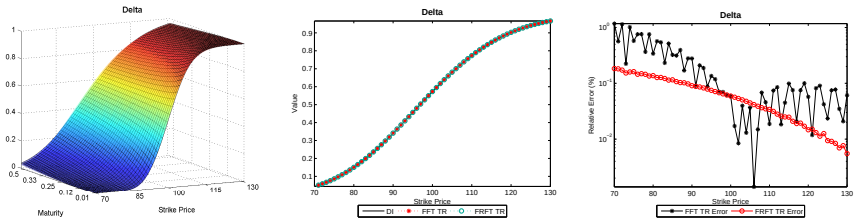
(a) Theta **(b)** Adjust **(c)** Errors

Figure 4.10: 3D Visualization, adjustments and errors for Bates Theta

Bates Theta			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	-9.1909	0.0000	0.0050
Finite Differences	-9.1912	-0.0038	0.0030
FFT Trapezoidal Rule	-9.2236	0.3564	0.0150
FFT Simpson's Rule	-9.2236	0.3564	0.0220
FRFT Trapezoidal Rule	-9.2236	0.3564	0.0020
FRFT Simpson's Rule	-9.2236	0.3564	0.0020
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06, \rho = 0.9$			
$\Lambda = 3$, $\mu_S = -0.05$, $\sigma_S = 10^{-4}$			

Table 4.5: Results for Bates Theta.

To finish with Greeks, we show the remaining figures for Delta, Rho and Vega 1 in this order.

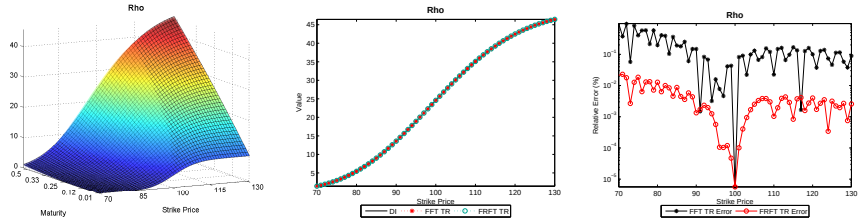


(a) Delta **(b)** Adjust **(c)** Errors

Figure 4.11: 3D Visualization, adjustments and errors for Bates Delta

It can be thought that Delta Greek in the Bates model should be calculated more precisely than Gamma, due to Delta is a first order Greek, whereas Gamma implies a second order derivative, but numerical results do not reflect this fact, being them, very similar in appearance. In any case, the error in adjustment is in arithmetic mean about 0.23% for FFT and 0.07% for FRFT, which they are some lightly lower than Gamma errors.

Figure (4.12), shows the Rho Greek.

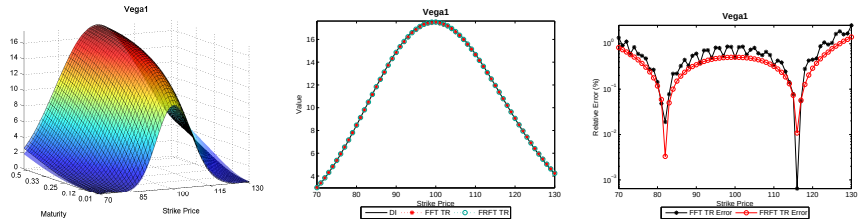


(a) Rho **(b)** Adjust **(c)** Errors

Figure 4.12: 3D Visualization, adjustments and errors for Bates Rho

For this Greek, we can observe in figure (4.12c) an accurate adjustment, being the errors in arithmetic mean of 0.18% for FFT and 4.7×10^{-3}% for FRFT. Furthermore, CPU times are shorter than FD method.

Next figure shows the results for the last Greek, Vega 1.

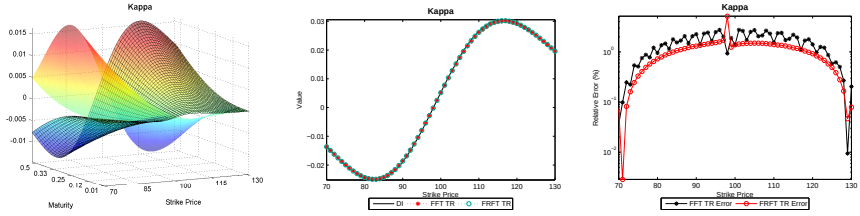


(a) Vega 1 **(b)** Adjust **(c)** Errors

Figure 4.13: 3D Visualization, adjustments and errors for Bates Vega 1

Vega 1 reflects the fact that both Fourier methods offer us a similar adjustment; 0.65% and 0.42% for FFT and FRFT algorithms respectively, having increased FRFT mean error as regards to the other cases.

We continuously now with other sensitivities. We begin analyze Kappa in first place.



(a) Kappa **(b)** Adjust **(c)** Errors

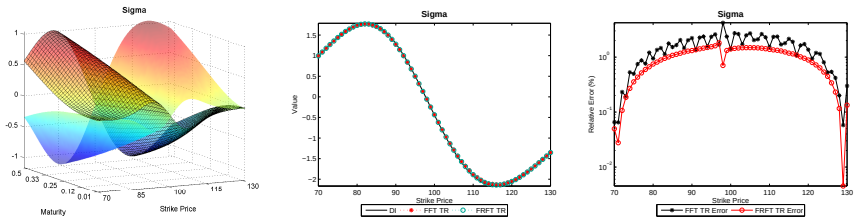
Figure 4.14: 3D Visualization, adjustments and errors for Bates Kappa

As we will see later when we have presented Kappa and Sigma show large errors in compare with other Greeks presented here, as it can be seen in figures (4.14) and (4.15) respectively, which it is due to it has been gathered here the jumps in stock price SDE with the fact that these derivatives for both parameters have been calculated symbolically. For Kappa, its errors in the adjustment through Fourier algorithms are in arithmetic mean: 1.48% for FFT and 1.04% for FRFT. The adjustment of Kappa is the worst among all done for this model. On the other hand, it can be noticed in table (4.6), where we can find moreover, that CPU times have been increased for all the methods except FD, which it is independent of any symbolic derivative.

Bates Kappa			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	0.0052	0.0000	0.0270
Finite Differences	0.0052	0.0001	0.0030
FFT Trapezoidal Rule	0.0052	-1.3846	0.0380
FFT Simpson's Rule	0.0052	-1.3846	0.0340
FRFT Trapezoidal Rule	0.0052	-1.3846	0.0190
FRFT Simpson's Rule	0.0052	-1.3846	0.0180
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06, \rho = 0.9$			
$\Lambda = 3$, $\mu_S = -0.05$, $\sigma_S = 10^{-4}$			

Table 4.6: Results for Bates Kappa.

Sigma is showed in the next figure.

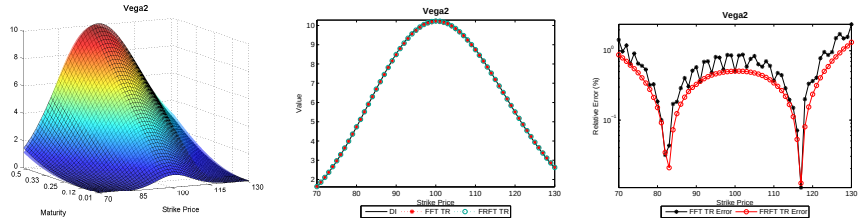


(a) Sigma **(b)** Adjust **(c)** Errors

Figure 4.15: 3D Visualization, adjustments and errors for Bates Sigma

Here, we can observed the same problems that we found in Kappa. Arithmetical mean errors are 1.51% and 0.97% for FFT and FRFT respectively.

We finish with Vega 2, which is presented in figure (4.16).



(a) Vega 2 **(b)** Adjust **(c)** Errors

Figure 4.16: 3D Visualization, adjustments and errors for Bates Vega 2

In this case, the adjustment and CPU times are very similar from Vega 1. Errors are 0.64% for FFT and 0.42% for FRFT.

4.4 The SVJJ (2000) Model

In the first part of this section, Greeks for the SVJJ model will be derived in their analytic formula. Other sensitivities as regards as parameters like kappa, sigma and theta are also having into account, but only the sensitivity as regards as theta will be derived. The remaining sensitivities have been calculated symbolically by means of MATLAB.

The goal of the second part of this section is to compare the results obtained when we compute the Greeks and other sensitivities for the SVJJ model by means of methods as finite differences or Fourier algorithms. We will compare them in relation with the results provided by Direct Integration of semi-closed solution via Gauss-Laguerre quadrature.

4.4.1 Greeks and other Sensitivities

Differentiating equation (4.1) and considering equation (1.9), we have that Delta, Gamma, Rho and Vegas are the same that obtained before for the Heston model. However, Theta acquires a different shape, as we will see through this section.

$$\Delta_{SVJJ} = \frac{\partial c}{\partial S} = e^{-q\tau} P_1 \tag{4.15}$$

$$\Gamma_{SVJJ} = \frac{\partial^2 c}{\partial S^2} = e^{-q\tau} \frac{\partial P_1}{\partial S} = \frac{e^{-q\tau}}{\pi S_t} \int_0^\infty Re\left[e^{-i\varphi \ln K} \phi_1(\varphi; x_t, v_t)\right] d\varphi \tag{4.16}$$

$$\rho_{SVJJ} = \frac{\partial^2 c}{\partial r} = K\tau e^{-r\tau} P_2 \tag{4.17}$$

while Theta is given by

$$\Theta_{SVJJ} = -\frac{\partial c}{\partial \tau} = -S_t e^{-q\tau}\left(-qP_1 + \frac{\partial P_1}{\partial \tau}\right) + Ke^{-r\tau}\left(-rP_2 + \frac{\partial P_2}{\partial \tau}\right) \tag{4.18}$$

Being

$$\frac{\partial P_j}{\partial \tau} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{\partial \phi_j}{\partial \tau} \times \frac{e^{-i\varphi \ln K}}{i\varphi}\right] d\varphi$$

where

$$\frac{\partial \phi_j}{\partial \tau} = exp(C_j + D_j v_t + \tilde{P}_j \lambda + i\varphi x_t)\left(\frac{\partial C_j}{\partial \tau} + \frac{\partial D_j}{\partial \tau} v_t + \frac{\partial \tilde{P}_j}{\partial \tau}\lambda\right)$$

and

$$\begin{aligned}
\frac{\partial C_j}{\partial \tau} &= (r-q)\varphi i + \frac{\kappa\theta}{\sigma^2}\left[b_j - \rho\sigma\varphi i + d_j + \frac{2g_j d_j e^{d_j\tau}}{1-g_j e^{d_j\tau}}\right] \\
\frac{\partial D_j}{\partial \tau} &= \frac{b_j - \rho\sigma\varphi i + d_j}{\sigma^2}\left[\frac{(g_j-1)d_j e^{d_j\tau}}{(1-g_j e^{d_j\tau})^2}\right] \\
\frac{\partial \tilde{P}_j}{\partial \tau} &= -(1+i\varphi\mu_J) + exp\left[i\varphi\mu_S + \frac{\sigma_S^2 (i\varphi)^2}{2}\right](\delta_j + \chi_j \xi_j)
\end{aligned}$$

where

$$\begin{aligned}
\delta_j &= \frac{\beta_j + d_j}{(\beta_j + d_j)c - 2\mu_V \alpha} \\
\chi_j &= \frac{4\mu_V \alpha}{(d_j c)^2 - (2\mu_V \alpha - \beta_j c)^2} \\
\xi_j &= \frac{\left[(d_j - \beta_j)c + 2\mu_V \alpha\right] c e^{-d_j\tau}}{2d_j c - \left[(d_j - \beta_j)c + 2\mu_V \alpha\right](1 - e^{-d_j\tau})}
\end{aligned}$$

and

$$\begin{aligned}
\alpha &= -\frac{(\varphi^2 + i\varphi)}{2} \\
\beta_j &= b_j - \rho\sigma_V i\varphi \\
\gamma &= \frac{\sigma_V^2}{2}
\end{aligned}$$

where b_j was defined in the Heston model section.

Finally, Vegas are given by

$$\mathcal{V}^1_{SVJJ} = \frac{\partial c}{\partial v} = \frac{\partial c}{\partial v_0} 2\sqrt{v_0}$$
$$\mathcal{V}^2_{SVJJ} = \frac{\partial c}{\partial \omega} = \frac{\partial c}{\partial \theta} 2\sqrt{\theta}$$

Being the first Vega

$$\mathcal{V}^1_{SVJJ} = Se^{-q\tau}\frac{\partial P_1}{\partial v_0} 2\sqrt{v_0} - Ke^{-r\tau}\frac{\partial P_2}{\partial v_0} 2\sqrt{v_0} \tag{4.19}$$

where

$$\frac{\partial P_j}{\partial v_0} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\varphi \ln K}\phi_j(\varphi; x_t, v_t) D_j(\tau, \varphi)}{i\varphi}\right] d\varphi.$$

and being the second Vega

$$\mathcal{V}^2_{SVJJ} = Se^{-q\tau}\frac{\partial P_1}{\partial \theta} 2\sqrt{\theta} - Ke^{-r\tau}\frac{\partial P_2}{\partial \theta} 2\sqrt{\theta} \tag{4.20}$$

where

$$\frac{\partial P_j}{\partial \theta} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\varphi \ln K}\phi_j(\varphi; x_t, v_t)\partial C_j/\partial\theta)}{i\varphi}\right] d\varphi.$$

and

$$\frac{\partial C_j}{\partial \theta} = \frac{\kappa}{\sigma^2}\left[(b_j - \rho\sigma\varphi i + d_j)\tau - 2\ln\left(\frac{1 - g_j e^{d_j\tau}}{1 - g_j}\right)\right]$$

4.4.2 Numerical Results

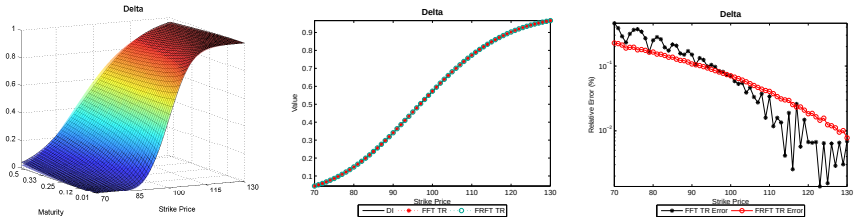
It will be presented here, the numerical results for the SVJJ model. Unlike of Bates model, this model includes jumps in variance SDE and of course, in the stock price equation. Due to these jumps, the accurate in values obtained will be lowered depending on the jumps parameters that has been considered. For this reason, it has been chosen a jump parameters that are not too large in the variance SDE, whereas jump parameters in stock price SDE, are the same that it has been employed in Bates model.

For this model, all the graphics and results showed in this section, corresponding to an European call option with the following conditions: $K = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06, \rho = 0.9$, $\Lambda = 3$, $\mu_S = 0.014$, $\sigma_S = 10^{-4}$, $\rho_J = -0.4$, $\mu_V = 0.01$ and it has been considered again, the same spot range of $S_0 \in [70, 130]$. In relation to Fourier algorithms, we have implemented FFT considering $\alpha = 1.75$, a grid of $N = 2^{10}$ integration points and an

upper limit of integration of $uplimit = 400$, whilst FRFT algorithm has been implemented considering $N = 2^8$ integration points an upper limit of integration of $uplimit = 200$. As it can see here, the number of integration points has been increased for FFT algorithm in one magnitude order, as well as the upper limit of integration, that it has been increased from 200 to 400. It is due to the fact that this model is a model that include jumps in both SPDE and for this reason, it is more difficult to get a better results in Greeks and other sensitivities if N and $uplimit$ are not increased. On the other hand, it has been taken the same parameters $\eta = 0.1$ and $\lambda = 0.005$ for FRFT algorithm. The reason for what it has been decided increased $uplimit$ to N in FRFT is due to the CPU times and errors are faster and accurate respectively than any other combination.

In this model, we show the table of results for Delta, Rho and Sigma only, whereas the remainder Greeks and other sensitivities will be showed graphically. As always, 3D visualizations have been obtained considering the extreme correlations of 0.9 and -0.9, being each of them represented by means of different shaded graphics. Bidimensional cuts have been taken at maturity for the parameters indicated in before.

We begin this section showing the results for Delta Greek, which it is presented in figure (4.17)



(a) Delta **(b)** Adjust **(c)** Errors

Figure 4.17: 3D Visualization, adjustments and errors for SVJJ Delta

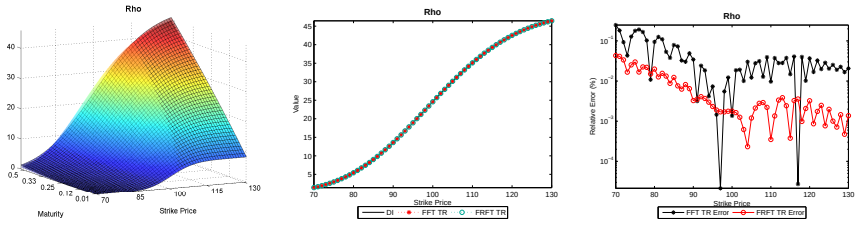
As it can seen in figure (4.17c), errors in the adjustment of Delta Greek are similar to the Bates model, despite of having two SDJE. If we analyze errors in arithmetic mean, we have that 0.11% is the error for FFT method, whilst 0.09% is the error for FRFT algorithm, which reflects the fact that both methods get a fine adjustment even in double jump conditions. However, CPU times are greater than Bates model, as it can see in table (4.7)

SVJJ Delta			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	0.5723	0.0000	0.0010
Finite Differences	0.5725	-0.0380	0.0030
FFT Trapezoidal Rule	0.5727	0.0704	0.0550
FFT Simpson's Rule	0.5727	0.0703	0.0800
FRFT Trapezoidal Rule	0.5727	0.0707	0.0010
FRFT Simpson's Rule	0.5727	0.0707	0.0010
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06$, $\rho = 0.9$			
$\Lambda = 3$, $\mu_S = 0.014$, $\sigma_S = 10^{-4}$, $\rho_J = -0.4$, $\mu_V = 0.01$			

Table 4.7: Results for SVJJ Delta.

All Fourier methods reach the same accuracy order when Delta Greek is computed for ATM options, but FRFT algorithm is the most faster among them, up to fifty-five times faster than FFT and even so faster than closed form integration. CPU times are logically greater for FFT than FRFT due to the fact that FFT has been implemented with a greater N and *uplimit* than FRFT, but both methods have been compared to reflect the fact that for reaching the same accuracy order, FRFT is with difference, the fastest algorithm.

We continue our analysis with Rho Greek.



(a) Rho **(b)** Adjust **(c)** Errors

Figure 4.18: 3D Visualization, adjustments and errors for SVJJ Rho

By its mathematical definition, Rho is a Greek that always have been well implemented by Fourier methods in all previous models and the SVJJ is not an exception. Error in the adjustment are very small, as it can seen in figure (4.18c), being $4.7 \times 10^{-2}\%$ and $7.5 \times 10^{-3}\%$ the error in arithmetic mean for FFT and FRFT respectively. Neither other Greek for this model is calculated

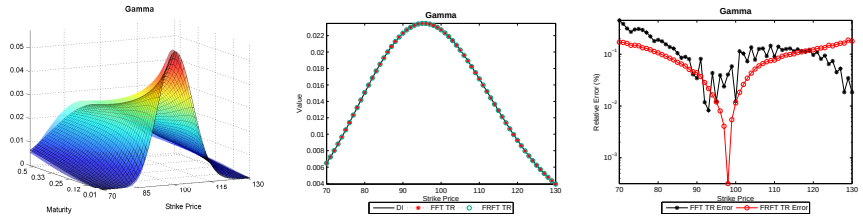
with an accuracy order so fine as Rho. Table (4.8) summarizes the results for ATM options.

SVJJ Rho			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	24.5811	0.0000	0.0010
Finite Differences	24.5948	-0.0558	0.0030
FFT Trapezoidal Rule	24.5814	0.0014	0.0790
FFT Simpson's Rule	24.5814	0.0014	0.0560
FRFT Trapezoidal Rule	24.5815	0.0018	0.0010
FRFT Simpson's Rule	24.5815	0.0018	0.0010
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06, \rho = 0.9$			
$\Lambda = 3$, $\mu_S = 0.014$, $\sigma_S = 10^{-4}$, $\rho_J = -0.4$, $\mu_V = 0.01$			

Table 4.8: Results for SVJJ Rho.

It can be observed here, how for the same accuracy order, FRFT continues offering us the best CPU times among Fourier methods and the configuration chosen for this algorithm, make it so fast as Gauss-Laguerre quadrature implemented in closed form and three times faster than FD method.

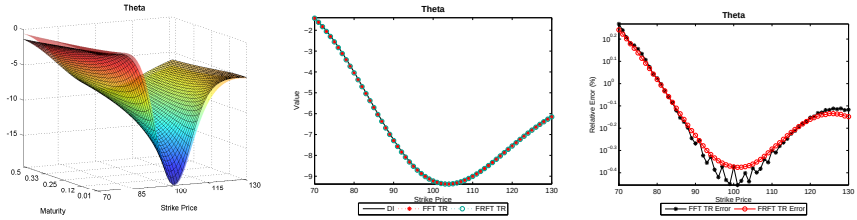
The remaining Greeks calculated will be commented briefly in the next paragraphs.



(a) Gamma **(b)** Adjust **(c)** Errors

Figure 4.19: 3D Visualization, adjustments and errors for SVJJ Gamma

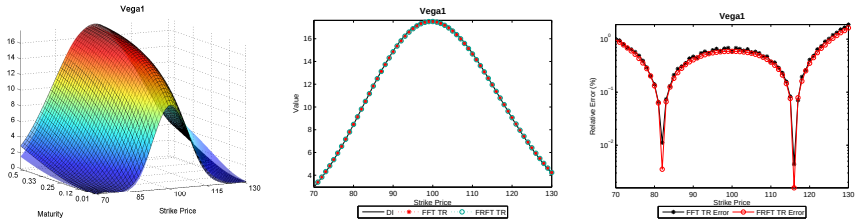
The error in the adjustment for Gamma Greek is in arithmetic mean 0.12% for FFT and 0.09% for FRFT, showing both Fourier algorithms a similar behaviour in this case. CPU times are similar to Delta Greek with any relevant increment.



(a) Theta **(b)** Adjust **(c)** Errors

Figure 4.20: 3D Visualization, adjustments and errors for SVJJ Theta

For Theta, the errors in arithmetic mean are 0.74% in both cases, as we can check in figure (4.20), where it be can observed that the adjustments are very close as regards to each other.

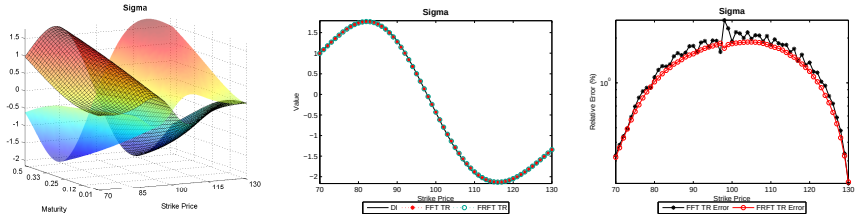


(a) Vega 1 **(b)** Adjust **(c)** Errors

Figure 4.21: 3D Visualization, adjustments and errors for SVJJ Vega 1

Vega 1 is the last Greek under study and it is a good example of FRFT is an enhanced version of FFT algorithm. It can be observed in figure (4.21), that both methods offer us errors that in arithmetic mean they are 0.55% for FFT and 0.49% FRFT, but where CPU times are up to seventy times more faster for FRFT than FFT algorithm, as it has been checked.

This section will be finished with an analysis of other sensitivities. We analyze Sigma in first place, whose results can been observed in figure (4.22)



(a) Sigma **(b)** Adjust **(c)** Errors

Figure 4.22: 3D Visualization, adjustments and errors for SVJJ Sigma

In the same way as it was commented for the last Greek, Vega 1, errors for Sigma are also very close for each other for both Fourier methods, as can be seen in figure (4.22c). However, it is important to point out that for first time in all analysis made so far, error are very large in this case, being them of 1.41% and 1.28% in arithmetic mean for FFT and FRFT respectively. As it was explained in other sections, it is manly due to the way in derivatives have been calculated[2], as well as the presence of jumps in both SPDE. Accordingly, CPU times have been increased for Sigma Greek for all the methods, as it can be observed in table (4.9)

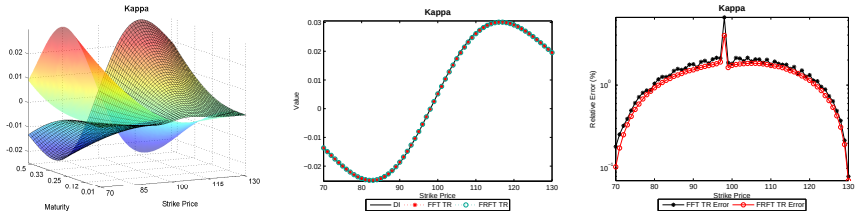
SVJJ Sigma			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	-0.4404	0.0000	0.0120
Finite Differences	-0.4403	0.0163	0.0030
FFT Trapezoidal Rule	-0.4320	-1.8962	0.1170
FFT Simpson's Rule	-0.4320	-1.8962	0.1030
FRFT Trapezoidal Rule	-0.4323	-1.8305	0.0150
FRFT Simpson's Rule	-0.4323	-1.8324	0.0160
$S_0 = 100$, $\kappa = 2$, $\theta = 0.06$, $\sigma_V = 0.1$, $v_0 = 0.06, \rho = 0.9$			
$\Lambda = 3$, $\mu_S = 0.014$, $\sigma_S = 10^{-4}$, $\rho_J = -0.4$, $\mu_V = 0.01$			

Table 4.9: Results for SVJJ Sigma.

In this case, Finite Differences is the best choice when we calculated derivatives in symbolic form, being it up to five times faster than FRFT algorithm and four times more faster than the integration of closed form via Gauss-Laguerre quadrature. FD is also the most accurate method, up to two magnitude orders compared with any Fourier method.

Next, it will be seen in figure (4.23), as Kappa shows a similar behaviour that Sigma.

[2]Derivatives for Kappa and Sigma have been calculated numerically by means of MATLAB.

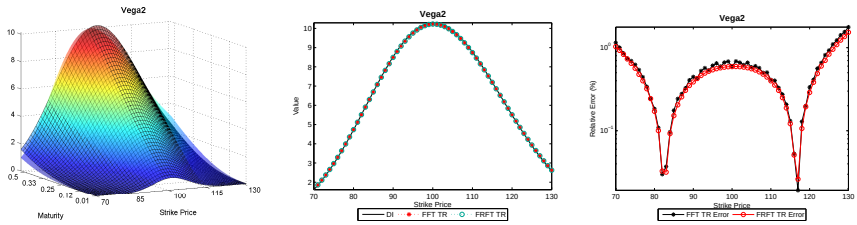


(a) Kappa **(b)** Adjust **(c)** Errors

Figure 4.23: 3D Visualization, adjustments and errors for SVJJ Kappa

If figure (4.23c) is observed, and we remember the values of errors in arithmetic mean for Sigma, it can be seen that errors for Kappa; 1.43% for FFT and 1.25% for FRFT are very close to the Sigma errors, which it can be explained, one more time, for the same reasons that it was commented before. It can be checked that CPU times are also very similar for Kappa. In any case, both Fourier methods are a reasonable approximation when we are trying to calculated other sensitivities, although FD is the most accurate and faster method of all.

This section finish with the graphics for Vega 2, summarizes in figure (4.24)



(a) Vega 2 **(b)** Adjust **(c)** Errors

Figure 4.24: 3D Visualization, adjustments and errors for SVJJ Vega 2

Vega 2 is calculated in a similar way as Vega 1, so errors are very close to the last one in this case, being them, 0.54% and 0.49% for FFT and FRFT respectively. In relation to CPU times, they are also very close to the times for Vega 1 Greek in both cases.

4.5 The Double Heston (2009) Model

In the first part of this section, Greeks for the Double Heston model will be showed in their analytic formula. Other sensitivities as regards as parameters like kappa, sigma and theta are also having into account, but only the sensitivity as regards as theta will be calculated. The remaining sensitivities have been calculated symbolically by means of MATLAB.

The goal of the second part of this section is to compare the results obtained when we compute the Greeks and other sensitivities for the Double Heston model by means of methods as finite differences or Fourier algorithms. We will compare them in relation with the results provided by Direct Integration of semi-closed solution via Gauss-Laguerre quadrature.

4.5.1 Greeks and other Sensitivities

Differentiating equation (4.1) and considering equations (3.13), we have that Delta, Gamma, Rho and Vegas are the same that obtained before for the Heston model. However, Theta acquires a different aspect, as we will see later. Delta and Gamma are respectively

$$\Delta_{DH} = \frac{\partial c}{\partial S} = e^{-q\tau} P_1 \tag{4.21}$$

$$\begin{aligned} \Gamma_{DH} &= \frac{\partial^2 c}{\partial S^2} \\ &= \frac{e^{-q\tau}}{\pi} \int_0^\infty Re\left[\frac{e^{-i\varphi \ln K} f(\varphi - i; x_t, v_{1,t}, v_{2,t})}{S_t^2 e^{(r-q)\tau}} \right] d\varphi \end{aligned} \tag{4.22}$$

whereas Rho is

$$\rho_{DH} = \frac{\partial^2 c}{\partial r} = K\tau e^{-r\tau} P_2 \tag{4.23}$$

while Theta is given by

$$\Theta_{DH} = -\frac{\partial c}{\partial \tau} = -S_t e^{-q\tau} \left(-qP_1 + \frac{\partial P_1}{\partial \tau} \right) + Ke^{-r\tau} \left(-rP_2 + \frac{\partial P_2}{\partial \tau} \right) \tag{4.24}$$

where

$$\frac{\partial P_1}{\partial \tau} = \frac{1}{\pi}\int_0^\infty Re\left\{\frac{e^{-i\varphi \ln K}}{i\varphi S_t e^{(r-q)\tau}}\left[\frac{\partial f}{\partial \tau} - (r-q)f\right]\right\} d\varphi$$
$$\frac{\partial P_2}{\partial \tau} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\varphi \ln K}}{i\varphi}\left(\frac{\partial f}{\partial \tau}\right)\right] d\varphi$$

and

$$\frac{\partial f}{\partial \tau} = exp\left(A + i\varphi x_t + B_1 v_{1,t} + B_2 v_{2,t}\right)\left(\frac{\partial A}{\partial \tau} + \frac{\partial B_1}{\partial \tau}v_{1,t} + \frac{\partial B_2}{\partial \tau}v_{2,t}\right)$$

Where A and B were defined in equation (3.16). Finally, the two Vegas are defined by

$$\mathcal{V}_{DH}^{1j} = Se^{-q\tau}\frac{\partial P_1}{\partial v_{0j}}2\sqrt{v_{0j}} - Ke^{-r\tau}\frac{\partial P_2}{\partial v_{0j}}2\sqrt{v_{0j}} \tag{4.25}$$

for $j = 1, 2.$ and considering that as $\mathcal{V}_{DH}^{11}$[3] is based on $v_1 = \sqrt{v_{01}}$ and $\mathcal{V}_{DH}^{12}$ is based on $v_2 = \sqrt{v_{02}}$, then

$$\frac{\partial P_1}{\partial v_{0j}} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\varphi \ln K}}{i\varphi S_t e^{(r-q)\tau}} f\left(\varphi - i; x_t, v_{1,t}, v_{2,t}\right) B_j(\tau, \varphi - i)\right] d\varphi$$
$$\frac{\partial P_2}{\partial v_{0j}} = \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-i\varphi \ln K}}{i\varphi} f\left(\varphi; x_t, v_{1,t}, v_{2,t}\right) B_j(\tau, \varphi)\right] d\varphi$$

4.5.2 Numerical Results

It will be presented here, the numerical results for the Double Heston model. Unlike of all previous model, in this section it has been more difficult make the trade-off between accuracy and CPU times choosing suitably the parameters N and *uplimit* for Fourier methods, so it has been considered include two different alternatives for FRFT algorithm for the purpose to prove the fine adjustment that it can be made with this method. These analysis will be collected with the inclusion of another function in error figures and the inclusion of two more lines of results in the tables.

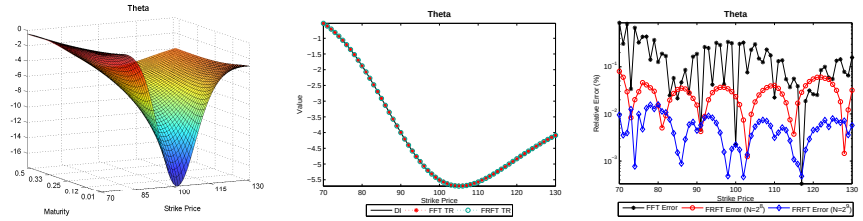
For this model, all the graphics and results showed in this section, corresponding to an European call option with the following conditions: $K = 100$, $\kappa_1 = 2$, $\theta_1 = 0.005$, $\sigma_1 = 0.2$, $v_{01} = 0.04, \rho_1 = 0.6$, $\kappa_2 = 1.5$, $\theta_2 = 0.006$, $\sigma_2 = 0.25$, $v_{02} = 0.03, \rho_2 = -0.6$ and it has been considered again, the same spot range of

[3]It is important to point out that in this model, we have not consider Vegas that involving derivatives as regards to θ

$S_0 \in [70, 130]$. On the other hand, Fourier algorithms have been implemented considering a dampening factor of $\alpha = 1.75$ and $N = 2^8$ and $N = 2^9$ integration points for FFT and FRFT algorithms respectively, whilst the upper limit of integration is 100 for both methods. Along this section, it will be checked that FRFT method with $N = 2^9$ will be the most accurate algorithm, while FRFT method with $N = 2^8$ will be faster than direct integration of semi-closed solution via Gauss-Laguerre quadrature[4].

In the next pages, it will be showed the table of results for Theta, Vega 11 and Vega 22 only, whereas the remainder Greeks and other sensitivities will be showed graphically. As always, 3D visualizations have been obtained considering the extreme correlations of 0.9 and -0.9, being each of them represented by means of different shaded graphics. Bidimensional cuts have been taken at maturity for the parameters previously indicated.

We begin this section showing the results for Theta Greek, which it is presented in figure (4.25)



(a) Theta **(b)** Adjust **(c)** Errors

Figure 4.25: 3D Visualization, adjustments and errors for Double Heston Theta

The new element in relation with other figures analyzed before is the inclusion of blue function in figure (4.25c), which represents the error in the adjustment when FRFT algorithm has been implemented considering $N = 2^9$ integration points. It can be seen here that with this configuration, error adjustment decreased in arithmetic mean from $2.9 \times 10^{-2}\%$ for FRFT algorithm with $N = 2^8$ to $5.3 \times 10^{-3}\%$, which represents one magnitude order. On the other and, error for FFT is approximately about 0.17%. In relation to CPU times, it can been checked that this magnitude order acquired for FRFT_9 algorithm, is accompanied by an increment of 45.71% of CPU time in arithmetic mean as regards to FRFT_8 algorithm, so it is clear here, that trade-off between accuracy and CPU time requires a very fine adjustment of Fourier parameters.

Table (4.10) summarizes the results for Theta Greek for ATM options.

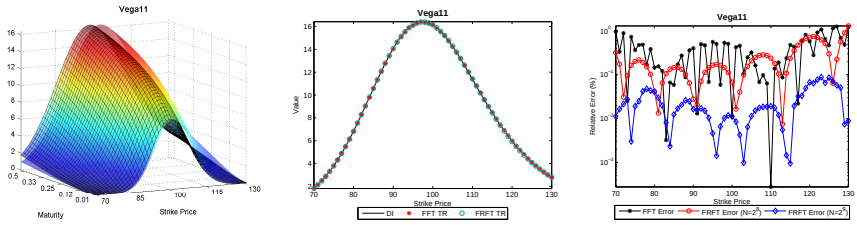
[4]From this moment, we refer to FRFT algorithm implemented with $N = 2^9$ as FRFT_9 and to FRFT algorithm implemented with $N = 2^8$ as FRFT_8.

Double Heston Theta			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	-5.5090	0.0000	0.0060
Finite Differences	-5.5094	-0.0088	0.0040
FFT Trapezoidal Rule	-5.5091	0.0022	0.0680
FFT Simpson's Rule	-5.5091	0.0022	0.0690
FRFT$_8$ TR	-5.5077	-0.0233	0.0020
FRFT$_9$ TR	-5.5091	0.0022	0.0020
FRFT$_8$ SR	-5.5077	-0.0227	0.0020
FRFT$_9$ SR	-5.5091	0.0022	0.0030
$S_0 = 100$, $\kappa_1 = 2$, $\theta_1 = 0.005$, $\sigma_1 = 0.2$, $v_{01} = 0.04$, $\rho_1 = 0.6$			
$\kappa_2 = 1.5$, $\theta_2 = 0.006$, $\sigma_2 = 0.25$, $v_{02} = 0.03$, $\rho_2 = -0.6$			

Table 4.10: Results for Double Heston Theta.

This table summarizes all previous comments for Theta Greek. In addition to everything mentioned above, one relevant aspect is that both FRFT algorithms are faster than any other method to compute Theta and in the case of FRFT$_9$ it is up to four times more accurate than FD method. It can be seen here, as to reach the same order that FRFT with FFT algorithm, it is necessary invest approximately thirty times more time, so as it has been discussing until now, FRFT is a best choice in comparison with FFT method.

We continue our analysis with Vega 11, showing in first place all its graphics in figure (4.26)



(a) Vega 11 **(b)** Adjust **(c)** Errors

Figure 4.26: 3D Visualization, adjustments and errors for Double Heston Vega 11

This time, it has been observed that error in the adjustment is in arithmetic mean 0.39% for FFT, 0.25% for FRFT$_8$ and 2.5×10^{-2}% for FRFT$_9$, so that all Fourier algorithms continues being an accurate way to compute Greeks. On

the other hand, CPU times are in inverse relation, being the fastest method FRFT_8, requiring a 51.35% less time in compute Vega 11 than FRFT_9. FFT method does not enter in this comparison, because it is much slower than FRFT algorithms. It is important to point out that FRFT_8 method is in arithmetic mean near to 198.54% faster than direct integration via Gauss-Laguerre.

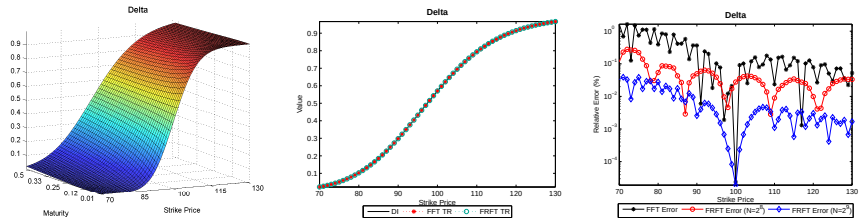
Table (4.11) shows results corresponding to the previous comments.

Double Heston Vega 11			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	16.1568	0.0000	0.0040
Finite Differences	16.1570	-0.0011	0.0040
FFT Trapezoidal Rule	16.1587	0.0113	0.0660
FFT Simpson's Rule	16.1587	0.0113	0.0820
FRFT_8 TR	16.1458	-0.0684	0.0020
FRFT_9 TR	16.1587	0.0112	0.0020
FRFT_8 SR	16.1460	-0.0670	0.0010
FRFT_9 SR	16.1587	0.0112	0.0020
$S_0 = 100,\ \kappa_1 = 2,\ \theta_1 = 0.005,\ \sigma_1 = 0.2,\ v_{01} = 0.04, \rho_1 = 0.6$			
$\kappa_2 = 1.5,\ \theta_2 = 0.006,\ \sigma_2 = 0.25,\ v_{02} = 0.03, \rho_2 = -0.6$			

Table 4.11: Results for Double Heston Vega 11.

In this table it can been observed that FD is the most accurate method for ATM options, but not the most faster, having ascertained that any FRFT method is more faster than the first one.

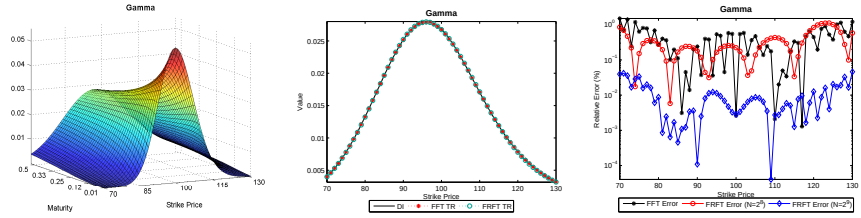
For the remaining Greeks, there are not any strangeness in the results, therefore they only will comment briefly.



(a) Delta **(b)** Adjust **(c)** Errors

Figure 4.27: 3D Visualization, adjustments and errors for Double Heston Delta

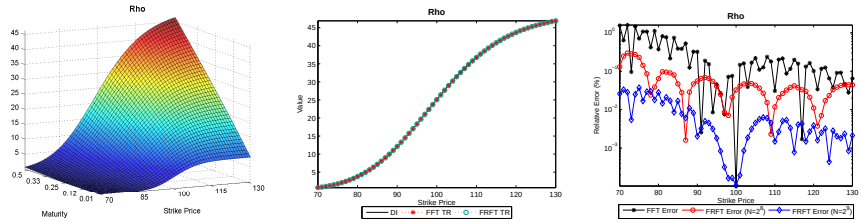
For Delta, it has been obtained that errors in arithmetic mean are: 0.31%, 5.0×10^{-2}% and 8.1×10^{-3}% for FFT, $FRFT_8$ and $FRFT_9$ algorithms respectively, reflecting these results the fact that magnitude order decreased one order in each refinement. Having CPU times into account, it has been checked that $FRFT_8$ is a 54.8% faster than $FRFT_9$ and 6.8% than DI form.



(a) Gamma **(b)** Adjust **(c)** Errors

Figure 4.28: 3D Visualization, adjustments and errors for Double Heston Gamma

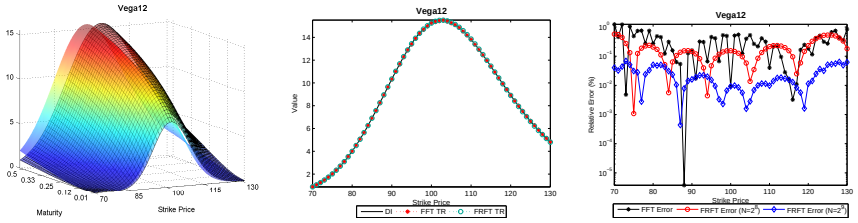
A similar behaviour it can be observed in Gamma results, since 0.43% 0.34% 1.1×10^{-2}% are the error in the adjustment for FFT, $FRFT_8$ and $FRFT_9$ respectively, whilst $FRFT_8$ is a 36.25% faster than $FRFT_9$ and a 2.51% than DI form.



(a) Rho **(b)** Adjust **(c)** Errors

Figure 4.29: 3D Visualization, adjustments and errors for Double Heston Rho

In the same line as it has been commented in the previous paragraphs, Rho adjustments present an identical behaviour, being 0.31% for FFT, 6.1×10^{-2}% for $FRFT_8$ and 8.2×10^{-3}% for $FRFT_9$ and $FRFT_8$ is near to 38.96% faster than $FRFT_9$ method and a 9.09% than DI form.

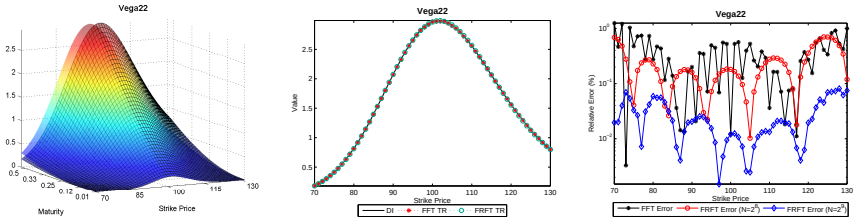


(a) Vega 12 **(b)** Adjust **(c)** Errors

Figure 4.30: 3D Visualization, adjustments and errors for Double Heston Vega 12

The last Greek that we analyze is Vega 12, whose adjustment analysis is very similar to Vega 11, being the errors in arithmetic mean of 0.17%, 2.8×10^{-2}% and 5.6×10^{-3}% for FFT, FRFT_8 and FRFT_9 respectively. However, the most relevant aspect is that for CPU times, it has been find that FRFT_8 is in arithmetic mean near to 47.43% faster than FRFT_9, but it is become up to 193.58% faster than DI form, which is a very remarkable result.

Next, it will be analyzed other sensitivities, where it will be only commented with table Vega 22



(a) Vega 22 **(b)** Adjust **(c)** Errors

Figure 4.31: 3D Visualization, adjustments and errors for Double Heston Vega 22

Vega 22 in Double Heston model is calculated with an moderate accuracy at it can be seen in figure (4.31c), where errors in arithmetic mean are 0.30%, 0.24% and 2.7×10^{-2}% for FFT, FRFT_8 and FRFT_9 respectively. This time, we have that FFT and FRFT_8 presented a similar precision order, but it is unimportant when CPU times are considered, because FRFT_8 is up to 57.53% faster than FRFT_9 and even 224.65% than DI form.

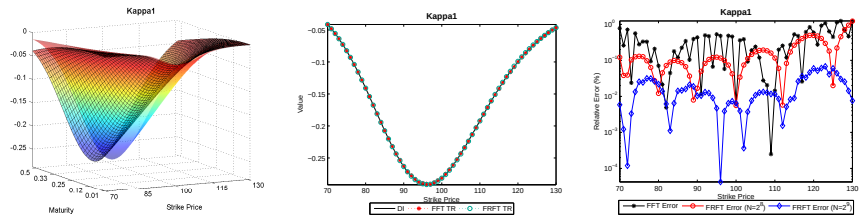
Table (4.12) shows the results for Vega 22.

Double Heston Vega 22			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	2.9707	0.0000	0.0040
Finite Differences	2.9707	-0.0004	0.0040
FFT Trapezoidal Rule	2.9711	0.0121	0.0790
FFT Simpson's Rule	2.9711	0.0121	0.0830
FRFT$_8$ TR	2.9653	-0.1813	0.0020
FRFT$_9$ TR	2.9711	0.0121	0.0020
FRFT$_8$ SR	2.9655	-0.1776	0.0010
FRFT$_9$ SR	2.9711	0.0121	0.0020
$S_0 = 100$, $\kappa_1 = 2$, $\theta_1 = 0.005$, $\sigma_1 = 0.2$, $v_{01} = 0.04$, $\rho_1 = 0.6$			
$\kappa_2 = 1.5$, $\theta_2 = 0.006$, $\sigma_2 = 0.25$, $v_{02} = 0.03$, $\rho_2 = -0.6$			

Table 4.12: Results for Double Heston Vega 22.

It can be checked that FD method is the most accurate method among all, but this accuracy has a disadvantage and it is CPU time, because FD is not so faster than any FRFT algorithm, but in the case of FFT, the last one is clearly a worse option than FD.

Figure (4.32) shows the results for Kappa 1.



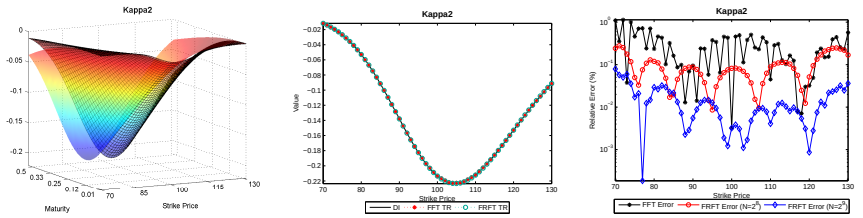
(a) Kappa 1 **(b)** Adjust **(c)** Errors

Figure 4.32: 3D Visualization, adjustments and errors for Double Heston Kappa 1

Until now, we have analyzed for this model some Greeks and other sensitivities, but Kappa 1 is the first that we will analyze that has been calculated symbolically, so due to our previous analysis we might be able to expected some relevant results, but if is not the case in relation to errors adjustment, because they are very similar to the errors in previous Greeks and parameter; 0.36% for FFT, 0.19% for FRFT$_8$ and 1.8×10^{-2}% for FRFT$_9$. However, it is true that there is one remarkable aspect and it is relative to CPU times, since in this case, the symbolic derivative makes an increment in times of 14.49%

when we are calculated Kappa 1 with $FRFT_8$ in relation to the CPU time elapsed when we do the same considering DI form. Anyway it is consequence only of the symbolic derivative, as it has been explained previously in other sections. $FRFT_8$ continues being the fastest method among Fourier algorithms, a 64.79% faster that $FRFT_9$ for example.

Kappa 2 is showed in figure (4.33)

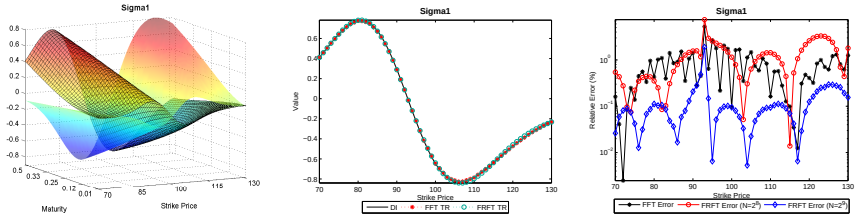


(a) Kappa 2 **(b)** Adjust **(c)** Errors

Figure 4.33: 3D Visualization, adjustments and errors for Double Heston Kappa 2

Any new it can be found here, due to everything relevant aspect have been commented in the last paragraph and Kappa 2 differs only from Kappa 1 in that derivative is calculated as regards to $\kappa 2$ instead of $\kappa 1$. Anyway, the errors in the adjustment are; 0.29%, 9.8×10^{-2}% and 1.5×10^{-2}% for FFT, $FRFT_8$ and $FRFT_9$ respectively, whereas in relation to CPU times we have that $FRFT_8$ is 63.88% faster than $FRFT_9$, but a 6.15% slower than DI form.

Sigma 1 is showed in figure (4.34)



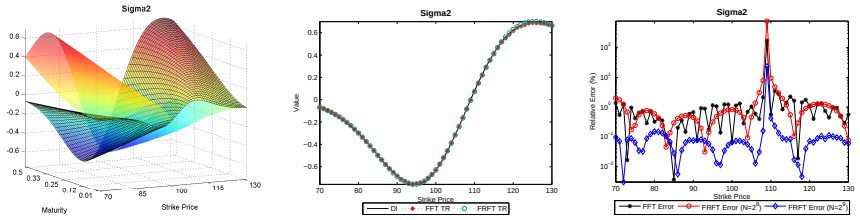
(a) Sigma 1 **(b)** Adjust **(c)** Errors

Figure 4.34: 3D Visualization, adjustments and errors for Double Heston Sigma 1

The results for Sigma 1 are in some way, surprising in what to an error adjustment it referred, because Sigma 1 and as we will see later, Sigma 2, are the only exceptions in Double Heston model considering this configuration in where mean error for $FRFT_8$ is greater than error in FFT, being they 0.82%, 1.28% and 0.14% for FFT, $FRFT_8$ and $FRFT_9$ respectively. It can be explained

due to values of σ_1 and σ_2 are shorter than κ_1 and κ_2, together with the fact that we calculated their derivatives symbolically also. $FRFT_8$ shows a CPU times a 65.90% faster than $FRFT_9$ and a 18.46% slower than DI form.

We will see now, the results for Sigma 2.

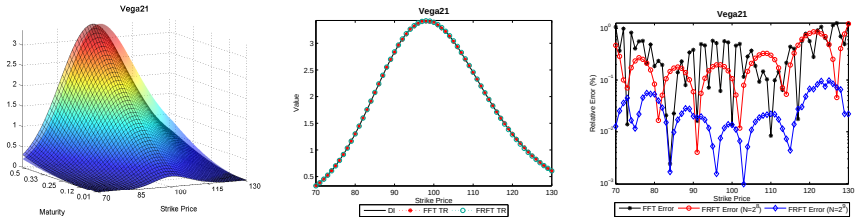


(a) Sigma 2 **(b)** Adjust **(c)** Errors

Figure 4.35: 3D Visualization, adjustments and errors for Double Heston Sigma 2

Sigma 2 presents the same characteristics mentioned before for Sigma 1, so the results will be only given without any relevant comment. Errors in the adjustment in arithmetic mean are 3.69%, 14.02% and 0.46% for FFT, $FRFT_8$ and $FRFT_9$ respectively, whereas $FRFT_8$ is faster than $FRFT_9$ in a 63.49%, but a 11.11% slower than DI form.

To finish this section, it will be showed in figure (4.36) results for Vega 21.



(a) Vega 21 **(b)** Adjust **(c)** Errors

Figure 4.36: 3D Visualization, adjustments and errors for Double Heston Vega 21

Vega 21 shows a similar behaviour to Vega 22, so we present here again the results, without any relevant input. Errors in the adjustment in arithmetic mean are 0.40%, 0.28% and 2.9×10^{-2}% for FFT, $FRFT_8$ and $FRFT_9$ respectively, whereas $FRFT_8$ is faster than $FRFT_9$ in a 49.36% and a 192.4% faster than DI form.

4.6 The Mikhailov and Nögel (2004) Model

This section does not include any analytic formula, since Greeks and other sensitivities for the Time-Dependent Heston model[5] have the same form as for the Heston model. The only difference is the way in which they have to been calculated, but it is straightforward once we know the final set of parameters $\Theta_N = \left(\kappa^N, \theta^N, \sigma^N, v_0^N, \rho^N\right)$, so they are not presented here.

The goal of the only part of this section is to compare the results obtained when we compute the Greeks and other sensitivities for the TD Heston model by means of methods as finite differences or Fourier algorithms. We will compare them in relation with the results provided by Direct Integration of semi-closed solution via Gauss-Laguerre quadrature.

4.6.1 Numerical Results

It will be presented here, the numerical results for Time-Dependent Heston model. Unlike of all previous model, in this section the results will be presented in a different way, since it has been opted for remove 3D visualization figures and in its place, graphics with CPU times have been included to analyse them for Greeks and other sensitivities. This choice has been motivated due to the fact that it has not sense consider a 3D figure, because we need to make several suppositions about the parameters or estimate the time-dependent parameters for each period, which is far of the scope of this Master Thesis. Instead of 3D visualization graphics, it has been though convenient introduce CPU times, since they will be able to help us with our times comparisons in the next analysis.

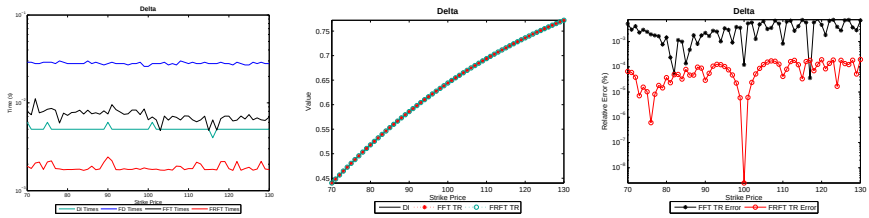
For this model, all the graphics and results showed in this section, corresponding to an European call option with the following conditions: $K = 100$, the static parameters $\theta = 0.1$, $\sigma = 0.2$, $v_0 = 0.1$ and $\rho = -0.3$, while κ varies from $\kappa = 1, 2, 4$ in the three periods, which we assume to be equal in length, considering maturity in this case and a difference of the previous models as $\tau = 5$. Furthermore, the model has been implemented considering $r = 0$, like Mikhailov and Nögel [MN04] made in table 1 of their article of 2004. At last, it has been considered the same spot range of $S_0 \in [70, 130]$.

[5] When we talk about the Time-Dependent Heston Model, we refer to the Mikhailov and Nögel Model.

On the other hand, Fourier algorithms have been implemented considering a dampening factor of $\alpha = 1.75$ and $N = 2^8$ integration points for both Fourier algorithms, whilst the upper limit of integration is 200 for FFT and 100 for FRFT algorithm.

In the next pages, it will be showed the table of results for Delta, Theta and Sigma only, whereas the remainder Greeks and other sensitivities will be showed graphically. As we have been doing until now, bidimensional cuts have been taken at maturity for the parameters indicated in before.

We begin this section showing the results for Delta Greek, which it is presented in figure (4.37)



(a) CPU times **(b)** Adjust **(c)** Errors

Figure 4.37: CPU times, adjustments and errors for Mikhailov and Nögel Delta

Figure (4.37a), shows a comparison among the different methods employed in order to compute Delta under TD Heston model. In this figure it can be observed that with the choice of parameters N and *uplimit*, we have that Fourier algorithms are both faster than FD method, concretely, in the FRFT case it is up to $1.43 \times 10^{-3}\%$ more than FD, but it is also, near to 198% times faster than DI form. Figure (4.37b) shows the adjustment of Delta Greek under Fourier algorithms via trapezoidal rule, whose errors in arithmetic mean can be noticed in figure (4.37c), where it can be observed that FRFT algorithm is two magnitude orders lower than FFT and FD method.

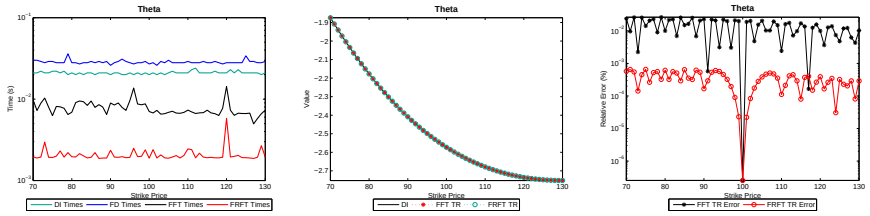
Table (4.13) shows the results of Delta at maturity for ATM options.

TD Heston Delta			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	0.6441	0.0000	0.0060
Finite Differences	0.6440	0.0022	0.0270
FFT Trapezoidal Rule	0.6441	0.0001	0.0050
FFT Simpson's Rule	0.6438	-0.0461	0.0060
FRFT Trapezoidal Rule	0.6441	0.0000	0.0010
FRFT Simpson's Rule	0.6441	0.0000	0.0020
$S_0 = 100,\ \theta = 0.1,\ \sigma = 0.2,\ v_0 = 0.1,\ \rho = -0.3$			
$\kappa_{\tau_1} = 4,\ \kappa_{\tau_2} = 2,\ \kappa_T = 1$			

Table 4.13: Results for Mikhailov and Nögel Delta.

The most significant result of table above is that FFT via Simpson's rule does not adjust Delta and as we will see later, any other sensitivity with the accuracy that might be able to expected. This a curiously fact, because FFT via Simpson's Rule algorithm is the same that it has been used for FRFT in this model and we can not find any explanation to this fact.

Theta will be the next Greek to be analyzed



(a) CPU times **(b)** Adjust **(c)** Errors

Figure 4.38: CPU times, adjustments and errors for Mikhailov and Nögel Theta

One relevant aspect it can be observed in figure (4.38a) and it is that ranking CPU mean times has changed in relation to Delta times ranking. FD method and FFT algorithm have swapped their positions now. CPU times relatives for DI, FD and FFT are respectively; 890%, 1.24×10^3% and 269% slower than FRFT algorithm. In relation to mean errors, they most accurate method is also FRFT, with an accuracy of 3.41×10^{-4}% in arithmetic mean, while errors for FD and FFT are respectively about 7.3×10^{-2}% and 7.6×10^{-3}%.

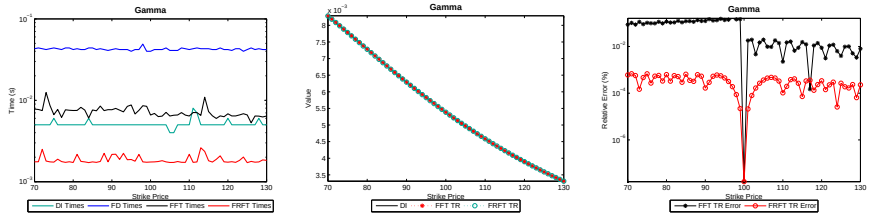
Next table shows results for ATM options.

TD Heston Theta			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	-2.5730	0.0000	0.0210
Finite Differences	-2.5750	-0.0780	0.0280
FFT Trapezoidal Rule	-2.5730	-0.0000	0.0050
FFT Simpson's Rule	-2.5728	-0.0082	0.0070
FRFT Trapezoidal Rule	-2.5730	-0.0000	0.0020
FRFT Simpson's Rule	-2.5730	-0.0000	0.0020
$S_0 = 100,\ \theta = 0.1,\ \sigma = 0.2,\ v_0 = 0.1,\ \rho = -0.3$			
$\kappa_{\tau_1} = 4,\ \kappa_{\tau_2} = 2,\ \kappa_T = 1$			

Table 4.14: Results for Mikhailov and Nögel Theta.

It can be noticed as FFT algorithm implemented via trapezoidal rule is by far, the worse Fourier algorithm as accuracy as concerned. Any other relevant aspect is remarkable in this table.

Gamma is presented in the next figure.



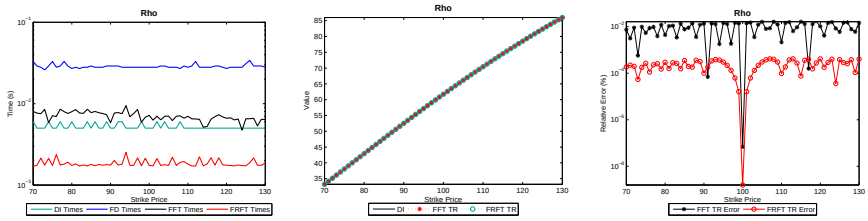
(a) CPU times **(b)** Adjust **(c)** Errors

Figure 4.39: CPU times, adjustments and errors for Mikhailov and Nögel Gamma

Here, we can find that CPU times ranking is the same than for Delta, being FRFT the fastest algorithm. Figure (4.39b) shows a different aspect to what we were used to see in the previous model, which it is consequence of the choice of parameters. Maybe, the most relevant aspect we can find it, when we analyze figure (4.39c), where it can be noticed that for OTM options, the adjustment via FFT is worse than ITM options, as direct consequence of FFT algorithm. It is important to point out that FRFT does not present this inconvenience, as it can see in the same graphic.

We will finish our study of TD Heston Greeks making a quick analysis for the remaining Greeks, Rho and Vega 1.

Rho will be commented in first place.

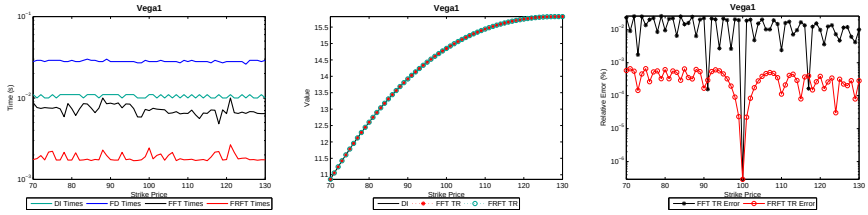


(a) CPU times **(b)** Adjust **(c)** Errors

Figure 4.40: CPU times, adjustments and errors for Mikhailov and Nögel Rho

It can be noticed that Rho shows similar characteristics to other Greeks commented previously. Figure (4.40a) is clear and figure (4.40c) show the error, which are in arithmetic mean; $473 \times 10^{-5}\%$, $9.7 \times 10^{-3}\%$ and $2.38 \times 10^{-4}\%$ for FD, FFT and FRFT respectively. This time, FD is more accurate than Fourier method. By contrast, it is three magnitude orders slower than FRFT.

Vega 1 is the last Greek that will be commented.

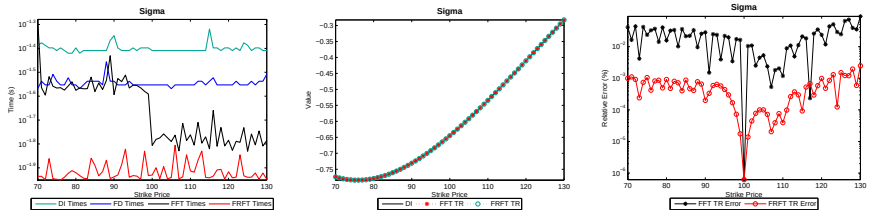


(a) CPU times **(b)** Adjust **(c)** Errors

Figure 4.41: CPU times, adjustments and errors for Mikhailov and Nögel Vega 1

This Greek does not present any interesting result that we have not commented before. Therefore, it will only given the results for figure (4.41c), since figure (4.41a) is clear. Therefore, errors in arithmetic mean are; $1.47 \times 10^{-4}\%$, $1.4 \times 10^{-2}\%$ and $3.41 \times 10^{-4}\%$ for FD, FFT and FRFT respectively. We can find again, as FRFT is the most convenient algorithm, due to it is the most accurate and fastest among all.

We conclude this section showing other sensitivities, as Kappa, Sigma and Vega 2. In this section, Sigma has been chosen to be analyzed together its table of results, so we will begin analyzed it in first place.



(a) CPU times **(b)** Adjust **(c)** Errors

Figure 4.42: CPU times, adjustments and errors for Mikhailov and Nögel Sigma

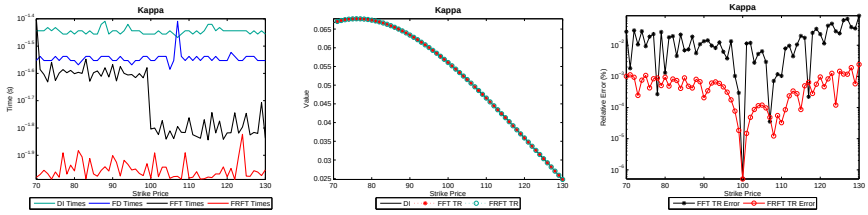
Sigma has been calculated via symbolic derivatives, so it can explain the results showed in figure (4.42a), where the most relevant aspect is that CPU times for FFT in the OTM options side is slower than CPU times for FFT in the ITM options side. CPU times are all the same order, 10^{-2}s, so choosing one or another method has to be conditioned by the mean error in the adjustment. Figure (4.42c) shows that mean error is about 65% for FD method, while Fourier methods presented errors in arithmetic mean under 10^{-2}% for the FFT case and under 10^{-4}% for FRFT case. We do not understand why FD errors are so higher in this case, since derivatives have been calculated symbolically and the method has been implemented in the same way that the models before.

TD Heston Sigma			
Method	**Value**	**Error (%)**	**Time (s)**
Closed Form	-0.6448	0.0000	0.0380
Finite Differences	-0.6521	-11.3213	0.0270
FFT Trapezoidal Rule	-0.6448	-0.0000	0.0160
FFT Simpson's Rule	-0.6450	0.0281	0.0170
FRFT Trapezoidal Rule	-0.6448	-0.0000	0.0110
FRFT Simpson's Rule	-0.6448	-0.0000	0.0110
$S_0 = 100$, $\theta = 0.1$, $\sigma = 0.2$, $v_0 = 0.1$, $\rho = -0.3$			
$\kappa_{\tau_1} = 4$, $\kappa_{\tau_2} = 2$, $\kappa_T = 1$			

Table 4.15: Results for Mikhailov and Nögel Sigma.

ATM options always presents the best adjustment and in spite of, error made in the calculation this Greek is about 11%, which is really large. Notice, that FFT Simpson's Rule continues presenting the worse adjust among Fourier methods.

Kappa is the other sensitivity that has been calculated symbolically. We will see now how it has affected to Kappa.

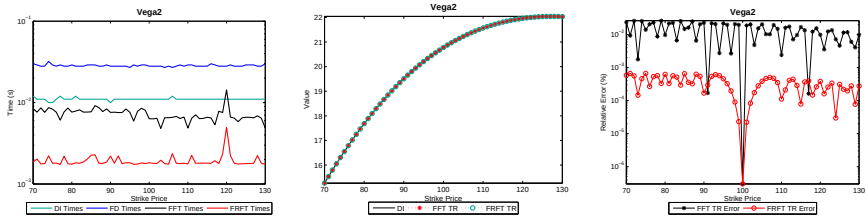


(a) CPU times **(b)** Adjust **(c)** Errors

Figure 4.43: CPU times, adjustments and errors Mikhailov and Nögel Kappa

Figures (4.43a) and (4.43c) are very similar to the figures (4.42a) and (4.42c) respectively, but in a shorter scale, so the same comments made before are valid here. As results, notice that mean error for FD is about 30%, while FFT presented a mean error about 10^{-2}% and FRFT an erro of 10^{-4}%.

This section conclude with the results for Vega 2.



(a) CPU times **(b)** Adjust **(c)** Errors

Figure 4.44: CPU times, adjustments and errors for Mikhailov and Nögel Vega 2

Ranking of CPU times is the same than Vega 1 ranking and it can be observed in figure (4.44a). Figure (4.44c) also shows the same aspect than error in Vega 1, but in a shorter scale. This time, mean errors in arithmetic mean are the same order order, 10^{-4}% for FD and FRFT, while FFT reachs an accuracy of 10^{-2}%.

5

Conclusions and Outlook

This chapter draw the conclusions and discuss other alternatives to Fourier algorithms within Quantitative Finances

Through previous chapters, it can be checked as algorithms based on Fourier transforms are much more efficiencies than methods as Monte Carlo when we prices options or as finite differences when we calculate greeks or other sensitivities. By efficient we refer to the fact that these algorithms are more accurate and faster than Monte Carlo and sometimes it is also true as regards to finite differences method.

It is also important to indicate that for a similar temporal magnitude order, Fourier methods provide at the same time prices for about 2^{11} strikes, while Monte Carlo simulations, provide only a single strike price. Here is the great advantage of Fourier methods over Monte Carlo simulations. So all the analyses performed represent a valid comparison in the only case that we are interested in knowing the price for a given strike, since otherwise, Fourier methods are much more powerful.

We have also made our study considering two Fourier algorithms implemented via trapezoidal or Simpson's rules and between them, we have checked that FRFT is usually faster than FFT and sometimes, it is also true when we compared FRFT algorithm with the integration of semi-closed solution via Gauss-Laguerre quadrature It is also remarkable, that there are many cases for which FRFT is a better alternative than FFT once we have suitably selected the parameters η and λ. The most of all our comparison have this fact into account and prove and reinforce the use of FRFT algorithm over FFT.

On the other hand, it is important to point out that main drawbacks of the Fourier algorithms are that they force to log-strikes to fall inside integration grid, so the methodology has to be adapted to this situation or if not, the methods are limited to pricing only options whose corresponding log-strike prices fall on that grid. To solve this situation, we have designed the first alternative and we have used and interpolation scheme. This is the reason for which sometimes accuracy order is not sufficiently good. Another drawback of Fourier algorithms is that the value of N must always be a power of 2, but the main problem arises when we need to price exotic options. In general in these cases, Fourier methods does not provide us with a solution, so we need to revert to Monte Carlo methods.

Finally, we conclude with some comments about the alternatives to Fourier methods to option pricing. These alternative consider other transforms, as it can be read in [Lin11], where Lin devoted its study to the Hilbert transform and its applications in computational Finance or [FO08], where Fang et al. develop an option pricing method for European options based on the Fourier-cosine series, and call it the COS method. However, noticed that both methods consider also Fourier techniques inside.

A

Mean Errors and Times for the Heston Model

Contents:

Appendix A.1 Detailed Tables and figures for the Heston Model

In this appendix, more detailed tables and figures corresponding to other simulation for the Heston Model can be downloaded from here 

B

Mean Errors and Times for the Bates Model

Contents:

Appendix B.1 Detailed Tables and figures for the Bates Model

In this appendix, more detailed tables and figures corresponding to other simulation for the Bates Model can be downloaded from here 

C

Mean Errors and Times for the SVJJ Model

Contents:

Appendix C.1 Detailed Tables and figures for the SVJJ Model

In this appendix, more detailed tables and figures corresponding to other simulation for the SVJJ Model can be downloaded from here 

D

Mean Errors and Times for the Double Heston Model

Contents:

Appendix D.1 Detailed Tables and figures for the Double Heston Model

In this appendix, more detailed tables and figures corresponding to other simulation for the Double Heston Model can be downloaded from here 

E

Mean Errors and Times for the Mikhailov and Nögel Model

Contents:

Appendix E.1 Detailed Tables and figures for the Mikhailov and Nögel Model

In this appendix, more detailed tables and figures corresponding to other simulation for TD-Heston Model can be downloaded from here 

F

Alternative Methodology for Greeks and other sensitivities

Contents:

Appendix F.1 Carr-Madan Formulation using FFT and FRFT

Following to [Rou13], it is possible to derive semi-closed formulas for greeks under Carr and Madan representation and compare the accuracy of the greeks obtained under this model and the computation time involved in the calculations. If we recall the equation (1.13) of Carr & Madan

$$c_T(k) = \frac{e^{-\alpha k}}{\pi} \int_0^\infty Re\left\{ e^{-ivk} \frac{e^{-r\tau}\phi_T\left[v-(\alpha+1)i\right]}{\alpha^2+\alpha-v^2+i(2\alpha+1)v} \right\} dv$$

Then, it is very simply show that greeks under this formulation are:

$$\begin{aligned} \Delta_{CM} &= \frac{\partial c_T(k)}{\partial S} \\ &= \frac{e^{-\alpha k}}{\pi S} \int_0^\infty Re\left\{ e^{-ivk} \frac{e^{-r\tau} iv\phi_T\left[v-(\alpha+1)i\right]}{\alpha^2+\alpha-v^2+i(2\alpha+1)v} \right\} dv \end{aligned} \tag{F.1}$$

$$\begin{aligned} \Gamma_{CM} &= \frac{\partial^2 c_T(k)}{\partial S^2} \\ &= \frac{e^{-\alpha k}}{\pi S^2} \int_0^\infty Re\left\{ e^{-ivk} \frac{e^{-r\tau}(iv-1)\phi_T\left[v-(\alpha+1)i\right]}{\alpha^2+\alpha-v^2+i(2\alpha+1)v} \right\} dv \end{aligned} \tag{F.2}$$

$$\begin{aligned} \Theta_{CM} &= -\frac{\partial c_T(k)}{\partial \tau} \\ &= \frac{e^{-\alpha k}}{\pi} \int_0^\infty Re\left\{ e^{-ivk} \frac{e^{-r\tau}\left(r\phi+\partial\phi/\partial\tau\right)}{\alpha^2+\alpha-v^2+i(2\alpha+1)v} \right\} dv \end{aligned} \tag{F.3}$$

$$\begin{aligned} \mathcal{V}^1_{CM} &= \frac{\partial c_T(k)}{\partial v} \\ &= \frac{e^{-\alpha k}}{\pi} \int_0^\infty Re\left\{ e^{-ivk} \frac{e^{-r\tau} D(\tau,v)\phi_T\left[v-(\alpha+1)i\right]}{\alpha^2+\alpha-v^2+i(2\alpha+1)v} \right\} dv \end{aligned} \tag{F.4}$$

And now it is possible adapt these expressions to implement the FFT and FRFT algorithms. We need construct the integration grid $\{v_j\}_{j=1}^N$, the log-strike grid $\{k_u\}_{u=1}^N$, and to calculate the points $x_j = exp\left[i\left(b-\ln(S_t)\right)v_j\right]\psi(v_j)w_j$ for $j = 1,\ldots,N$. We consider now that

$$\psi(v_j) = \frac{e^{-r\tau}\phi_T\left[v_j-(\alpha+1)i\right]}{\alpha^2+\alpha-v_j^2+i(2\alpha+1)v_j}$$

where now, ϕ corresponding to derivative of the second[1] characteristic function for the desired greek using the expression calculated in 3.

[1] For the Double Heston model we have only one characteristic function, so that in this case, we referring it as the derivative of the cf, instead of the second cf.

With this, we obtain $\hat{x}_u = C(k_u)$, and the value of the greek is given by

$$\hat{x}_u = \frac{\eta e^{-\alpha k_u}}{\pi} \sum_{j=1}^{N} Re\left\{e^{-i\frac{2\pi}{N}(j-1)(u-1)} x_j\right\} \qquad \text{for u = 1,...,N} \quad \text{(F.5)}$$

G

Final Presentation

Contents:

Appendix G.1 The beamer presentation of the Master Thesis (10/07/2014)

This Master Thesis was defended at Thursday, July 10th, 2014 and the final presentation can be download from here: 

Bibliography

[Bat96] David S Bates. Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *Review of financial studies*, 9(1):69–107, 1996.

[CHJ09] Peter Christoffersen, Steven Heston, and Kris Jacobs. The shape and term structure of the index option smirk: Why multifactor stochastic volatility models work so well. *Management Science*, 55(12):1914–1932, 2009.

[Cho04] Kyriakos Chourdakis. Option pricing using the fractional fft. *Journal of Computational Finance*, 8(2):1–18, 2004.

[CIJR85] John C Cox, Jonathan E Ingersoll Jr, and Stephen A Ross. A theory of the term structure of interest rates. *Econometrica: Journal of the Econometric Society*, pages 385–407, 1985.

[CM99] Peter Carr and Dilip Madan. Option valuation using the fast fourier transform. *Journal of computational finance*, 2(4):61–73, 1999.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Math. comput*, 19(90):297–301, 1965.

[DPS00] Darrell Duffie, Jun Pan, and Kenneth Singleton. Transform analysis and asset pricing for affine jump-diffusions. *Econometrica*, 68(6):1343–1376, 2000.

[FO08] Fang Fang and Cornelis W Oosterlee. A novel pricing method for european options based on fourier-cosine series expansions. *SIAM Journal on Scientific Computing*, 31(2):826–848, 2008.

[FR08] Gianluca Fusai and Andrea Roncoroni. *Implementing models in quantitative finance: methods and cases*, volume 1. Springer Berlin, 2008.

[GP51] J Gil-Pelaez. Note on the inversion theorem. *Biometrika*, 38(3-4):481–482, 1951.

[Hes93] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of financial studies*, 6(2):327–343, 1993.

[KP92] Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer, 1992.

[Lin11] Xiong Lin. *The Hilbert Transform and its Applications in Computational finance*. PhD thesis, University of Illinois at Urbana-Champaign, 2011.

[LK06] Roger Lord and Christian Kahl. Optimal fourier inversion in semi-analytical option pricing. Technical report, Tinbergen Institute Discussion Paper, 2006.

[MN04] Sergei Mikhailov and Ulrich Nogel. Heston ś stochastic volatility model: Implementation, calibration and some extensions. *Wilmott magazine*, 2004.

[Ng05] ManWo Ng. *Option Pricing via the FFT*. PhD thesis, Master Thesis, Applied Institute of Mathematics, TU Delft, 2005.

[PK12] Warrick Poklewski-Koziell. *Stochastic volatility models: calibration, pricing and hedging*. PhD thesis, 2012.

[Rou13] Fabrice D Rouah. *The Heston Model and Its Extensions in Matlab and C#*. John Wiley & Sons, 2013.

[Zhu09] Jianwei Zhu. *Applications of Fourier Transform to Smile Modeling: Theory and Implementation*. Springer, 2009.