วิธีเชิงตัวเลขสำหรับแบบจำลองของคอกซ์-อินเกอซอล-รอสส์และแบบจำลองของความแปรปรวนที่ความยืดหยุ่นคงตัวที่ถูกขยายโดยการกระโดด

นายภูรินทร์ กลั่นกล้า

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562

NUMERICAL METHODS FOR JUMP-EXTENDED COX-INGERSOLL-ROSS AND CONSTANT ELASTICITY OF VARIANCE MODELS

Mr. Purin Klunklar

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Applied Mathematics and

Computational Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2019

Thesis Title	NUMERICAL METHODS FOR JUMP-EXTENDED COX-INGERSOLL-ROSS AND CONSTANT ELASTICITY OF VARIANCE MODELS
By	Mr. Purin Klunklar
Field of Study	Applied Mathematics and Computational Science
Thesis Advisor	Associate Professor Petarpa Boonserm, Ph.D.
Thesis Co-advisor	Raywat Tanadkithirun, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

. . . . Dean of the Faculty of Science

(Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

. . . . Chairman

(Associate Professor Khamron Mekchay, Ph.D.)

. . . . Thesis Advisor

(Associate Professor Petarpa Boonserm, Ph.D.)

. . . . Thesis Co-advisor

(Raywat Tanadkithirun, Ph.D.)

. . . . Examiner

(Associate Professor Ratinan Boonklurb, Ph.D.)

. . . . External Examiner

(Sirod Sirisup, Ph.D.)

ภูรินทร์ กลั่นกล้า : วิธีเชิงตัวเลขสำหรับแบบจำลองของคอกซ์-อินเกอซอล-รอสส์และแบบจำลองของความแปรปรวนที่ความยืดหยุ่นคงตัวที่ถูกขยายโดยการกระโดด. (NUMERICAL METHODS FOR JUMP-EXTENDED COX-INGERSOLL-ROSS AND CONSTANT ELASTICITY OF VARIANCE MODELS) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร. เพชรอาภา บุญเสริม, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม : อ.ดร. เรวัต ถนัดกิจหิรัญ 71 หน้า.

แบบจำลองคอกซ์-อินเกอซอล-รอสส์ และแบบจำลองของความแปรปรวนที่มีความยืดหยุ่นคงตัวที่ถูกขยายโดยการกระโดด เป็นแบบจำลองที่นิยมใช้ในการทำนายอัตราดอกเบี้ยหรือราคาหุ้น ในงานนี้ เราได้ทำการหาคำตอบของแบบจำลองข้างต้นแบบทางตรงด้วยระเบียบวิธีคำนวณเชิงตัวเลขแปดวิธี ประกอบไปด้วย ออยเลอร์มารุยามะ ออยเลอร์มารุยามะอย่างง่าย ออยเลอร์มารุยามะที่มีการดัดแปลงการกระโดด ออยเลอร์มารุยามะที่มีการดัดแปลงการกระโดดอย่างง่าย อันดับสองแบบอ่อนที่มีการดัดแปลงการกระโดด อันดับสองแบบอ่อนที่มีการดัดแปลงการกระโดดอย่างง่าย อันดับสองแบบไม่มีอนุพันธ์ที่มีการดัดแปลงการกระโดด และอันดับสองแบบไม่มีอนุพันธ์ที่มีการดัดแปลงกรกระโดดอย่างง่าย โดยในงานนี้ เราสนใจในวิธีการแปลงจากแปดระเบียบวิธีคำนวณเชิงตัวเลขด้วย เราได้ทำการเปรียบเทียบประสิทธิภาพของระเบียบวิธีคำนวณเชิงตัวเลขโดยการทดสอบความเป็นบวกของคำตอบเชิงตัวเลข หาอันดับการลู่เข้าแบบอ่อน และระยะเวลาการคำนวณของแต่ละวิธี

ภาควิชา	คณิตศาสตร์และ	ลายมือชื่อนิสิต ....
	วิทยาการคอมพิวเตอร์	ลายมือชื่อ อ.ที่ปรึกษาหลัก ....
สาขาวิชา	คณิตศาสตร์ประยุกต์	ลายมือชื่อ อ.ที่ปรึกษาร่วม ....
	และวิทยาการคณนา	
ปีการศึกษา	2562	

6071984923 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE
KEYWORDS : JUMP-EXTENDED CIR AND CEV MODELS / JUMP-ADAPTED METHOD, WEAK ORDER OF CONVERGENCE / ITÔ TRANSFORMATION

PURIN KLUNKLAR : NUMERICAL METHODS FOR JUMP-EXTENDED COX-INGERSOLL-ROSS AND CONSTANT ELASTICITY OF VARIANCE MODELS. ADVISOR : ASSOC. PROF. PETARPA BOONSERM, Ph.D., CO-ADVISOR : RAYWAT TANADKITHIRUN, Ph.D., 71 pp.

The jump-extended Cox-Ingersoll-Ross and jump-extended constant elasticity of variance models are stochastic differential equations (SDEs) used to forecast interest rates or stock prices. We simulate these SDEs directly by eight numerical methods: Euler Maruyama method, simplified Euler method, jump-adapted Euler method, jump-adapted simplified Euler method, jump-adapted order two weak method, jump-adapted simplified order two weak method, jump-adapted order two derivative free method and jump-adapted simplified order two derivative free method. The transformed approach is also applied with these eight numerical methods. We compare their performance by testing the positivity preserving of numerical solutions and finding their weak orders of convergence as well as their run time.

Department :	Mathematics and Computer Science	Student's Signature ....
		Advisor's Signature ....
Field of Study :	Applied Mathematics and Computational Science	Co-advisor's Signature ....
Academic Year :	2019	

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my dissertation advisor, Associate Professor Dr. Petarpa Boonserm and my coadvisor Dr. Raywat Tanadkithirun for the continued support on my study in the master's degree. Their encouragement, motivation and guidance helped me during the time for writing researches and this dissertation until it was accomplished. I further would like to thank all of my dissertation committees: Associate Professor Dr. Khamron Mekchay, Associate Professor Dr. Ratinan Boonklurb and Dr. Sirod Sirisup, for their insightful comments and suggestions which motivated me to extend my research from various perspectives. Moreover, I would like to express my special appreciation and thanks to my financial sponsors, "Development and Promotion of Science and Technology Talents Project (DPST)" for the scholarship and funding to present my research on international conferences on applied mathematics and computational sciencs. My sincere thanks also go to the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University which give me an opportunity that I received throughout my graduate studies. Finally, I would like to thank my family for supporting me throughout writing this dissertation. Also, I wish to express my gratitude to all friends and colleagues, who stayed with me and provided their encouragement, relaxation, great suggestions and supports in many ways during a hard time studying in my master degree.

CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In 1976, Cox and Ross [3] presented constant elasticity of variance (CEV) model to forecast stock prices at time t. The CEV model has the form

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t^{\gamma} \mathrm{d}W_t,$$

where S_t is the value of the stock at time t, μ is a parameter characterising the drift, σ^2 is the instantaneous variance of the return, γ is the elasticity parameter of the local volatility and W_t is a Wiener process. To get a more realistic stochastic differential equation (SDE) model, a jump process should be added into the model. In the same year, Merton [7] introduced an SDE with jumps driven by a Poisson process which has the form

$$\mathrm{d}S_t = (\alpha - \widetilde{\lambda}\kappa) S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t + S_t \mathrm{d}N_t,$$

where α is the instantaneous expected return on the stock, σ^2 is the instantaneous variance of the return, $\widetilde{\lambda}$ is the mean number of arrivals per unit time, N_t is a Poisson process with rate λ independent of W_t and κ is the random variable representing percentage change in the stock price if the Poisson event occurs. In 1985, Cox, Ingersoll and Ross [2] proposed Cox-Ingersoll-Ross (CIR) model to predict an interest rate. The model has the form

$$\mathrm{d}r_t = \kappa(\theta - r_t)\mathrm{d}t + \sigma\sqrt{r_t}\mathrm{d}W_t,$$

where r_t is the value of interest rate at time t, κ is the rate of convergence of the process, θ is the long run mean for the process and σ^2 is the instantaneous variance of the return.

In 2012, Beliava and Nawalkha [1] proposed a jump-extended CEV model

$$\mathrm{d}r_t = \kappa(\theta - r_t)\mathrm{d}t + \sigma r_t^p \mathrm{d}W_t + f(r_t, J)\mathrm{d}N_t,$$

where $p \geq \frac{1}{2}$ and the function $f(r_t, J)$ depends on r_t and the jump variable J. They focused on two specific type of jumps. The first one is $f(r_t, J) = J$ where J is exponentially distributed with mean $\frac{1}{\lambda}$ and the second one is $f(r_t, J) = r_t\left(e^J - 1\right)$ where J is normally distributed with mean $\widetilde{\mu}$ and variance $\widetilde{\sigma}^2$. In 2017, Yang and Wang [12] suggested a transformed jump-adapted backward Euler method to apply with jump-extended CIR and CEV models

$$\mathrm{d}X_t = \kappa\left(\theta - X_{t-}\right)\mathrm{d}t + \sigma X_{t-}^{\alpha}\mathrm{d}W_t + \gamma X_{t-}\mathrm{d}N_t, \tag{1.1}$$

where X_t is the target stochastic process, $X_{t-} = \lim\limits_{s \to t-} X_s$, α is the parameter that controls the effect of the current value of the process to its variation and γ is the parameter that controls the size of jumps. If $\alpha = \frac{1}{2}$, then it is called jump-extended CIR (JCIR) model. If $\alpha \in \left(\frac{1}{2}, 1\right)$, then it is called jump-extended CEV (JCEV) model. They transformed the SDE (1.1) into another SDE via the transformation $Y_t = X_t^{1-\alpha}$ using the Itô formula with jumps. The transformed SDE has the form

$$\mathrm{d}Y_t = f_\alpha(Y_{t-})\mathrm{d}t + (1-\alpha)\sigma\mathrm{d}W_t + \left[\left(Y_{t-}^{\frac{1}{1-\alpha}} + \gamma Y_{t-}^{\frac{1}{1-\alpha}}\right)^{1-\alpha} - Y_{t-}\right]\mathrm{d}N_t, \tag{1.2}$$

with

$$f_\alpha(y) = (1-\alpha)\left(\kappa\theta y^{\frac{-\alpha}{1-\alpha}} - \kappa y - \frac{\alpha\sigma^2}{2}y^{-1}\right)$$

for $\alpha \in \left[\frac{1}{2}, 1\right)$. They claimed that their transformation secures the positivity preserving of the solution for the SDEs. In 2007, Bruti-Liberati and Platen [6] presented a survey paper for a bunch of numerical schemes used to solve SDEs with jumps in both strong and weak senses. They applied those schemes to the SDEs with jumps in [9]. However, the jump process used in their work is only a Poisson process.

In our research, we focus on JCIR and JCEV models when the jump process is a compound Poisson process. Our JCIR and JCEV models have the form

$$\mathrm{d}X_t = \kappa(\theta - X_{t-})\mathrm{d}t + \sigma X_{t-}^{\alpha}\mathrm{d}W_t + \gamma X_{t-}\mathrm{d}J_t, \tag{1.3}$$

where X_t is the target stochastic process, $X_{t-} = \lim_{s \to t-} X_s$, κ is the rate of convergence of the process, θ is the long run mean for the process, σ^2 is the instantaneous variance of the return, α is the parameter that controls the effect of the current value of the process to its variation, W_t is a Wiener process, γ is the parameter that controls the size of jumps and J_t is a compound Poisson process with intensity λ and shifted lognormal jump size distribution H with parameters $\widetilde{\mu}$, $\widetilde{\sigma}^2$ and the shift of size one, i.e., $\log(H+1) \sim \mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2)$. From now on, we denote this jump size distribution by $H \sim SLog\mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2, 1)$. The SDE (1.3) is actually the JCEV model in [1] with $H \sim SLog\mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2, 1)$. Here, we assume that $X_0 > 0$, $\gamma > 0$ and one of the following two conditions holds: (i) $\alpha = \frac{1}{2}$, κ, θ, $\sigma > 0$ and $2\kappa\theta > 0$; (ii) $\alpha \in \left(\frac{1}{2}, 1\right)$, κ, θ, $\sigma > 0$. These conditions ensure the regularity and moment conditions [1, 5].

We will transform (1.3) via the transformation $f(t, X_t) = X_t^{1-\alpha}$, which is the suggested transformation in [1], using the Itô formula with jumps in [10]. We choose seven numerical methods from [6] and another modified method to solve for numerical solutions of (1.3) in both direct and transformed approaches. Then, we compare their efficiency by testing their positivity preserving of the numerical solutions, finding weak orders of convergence and run time.

CHAPTER II

BACKGROUND KNOWLEDGE

In this chapter, we present basic knowledge about some important distributions and processes, SDEs with jumps, Itô formula and numerical methods which we use in this work.

2.1 Basic knowledge

2.1.1 Normal distribution

A random variable X is said to have a normal distribution with parameters $\widetilde{\mu} \in \mathbb{R}$ and $\widetilde{\sigma}^2 > 0$, denoted by $X \sim \mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2)$, if its probability density function (PDF) is given by

$$f(x \mid \widetilde{\mu}, \widetilde{\sigma}^2) = \frac{1}{\widetilde{\sigma}\sqrt{2\pi}} e^{-\frac{(x-\widetilde{\mu})^2}{2\widetilde{\sigma}^2}}, \qquad \text{for all } x \in \mathbb{R}.$$

If $X \sim \mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2)$, then $\mathrm{E}[X] = \widetilde{\mu}$ and $\mathrm{Var}[X] = \widetilde{\sigma}^2$. Fig. 2.1 shows the PDF of the normal distribution with parameters $\widetilde{\mu} = 0$ and $\widetilde{\sigma}^2 = 1$.

2.1.2 Lognormal distribution

A random variable X is said to have a lognormal distribution with parameters $\widetilde{\mu} \in \mathbb{R}$ and $\widetilde{\sigma}^2 > 0$, denoted by $X \sim Log\mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2)$ or $\ln(X) \sim \mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2)$, if its PDF is given by

$$f(x \mid \widetilde{\mu}, \widetilde{\sigma}^2) = \frac{1}{\widetilde{\sigma}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\ln(x)-\widetilde{\mu}}{\widetilde{\sigma}}\right)^2}, \qquad \text{for all } x > 0$$

If $X \sim Log\mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2)$, then $\mathrm{E}[X] = e^{\widetilde{\mu}+\frac{1}{2}\widetilde{\sigma}^2}$ and $\mathrm{Var}[X] = e^{2\widetilde{\mu}+\widetilde{\sigma}^2}\left(e^{\widetilde{\sigma}^2} - 1\right)$.

In this work, we focus on a shifted lognormal distribution with shift 1 called H denoted by $H \sim SLog\mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2, 1)$ or $\ln(H+1) \sim \mathcal{N}\left(\widetilde{\mu}, \widetilde{\sigma}^2\right)$. Fig. 2.2 shows the PDFs of

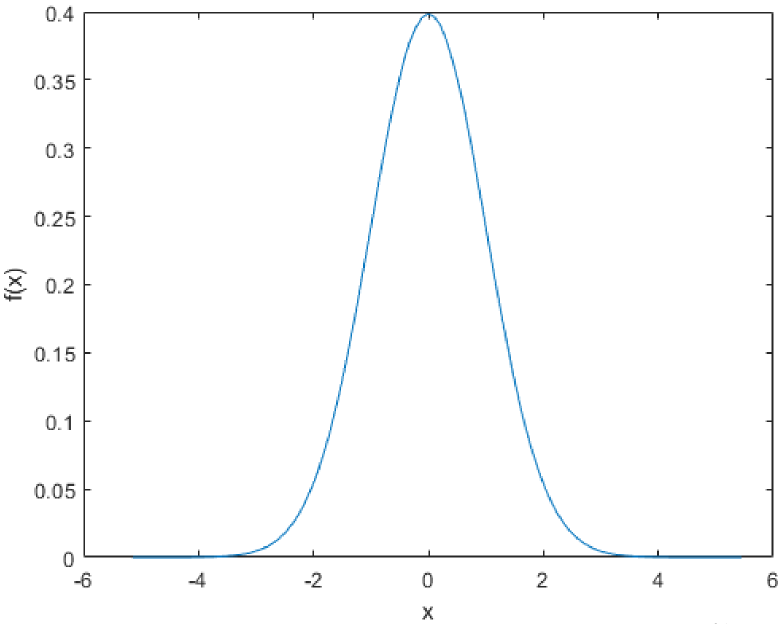


Figure 2.1: The PDF of normal distribution with parameters $\widetilde{\mu} = 0$, and $\widetilde{\sigma}^2 = 1$

$Log\mathcal{N}(0, 0.1)$ and $SLog\mathcal{N}(0, 0.1, 1)$.

2.1.3 Poisson distribution

A random variable X is said to have a Poisson distribution with parameter $\lambda > 0$, denoted by $X \sim \text{Poi}(\lambda)$, if its probability mass function (PMF) is given by

$$f(x \mid \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}, \quad \text{for all } x \in \mathbb{N} \cup \{0\}.$$

If $X \sim \text{Poi}(\lambda)$, then $\text{E}[X] = \text{Var}[X] = \lambda$. Fig. 2.3 shows the PMF of the Poisson distribution with parameter $\lambda = 5$ on $\{0, 1, 2, ..., 10\}$.

2.1.4 Wiener process

A stochastic process is a collection of random variables. The Wiener process $\{W_t\}_{t\in[0,T]}$ is a stochastic process characterized by the following properties.

1. $W_0 = 0$ almost surely.

2. W_t has independent increments, i.e., for every $0 \le s < t \le u < v \le T$, the increments $W_t - W_s$ and $W_v - W_u$ are independent.

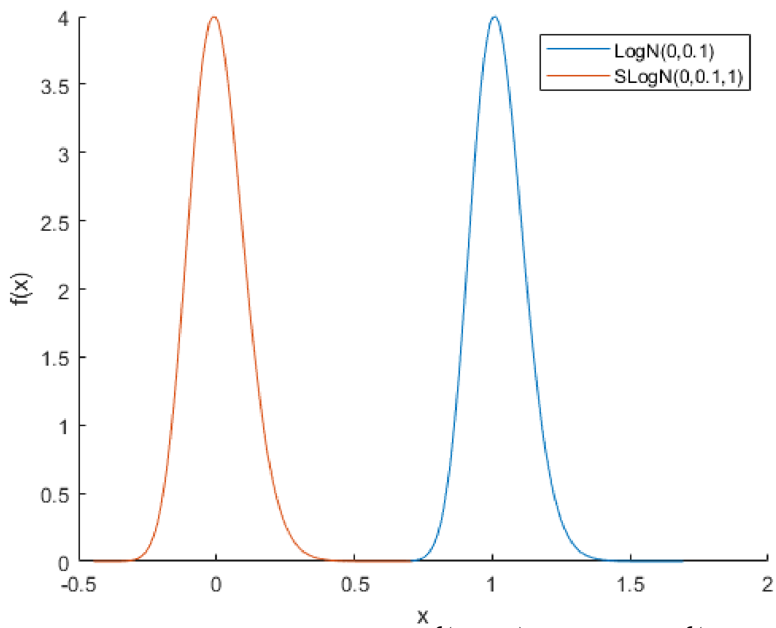


Figure 2.2: The PDFs of $Log\mathcal{N}(0, 0.1)$ and $SLog\mathcal{N}(0, 0.1, 1)$

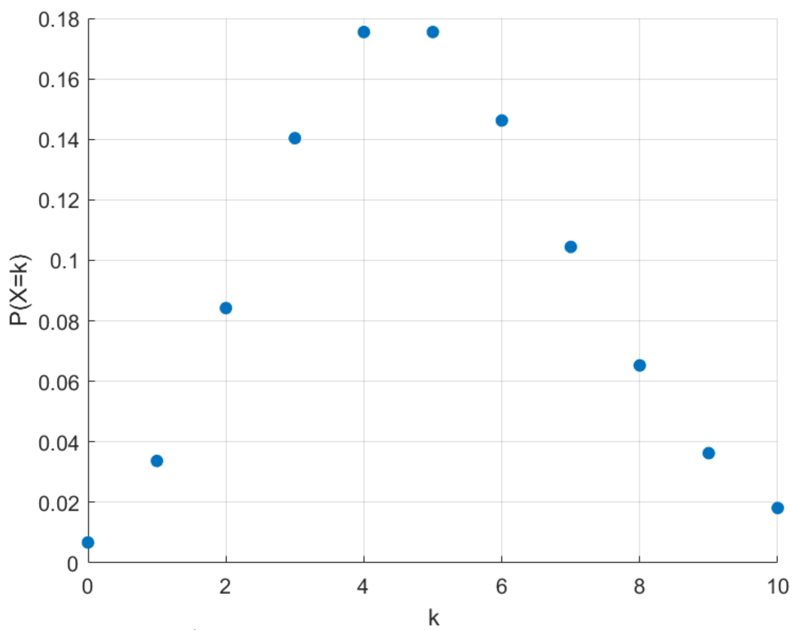


Figure 2.3: The PMF of Poisson distribution with parameters $\lambda = 5$ on $\{0, 1, 2, ..., 10\}$

3. For $0 \leq s < t \leq T$, $W_t - W_s \sim \mathcal{N}(0, t-s)$.

4. W_t has continuous sample paths.

2.1.5 Poisson process

A counting process $\{C_t\}_{t\in[0,T]}$ is a stochastic process characterized by the following properties.

1. C_t is a non-negative integer for each t.

2. C_t is non-decreasing in t.

3. C_t is right continuous.

A Poisson process $\{N_t\}_{t\in[0,T]}$ with intensity $\lambda > 0$ is a counting process characterized by the following properties.

1. $N_t = 0$.

2. It has independent increments, i.e., for every $0 \leq s < t \leq u < v \leq T$, the increments $N_t - N_s$ and $N_v - N_u$ are independent.

3. The number of events in any interval of length τ has a Poisson distribution with intensity $\lambda\tau$, i.e., for $0 \leq s < t \leq T$, the increment $N_t - N_s \sim \mathrm{Poi}(\lambda(t-s))$.

A waiting time τ between consecutive jumps of a Poison process with intensity λ is exponentially distributed with mean $\frac{1}{\lambda}$ and the PDF is

$$f(x \mid \lambda) = \begin{cases} \lambda e^{-\lambda x}, & \text{for } x > 0. \\ 0, & \text{for } x \leq 0. \end{cases}$$

2.1.6 Compound Poisson process

A compound Poisson process $\{J_t\}_{t\in[0,T]}$ is defined by

$$J_t = \sum_{i=1}^{N_t} \xi_i,$$

where $\{N_t\}_{t\in[0,T]}$ is a Poisson process with intensity $\lambda > 0$ and $\{\xi_i\}_{i\in\mathbb{N}}$ is a sequence of independent and identically distributed (i.i.d.) random variables representing the corresponding jump sizes which has a common distribution $\mathcal{D}$.

2.2 SDE with jumps

In our work, we consider SDEs with jumps in the form

$$\mathrm{d}X_t = a(t, X_t)\mathrm{d}t + b(t, X_t)\mathrm{d}W_t + c(t, X_{t-})\mathrm{d}J_t, \tag{2.1}$$

where the actual meaning is an integral equation

$$X_t = X_0 + \int_0^t a(s, X_s)\mathrm{d}s + \int_0^t b(s, X_s)\mathrm{d}W_s + \int_0^t c(s, X_{s-})\mathrm{d}J_s,$$

where a, b and c are functions of two variables, W_t is a Wiener process and J_t is a compound Poisson process. The integral $\int_0^t a(s, X_s)\mathrm{d}s$ is interpreted in the sense of Riemann integral [11], the integral $\int_0^t b(s, X_s)\mathrm{d}W_s$ is an Itô integral [4] and the last integral $\int_0^t c(s, X_{s-})\mathrm{d}J_s$ can be written as $\sum_{i=N_{t_n}+1}^{N_{t_{n+1}}} c(\tau_i, X_{\tau_i -})\xi_i$ where N_t is a Poisson process with intensity λ , τ_i's are jump time and ξ_i's have a common distribution $\mathcal{D}$. Here, functions a, b and c must satisfy the regularity condition in order to make these integrals well-defined [8].

2.3 Itô formula

Consider the SDE with jumps

$$\mathrm{d}X_t = a(t, X_{t^-})\mathrm{d}t + b(t, X_{t^-})\mathrm{d}W_t + c(t, X_{t^-})\mathrm{d}J_t, \tag{2.2}$$

where J_t is a compound Poisson process with a corresponding Poisson process N_t and a sequence of jump sizes $\{\xi_i\}_{i\in\mathbb{N}}$. If f is a twice continuously differentiable function, then the Itô formula for $f(X_t)$ is given by [10]

$$f(X_t) = f(X_0) + \int_0^t b(s,X_s)f'(X_s)\mathrm{d}W_s + \frac{1}{2}\int_0^t f''(X_s)b^2(s,X_s)\mathrm{d}s \qquad (2.3)$$
$$+ \int_0^t a(s,X_s)f'(X_s)\mathrm{d}s + \int_0^t \left(f\left(X_{s-} + c(s,X_s)\xi_{N_s}\right) - f(X_{s-})\right)\mathrm{d}N_s.$$

Proposition 2.3.1. [10] Let $(\phi_t)_{t\in\mathbb{R}^+}$ be a stochastic process adapted to the filtration generated by $(Y_t)_{t\in\mathcal{R}^+}$, and such that

$$\mathrm{E}\left[\int_0^T |\phi_t|\mathrm{d}t\right] < \infty, \qquad \text{for all } T > 0.$$

Let J_t be a compound Poisson process with a corresponding Poisson process N_t with intensity λ and a jump size distribution $\mathcal{D}$. The expected value of the squared compound Poisson stochastic integral can be computed as

$$\mathrm{E}\left[\int_0^T \phi_{t-}\mathrm{d}J_t\right] = \mathrm{E}\left[\int_0^T \phi_{t-}\xi_t\mathrm{d}N_t\right] = \lambda\,\mathrm{E}[\mathcal{D}]\,\mathrm{E}\left[\int_0^T \phi_t\mathrm{d}t\right]. \qquad (2.4)$$

2.4 Numerical methods

In general, a jump-diffusion SDE has a form (2.2), for $t \in [0,T]$. To find a numerical solution for this SDE, we first construct an equidistant time discretization $0 = t_0 < t_1 < t_2 < \cdots < t_N = T$, where N is the number of sub-intervals for $[0,T]$, so that the time step size is $\Delta = \frac{T}{N}$ and $t_n = n\Delta$ for all $n \in \{0,1,2,\ldots,N\}$. However, for jump-adapted numerical methods, we have to build an equidistant time discretization $0 = \widetilde{t}_0 < \widetilde{t}_1 < \widetilde{t}_2 < \cdots < \widetilde{t}_N = T$ with time step size $\Delta = \frac{T}{N}$, draw all jump times in $[0,T]$ namely τ_i for $i \in \{1,2,\ldots,L\}$. Then, we combine the equidistant times and jump times into the final time discretization $0 < t_0 < t_1 < t_2 < \cdots < t_{\widetilde{N}} = T$, which is $\{\widetilde{t}_i\}_{i=0}^N \cup \{\tau_i\}_{i=1}^L$. Let Y_n be a numerical solution at time t_n. For simplicity, we denote $a = a(t_n,Y_n), b = b(t_n,Y_n)$, $c = c(t_n,Y_n)$, $\Delta_n = t_{n+1} - t_n$, $\Delta W_n = W_{t_{n+1}} - W_{t_n} \sim \mathcal{N}(0,\Delta_n)$ and $\Delta J_n = J_{t_{n+1}} - J_{t_n} = \sum_{i=N_{t_n}+1}^{N_{t_{n+1}}} \xi_i$, where $\xi_i \sim \mathcal{D}$. We choose seven methods from [6] which are described in Subsections 2.4.1- 2.4.7. The last selected method is a simplified version of the method in Subsection 2.4.7. For every methods, Y_0 is set to be X_0.

2.4.1 Euler Maruyama method

This method is sometimes called just Euler method. The Euler Maruyama method has the form

$$Y_{n+1} = Y_n + a\Delta_n + b\Delta W_n + c\Delta J_n.$$

2.4.2 Simplified Euler method

The simplified Euler method has the form

$$Y_{n+1} = Y_n + a\Delta_n + b\Delta\widehat{W}_n + c\xi_n\Delta\widehat{p}_n,$$

where $P(\Delta\widehat{W}_n \pm \sqrt{\Delta_n}) = \frac{1}{2}$, $\xi_n \sim \mathcal{D}$ and

$$P\left(\Delta\hat{p}_n = \frac{1}{2}\left(1 + 2\lambda\Delta_n \pm \sqrt{1 + 4\lambda\Delta_n}\right)\right) = \frac{1}{2} \mp \frac{1}{2\sqrt{1 + 4\lambda\Delta_n}}.$$

2.4.3 Jump-adapted Euler method

The jump-adapted Euler method has the form

$$Y_{n+1-} = Y_n + a\Delta_n + b\Delta W_n$$

and

$$Y_{n+1} = Y_{n+1-} + c(Y_{n+1-})(J_{n+1} - J_{n+1-}).$$

2.4.4 Jump-adapted simplified Euler method

The jump-adapted simplified Euler method has the form

$$Y_{n+1-} = Y_n + a\Delta_n + b\Delta\widehat{W}_n$$

and

$$Y_{n+1} = Y_{n+1-} + c(Y_{n+1-})(J_{n+1} - J_{n+1-}),$$

where $P(\Delta\widehat{W}_n = \pm\sqrt{\Delta_n}) = \frac{1}{2}$.

2.4.5 Jump-adapted order two weak method

The jump-adapted order two weak method has the form

$$
\begin{aligned}
Y_{n+1-} = {} & Y_n + a\Delta_n + b\Delta W_n + \frac{bb'}{2}\left((\Delta W_n)^2 - \Delta_n\right) + \frac{1}{2}\left(aa' + \frac{1}{2}a''b^2\right)\Delta_n^2 \\
& + \frac{1}{2}\left(a'b + ab' + \frac{1}{2}b''b^2\right)\Delta W_n \Delta_n
\end{aligned}
$$

and

$$Y_{n+1} = Y_{n+1-} + c(Y_{n+1-})\left(J_{n+1} - J_{n+1-}\right).$$

2.4.6 Jump-adapted simplified order two weak method

The jump-adapted simplified order two weak method has the form

$$
\begin{aligned}
Y_{n+1-} = {} & Y_n + a\Delta_n + b\Delta\widetilde{W}_n + \frac{bb'}{2}\left((\Delta\widetilde{W}_n)^2 - \Delta_n\right) + \frac{1}{2}\left(aa' + \frac{1}{2}a''b^2\right)\Delta_n^2 \\
& + \frac{1}{2}\left(a'b + ab' + \frac{1}{2}b''b^2\right)\Delta\widetilde{W}_n \Delta_n
\end{aligned}
$$

and

$$Y_{n+1} = Y_{n+1-} + c(Y_{n+1-})\left(J_{n+1} - J_{n+1-}\right),$$

where $P(\widetilde{W}_n = \pm\sqrt{2\Delta_n}) = \frac{1}{6}$ and $P(\Delta\widetilde{W}_n = 0) = \frac{2}{3}$.

2.4.7 Jump-adapted order two derivative free method

The jump-adapted order two derivative free method has the form

$$
\begin{aligned}
Y_{n+1-} = {} & Y_n + \frac{1}{2}\left(a(t_n, \bar{Y}_n) + a(t_n, Y_n)\right)\Delta_n + \frac{1}{4}\left(b(t_n, \bar{Y}_n^{+})\right. \\
& \left. + \, b(t_n, \bar{Y}_n^{-}) + 2b(t_n, Y_n)\right)\Delta W_n \\
& + \frac{1}{4\sqrt{\Delta_n}}\left(b(t_n, \bar{Y}_n^{+}) - b(t_n, \bar{Y}_n^{-})\right)\left((\Delta W_n)^2 - \Delta_n\right)
\end{aligned}
$$

and

$$Y_{n+1} = Y_{n+1-} + c(Y_{n+1-})\left(J_{n+1} - J_{n+1-}\right),$$

with supporting values

$$\bar{Y}_n = Y_n + a\Delta_n + b\Delta W_n$$

and

$$\bar{Y}_n^{\pm} = Y_n + a\Delta_n \pm b\sqrt{\Delta_n}.$$

2.4.8 Jump-adapted simplified order two derivative free method

The jump-adapted simplified order two derivative free method has the form

$$\begin{aligned} Y_{n+1-} = & Y_n + \frac{1}{2}\left(a(t_n, \bar{Y}_n) + a(t_n, Y_n)\right)\Delta_n + \frac{1}{4}\left(b(t_n, \bar{Y}_n^{+})\right. \\ & \left. + b(t_n, \bar{Y}_n^{-}) + 2b(t_n, Y_n)\right)\Delta\widehat{W}_n \\ & + \frac{1}{4\sqrt{\Delta_n}}\left(b(t_n, \bar{Y}_n^{+}) - b(t_n, \bar{Y}_n^{-})\right)\left((\Delta\widehat{W}_n)^2 - \Delta_n\right) \end{aligned}$$

and

$$Y_{n+1} = Y_{n+1-} + c(Y_{n+1-})\left(J_{n+1} - J_{n+1-}\right),$$

with supporting values

$$\bar{Y}_n = Y_n + a\Delta_n + b\Delta\widehat{W}_n$$

and

$$\bar{Y}_n^{\pm} = Y_n + a\Delta_n \pm b\sqrt{\Delta_n},$$

where $P(\Delta\widehat{W}_n = \pm\sqrt{\Delta_n}) = \frac{1}{2}$.

From now on, we call Euler Maruyama method, simplified Euler method, jump-adapted Euler method, jump-adapted simplified Euler method, jump-adapted order two weak metod, jump-adapted simplified order two weak method, jump-adapted order two derivative free method and jump-adapted simplified order two derivative free method by EM, SE, JE, JSE, JW, JSW, JD and JSD, respectively.

2.5 Weak order of convergence

For a certain numerical method, the discrete time approximation Y_T converges weakly with order β to X_T, if for each $g \in \mathcal{C}_P^{2\beta+1}(\mathbb{R}, \mathbb{R})$, there exists a positive constant

C independent of Δ, such that

$$\varepsilon_w(\Delta) := \mid E[g(X_T)] - E[g(Y_T)] \mid \leq C\Delta^{\beta}, \tag{2.5}$$

for all sufficiently small Δ. Here, $\mathcal{C}_p^{2(\beta+1)}(\mathbb{R},\mathbb{R})$ denotes the space of $2(\beta+1)$ continuously differentiable functions, which together with their derivative of order up to $2(\beta+1)$ have polynomial growth. This means that for $g \in \mathcal{C}_p^{2(\beta+1)}(\mathbb{R},\mathbb{R})$, there exist constants $K > 0$ and $r \in \mathbb{N}$ depending on g such that

$$\mid g^{(j)}(y) \mid \leq K(1 + |y|^{2r}), \tag{2.6}$$

for all $y \in \mathbb{R}$ and $j \leq 2(\beta+1)$.

To find a weak order of convergence for a certain method, we have to find the highest β that holds the inequality (2.5). In practice, we select the function g in (2.5) and (2.6) to be the identity function. Then, perform a linear regression of

$$\log\left(\varepsilon_w(\Delta)\right) = \log(C) + \beta\log\left(\Delta\right), \tag{2.7}$$

where $\log(\Delta)$ is the explanatory variable and $\log\left(\varepsilon_w(\Delta)\right)$ is the response variable.

CHAPTER III

METHODOLOGY

In this chapter, we explain how we proceed our research. Section 3.1 explains how to transform the SDE (1.3) into another SDE which has a constant drift coefficient. Section 3.2 provides eight numerical schemes for the SDE (1.3) and the corresponding eight numerical schemes for the transformed SDE. We test the positivity preserving of the numerical solutions for the sixteen schemes in Section 3.3. We derive the formula for the expectation of the exact solution in Section 3.4. Section 3.5 shows how to numerically find weak orders of convergence for the sixteen schemes. Then, we compare their performance in Chapter IV and conclude in Chapter V. Fig. 3.1 shows a diagram of our procedure described above.

We set parameters $\kappa = 2$, $\theta = 50$, $\sigma = 0.30$, $\gamma = 0.80$, $\lambda = 5$, $X_0 = 100$ and $\mathcal{D} \sim SLog\mathcal{N}(0, 0.1, 1)$. We choose $\alpha = \frac{3}{4}$ for JCEV model and of course, $\alpha = \frac{1}{2}$ for JCIR model for all simulations.

3.1 Transformed approach

For a transformed approach, the regular SDE (2.1) can be transformed by (2.3). Applying the transformation $f(X_t) = X_t^{1-\alpha}$ suggested in [12] to the SDE (1.3), we obtain

$$
\begin{aligned}
X_t^{1-\alpha} = X_0^{1-\alpha} &+ \int_0^t \sigma X_s^{\alpha}(1-\alpha)X_s^{-\alpha}\mathrm{d}W_s + \frac{1}{2}\int_0^t (-\alpha)(1-\alpha)X_s^{-\alpha-1}\left(\sigma X_s^{\alpha}\right)^2 \mathrm{d}s \\
&+ \int_0^t \kappa(\theta - X_s)(1-\alpha)X_s^{-\alpha}\mathrm{d}s + \int_0^t \left((X_{s-} + \gamma X_{s-}\xi_{N_s})^{1-\alpha} - X_{s-}^{1-\alpha} \right) \mathrm{d}N_s.
\end{aligned}
$$

We substitute $X_t^{1-\alpha}$ by U_t. Therefore, the transformed SDE of (1.3) is

$$
\begin{aligned}
\mathrm{d}U_t = &\left(\frac{(\alpha^2 - \alpha)\sigma^2}{2U_t} + \kappa(1-\alpha)\left(\theta U_t^{-\frac{\alpha}{1-\alpha}} - U_t \right) \right) \mathrm{d}t \\
&+ \sigma(1-\alpha)\mathrm{d}W_t + U_{t-}\widetilde{\xi}_{N_t}\mathrm{d}N_t, \qquad (3.1)
\end{aligned}
$$

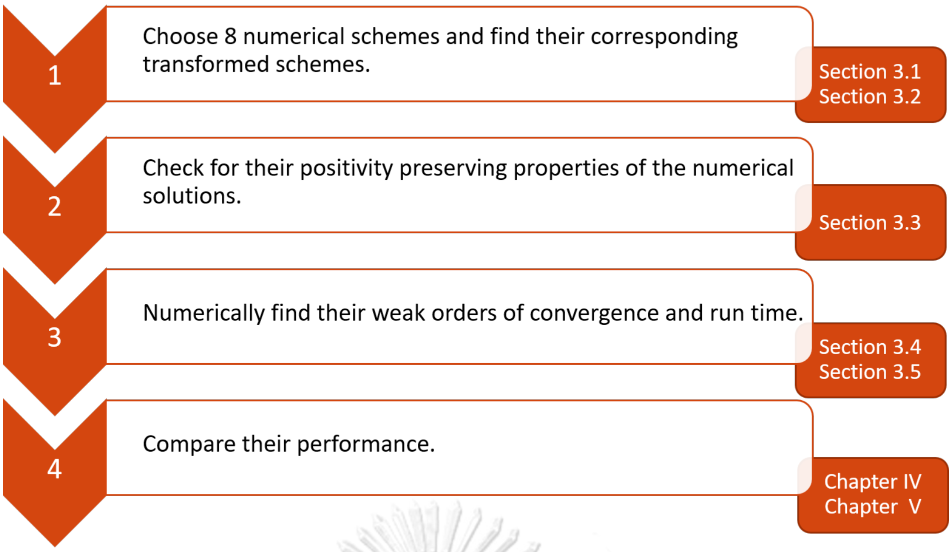


Figure 3.1: Procedure diagram

where $\widetilde{\xi}_{N_t} = (1 + \gamma\xi_{N_t})^{1-\alpha} - 1$ and $\xi_{N_t} \sim SLog\mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2, 1)$.

We use the time-discretization notation from Section 2.4. Let $\{x_n\}_{n\in\{0,1,2,\dots,N\}}$ be a numerical solution of SDE (1.3) from the direct approach at time t_n and $\{v_n\}_{n\in\{0,1,2,\dots,N\}}$ be a numerical solution of the SDE (3.1) from the transformed approach at time t_n. For the direct approach, we find numerical solutions x_n of (1.3) directly. For the transformed approach, to get a numerical solution of the original SDE (1.3), we need to transformed v_n back via the transformation $x_n = v_n^{\frac{1}{1-\alpha}}$. Fig. 3.2 shows a diagram of the procedure for both direct and transformed approaches to get numerical solutions of the original SDE (1.3).

Since there are eight methods for the direct approach, we also apply them to (3.1) for the transformed approach. For the eight transformed schemes for (3.1), we call them transformed Euler Maruyama scheme, transformed simplified Euler scheme, transformed jump-adapted Euler scheme, transformed jump-adapted simplified Euler scheme, transformed jump-adapted order two weak scheme, transformed jump-adapted simplified order two weak scheme, transformed jump-adapted order two derivative free scheme and transformed jump-adapted simplified order two derivative free scheme. We abbreviate them by TEM, TSE, TJE, TJSE, TJW, TJSW, TJD and TJSD, respectively, where T stands for "Transformed".

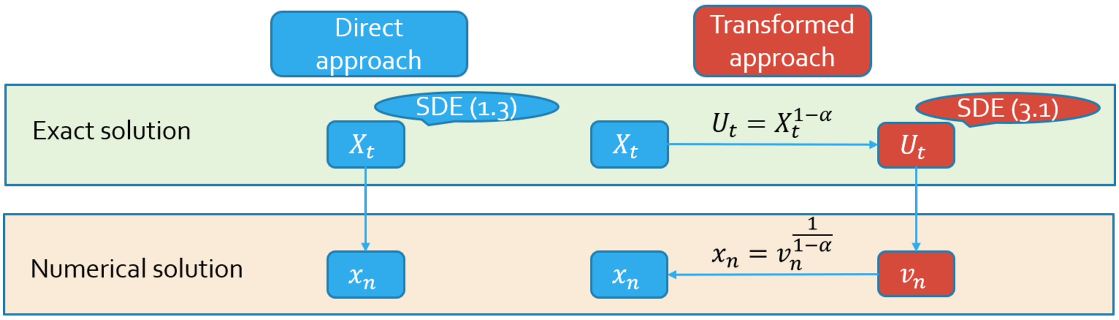


Figure 3.2: Direct approach and transformed approach diagram

3.2 Numerical schemes

We use numerical schemes in this section to solve for numerical solutions of SDEs (1.3) and (3.1).

3.2.1 EM

The EM for the SDE (1.3) has the form

$$x_{n+1} = x_n + \kappa(\theta - x_n)\Delta_n + \sigma x_n^\alpha \Delta W_n + \gamma x_n \sum_{i=N_n+1}^{N_{n+1}} \xi_i.$$

3.2.2 SE

The SE for the SDE (1.3) has the form

$$x_{n+1} = x_n + \kappa(\theta - x_n)\Delta_n + \sigma x_n^\alpha \Delta \widehat{W}_n + \gamma x_n \xi_n \Delta \widehat{p}_n.$$

3.2.3 JE

The JE for the SDE (1.3) has the form

$$x_{n+1-} = x_n + \kappa(\theta - x_n)\Delta_n + \sigma x_n^\alpha \Delta W_n$$

and

$$x_{n+1} = x_{n+1-} + \gamma x_{n+1-} \xi_n.$$

3.2.4 JSE

The JSE for the SDE (1.3) has the form

$$x_{n+1-} = x_n + \kappa(\theta - x_n)\Delta_n + \sigma x_n^\alpha \Delta\widehat{W}_n$$

and

$$x_{n+1} = x_{n+1-} + \gamma x_{n+1-}\xi_n.$$

3.2.5 JW

The JW for the SDE (1.3) has the form

$$\begin{aligned} x_{n+1-} = &x_n + \kappa(\theta - x_n)\Delta_n + \sigma x_n^\alpha \Delta W_n + \frac{\sigma^2 \alpha x_n^{2\alpha-1}}{2}\Big((\Delta W_n)^2 - \Delta_n\Big) \\ &- \frac{1}{2}\kappa^2(\theta - x_n)\Delta_n^2 \\ &+ \frac{1}{2}\Big(-\kappa\sigma x_n^\alpha + \kappa\sigma\alpha(\theta - x_n) + \frac{1}{2}\sigma^3\alpha(\alpha-1)x_n^{3\alpha-2}\Big)\Delta W_n \Delta_n \end{aligned}$$

and

$$x_{n+1} = x_{n+1-} + \gamma x_{n+1-}\xi_n.$$

3.2.6 JSW

The JSW for the SDE (1.3) has the form

$$\begin{aligned} x_{n+1-} = &x_n + \kappa(\theta - x_n)\Delta_n + \sigma x_n^\alpha \Delta\widetilde{W}_n + \frac{\sigma^2 \alpha x_n^{2\alpha-1}}{2}\Big((\Delta\widetilde{W}_n)^2 - \Delta_n\Big) \\ &- \frac{1}{2}\kappa^2(\theta - x_n)\Delta_n^2 \\ &+ \frac{1}{2}\Big(-\kappa\sigma x_n^\alpha + \kappa\sigma\alpha(\theta - x_n) + \frac{1}{2}\sigma^3\alpha(\alpha-1)x_n^{3\alpha-2}\Big)\Delta\widetilde{W}_n \Delta_n \end{aligned}$$

and

$$x_{n+1} = x_{n+1-} + \gamma x_{n+1-}\xi_n.$$

3.2.7 JD

The JD for the SDE (1.3) has the form

$$\begin{aligned} x_{n+1-} = &x_n + \frac{1}{2}\kappa(2\theta - \bar{x}_n - x_n)\Delta_n + \frac{1}{4}\sigma(\bar{x}_n^{+\alpha} + \bar{x}_n^{-\alpha} + 2x_n^\alpha)\Delta W_n \\ &+ \frac{\sigma(\bar{x}_n^{+\alpha} - \bar{x}_n^{-\alpha})(\Delta W_n)^2 - \Delta_n}{4\sqrt{\Delta_n}} \end{aligned}$$

and

$$x_{n+1} = x_{n+1-} + \gamma x_{n+1-}\xi_n,$$

with supporting values

$$\bar{x}_n = x_n + \kappa(\theta - x_n)\Delta_n + \sigma x_n^{\alpha}\Delta W_n$$

and

$$\bar{x}_n^{\pm} = x_n + \kappa(\theta - x_n)\Delta_n \pm \sigma x_n^{\alpha}\sqrt{\Delta_n}.$$

3.2.8 JSD

The JSD order two scheme for the SDE (1.3) has the form

$$\begin{aligned} x_{n+1-} =& x_n + \frac{1}{2}\kappa(2\theta - \bar{x}_n - x_n)\Delta_n + \frac{1}{4}\sigma(\bar{x}_n^{+\alpha} + \bar{x}_n^{-\alpha} + 2x_n^{\alpha})\Delta\widehat{W}_n \\ &+ \frac{\sigma(\bar{x}_n^{+\alpha} - \bar{x}_n^{-\alpha})(\Delta\widehat{W}_n)^2 - \Delta_n}{4\sqrt{\Delta_n}} \end{aligned}$$

and

$$x_{n+1} = x_{n+1-} + \gamma x_{n+1-}\xi_n,$$

with supporting values

$$\bar{x}_n = x_n + \kappa(\theta - x_n)\Delta_n + \sigma x_n^{\alpha}\Delta\widehat{W}_n,$$

and

$$\bar{x}_n^{\pm} = x_n + \kappa(\theta - x_n)\Delta_n \pm \sigma x_n^{\alpha}\sqrt{\Delta_n}.$$

3.2.9 TEM

Recall that for the transformed scheme $\widetilde{\xi}_n = (1 + \gamma\xi_n)^{1-\alpha} - 1$, where $\xi_n \sim SLog\mathcal{N}(0, 0.1, 1)$. The TEM for the SDE (3.1) has the form

$$\begin{aligned} v_{n+1} =& v_n + \left(\frac{(\alpha^2 - \alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n\right)\right)\Delta_n \\ &+ \sigma(1-\alpha)\Delta W_n + v_n\sum_{i=N_n+1}^{N_{n+1}}\widetilde{\xi}_i. \end{aligned}$$

3.2.10 TSE

The TSE for the SDE (3.1) has the form

$$\begin{aligned} v_{n+1} = v_n &+ \left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n\right)\right)\Delta_n \\ &+ \sigma(1-\alpha)\Delta\widehat{W}_n + v_n\widetilde{\xi}_n\Delta\widehat{p}_n. \end{aligned}$$

3.2.11 TJE

The TJE for the SDE (3.1) has the form

$$v_{n+1-} = v_n + \left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n\right)\right)\Delta_n + \sigma(1-\alpha)\Delta W_n$$

and

$$v_{n+1} = v_{n+1-} + v_{n+1-}\widetilde{\xi}_n.$$

3.2.12 TJSE

The TJSE for the SDE (3.1) has the form

$$v_{n+1-} = v_n + \left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n\right)\right)\Delta_n + \sigma(1-\alpha)\Delta\widehat{W}_n$$

and

$$v_{n+1} = v_{n+1-} + v_{n+1-}\widetilde{\xi}_n.$$

3.2.13 TJW

The TJW for the SDE (3.1) has the form

$$\begin{aligned} v_{n+1-} = v_n &+ \left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n\right)\right)\Delta_n + \sigma(1-\alpha)\Delta W_n \\ &+ \frac{1}{2}\Bigg(\left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n\right)\right) \\ &\times \left(-\frac{(\alpha^2-\alpha)\sigma^2}{2v_n^2} + \kappa(1-\alpha)\theta\left(-\frac{\alpha}{1-\alpha} v_n^{-\frac{1}{1-\alpha}} - 1\right)\right. \\ &+ \frac{1}{2}\left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n^3} + \kappa(1-\alpha)\theta\frac{\alpha}{(1-\alpha)^2} v_n^{\frac{\alpha-2}{1-\alpha}}\right)\Bigg)(\sigma(1-\alpha))^2\Bigg)\Delta_n^2 \\ &+ \frac{1}{2}\Bigg(\left(-\frac{(\alpha^2-\alpha)\sigma^2}{2v_n^2} + \kappa(1-\alpha)\theta\left(-\frac{\alpha}{1-\alpha} v_n^{-\frac{1}{1-\alpha}} - 1\right)\right) \\ &\times \sigma(1-\alpha)\Bigg)\Delta W_n\Delta_n \end{aligned}$$

and

$$v_{n+1} = v_{n+1-} + v_{n+1-}\widetilde{\xi}_n.$$

3.2.14 TJSW

The TJSW for the SDE (3.1) has the form

$$\begin{aligned}
v_{n+1-} = v_n &+ \left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n\right)\right)\Delta_n + \sigma(1-\alpha)\Delta\widetilde{W}_n \\
&+ \frac{1}{2}\Bigg(\left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n\right)\right) \\
&\times \left(-\frac{(\alpha^2-\alpha)\sigma^2}{2v_n^2} + \kappa(1-\alpha)\theta\left(-\frac{\alpha}{1-\alpha}v_n^{-\frac{1}{1-\alpha}} - 1\right)\right. \\
&+ \left.\frac{1}{2}\left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n^3} + \kappa(1-\alpha)\theta\frac{\alpha}{(1-\alpha)^2}v_n^{\frac{\alpha-2}{1-\alpha}}\right)\right)(\sigma(1-\alpha))^2\Bigg)\Delta_n^2 \\
&+ \frac{1}{2}\Bigg(\left(-\frac{(\alpha^2-\alpha)\sigma^2}{2v_n^2} + \kappa(1-\alpha)\theta\left(-\frac{\alpha}{1-\alpha}v_n^{-\frac{1}{1-\alpha}} - 1\right)\right) \\
&\times \sigma(1-\alpha)\Bigg)\Delta\widetilde{W}_n\Delta_n
\end{aligned}$$

and

$$v_{n+1} = v_{n+1-} + v_{n+1-}\widetilde{\xi}_n.$$

3.2.15 TJD

The TJD for the SDE (3.1) has the form

$$\begin{aligned}
v_{n+1-} = v_n + \frac{1}{2}\Bigg(&\left(\frac{(\alpha^2-\alpha)\sigma^2}{2\bar{v}_n} + \kappa(1-\alpha)\left(\theta\bar{v}_n^{-\frac{\alpha}{1-\alpha}} - \bar{v}_n\right)\right) \\
&+ \left(\frac{(\alpha^2-\alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n\right)\right)\Bigg)\Delta_n \\
&+ (\sigma(1-\alpha))\,\Delta W_n
\end{aligned}$$

and

$$v_{n+1} = v_{n+1-} + v_{n+1-}\widetilde{\xi}_n,$$

with supporting value

$$\bar{v}_n = v_n + \left(\frac{(\alpha^2 - \alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n \right) \right) \Delta_n + \sigma(1-\alpha)\Delta W_n.$$

3.2.16 TJSD

The TJSD for the SDE (3.1) has the form

$$\begin{aligned} v_{n+1-} = v_n + \frac{1}{2}\Bigg(& \left(\frac{(\alpha^2 - \alpha)\sigma^2}{2\bar{v}_n} + \kappa(1-\alpha)\left(\theta \bar{v}_n^{-\frac{\alpha}{1-\alpha}} - \bar{v}_n \right) \right) \\ & + \left(\frac{(\alpha^2 - \alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n \right) \right) \Bigg)\Delta_n \\ & + (\sigma(1-\alpha))\, \Delta\widehat{W}_n \end{aligned}$$

and

$$v_{n+1} = v_{n+1-} + v_{n+1-}\widetilde{\xi}_n,$$

with supporting value

$$\bar{v}_n = v_n + \left(\frac{(\alpha^2 - \alpha)\sigma^2}{2v_n} + \kappa(1-\alpha)\left(\theta v_n^{-\frac{\alpha}{1-\alpha}} - v_n \right) \right) \Delta_n + \sigma(1-\alpha)\Delta\widehat{W}_n.$$

3.3 Test for positivity preserving

In this section, we test positivity preserving of numerical solutions to support that our simulation do not provide the negative paths. Because fractional roots with even denominator appear in JCIR and JCEV models, sample paths should always be positive. If numerical solutions obtained from using a certain numerical method became negative, we would get complex numbers as part of the numerical solutions. This indicates that the numerical method is not valid to simulate JCIR and JCEV models. Recall that we set parameters $\kappa = 2$, $\theta = 50$, $\sigma = 0.30$, $\gamma = 0.80$, $\lambda = 5$, $X_0 = 100$ and $\mathcal{D} \sim SLog\mathcal{N}(0, 0.1, 1)$. We choose $\alpha = \frac{3}{4}$ for JCEV model and of course, $\alpha = \frac{1}{2}$ for JCIR model for all simulations. We set T to be 2, 4 and 8 and Δ to be $\frac{1}{4}$, $\frac{1}{8}$ and $\frac{1}{16}$. Therefore, for each numerical scheme, we have nine cases to simulate sample paths. For each case, we simulate 10,000 sample paths. Hence, we simulate 90,000 sample paths for each scheme.

If the numerical scheme provides at lease one negative sample path, we will reject that numerical scheme.

3.4 Expectation of exact solution

To find the exact expectation of X_t in (1.3), we can use the expected value of the compound Poisson stochastic integral (2.4). For SDE (1.3), we have that

$$\begin{aligned} \mathrm{E}\left[X_t\right] = {} & \mathrm{E}[X_0] + \mathrm{E}\left[\int_0^t \kappa(\theta - X_s)\mathrm{d}s\right] + \mathrm{E}\left[\int_0^t \sigma X_s^{\alpha}\mathrm{d}W_s\right] \\ & + \mathrm{E}\left[\int_0^t \gamma X_{s-}\mathrm{d}J_s\right]. \end{aligned} \tag{3.2}$$

Then, we apply (2.4) to (3.2) and have that

$$\mathrm{E}\left[X_t\right] = X_0 + \kappa\theta t - (-\kappa + \gamma\lambda\,\mathrm{E}\left[\mathcal{D}\right]) \int_0^t \mathrm{E}\left[X_s\right]\mathrm{d}s.$$

Let $f(t) = \mathrm{E}\left[X_t\right]$. Then, we have

$$f(t) = X_0 + \kappa\theta t + (-\kappa + \gamma\lambda\,\mathrm{E}\left[\mathcal{D}\right]) \int_0^t f(s)\mathrm{d}s. \tag{3.3}$$

To solve this ordinary differential equation (ODE), we take derivative $\frac{\mathrm{d}}{\mathrm{d}t}$ to (3.3) and let $A = -\kappa + \gamma\lambda\,\mathrm{E}[\mathcal{D}]$. Then, we get

$$\frac{\mathrm{d}}{\mathrm{d}t} f(t) = \kappa\theta + A f(t).$$

Then,

$$\ln\left|\frac{\kappa\theta + A f(t)}{\kappa\theta + A f(0)}\right| = At$$

Substituting $f(0) = \mathrm{E}[X_0] = X_0$ and $f(t) = \mathrm{E}[X_t]$, we have

$$\begin{aligned} |\kappa\theta + A\,\mathrm{E}[X_t]| &= |\kappa\theta + AX_0|e^{At} \\ \kappa\theta + A\,\mathrm{E}[X_t] &= \pm|\kappa\theta + AX_0|e^{At} \\ \mathrm{E}[X_t] &= \frac{-\kappa\theta \pm |\kappa\theta + AX_0|e^{At}}{A}. \end{aligned} \tag{3.4}$$

Applying the initial condition $E[X_0] = X_0$, we have that

$$X_0 = \frac{-\kappa\theta \pm |\kappa\theta + AX_0|}{A}. \tag{3.5}$$

To satisfy (3.5), we have to choose the operator + and $|\kappa\theta + AX_0| = (\kappa\theta + AX_0)$, so that

$$X_0 = \frac{-\kappa\theta + (\kappa\theta + AX_0)}{A} = X_0. \tag{3.6}$$

From (3.4) and (3.6), we can conclude that the exact expectation of (1.3) at time t is

$$\mathrm{E}[X_t] = \frac{-\kappa\theta + (\kappa\theta + AX_0)\,e^{At}}{A}, \tag{3.7}$$

where $A = -\kappa + \gamma\lambda\,\mathrm{E}[\mathcal{D}]$.

3.5 Finding weak orders of convergence

Recall that we set parameters $\kappa = 2$, $\theta = 50$, $\sigma = 0.30$, $\gamma = 0.80$, $\lambda = 5$, $X_0 = 100$ and $\mathcal{D} \sim SLog\mathcal{N}(0, 0.1, 1)$. We choose $\alpha = \frac{3}{4}$ for JCEV model and of course, $\alpha = \frac{1}{2}$ for JCIR model for all simulations. From (2.7), to numerically find weak order of convergence β and intercept $\log(C)$ for each scheme, we have to perform linear regressions with various values of Δ. Here, we use the same notation from Sections 2.5 and 3.1. Let $\mathrm{E}\left[x_N^\Delta\right]$ be the expectation of numerical solutions using a certain numerical scheme with time step of size on the time domain $[0, T]$. Note that x_N^Δ is a numerical solution at time T.

In our linear regression simulation, we choose $T = 1$ and $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ and $\frac{1}{64}$. To find the expectation of numerical solution $\mathrm{E}\left[x_N^\Delta\right]$ for each value of Δ, we simulate 10^7

sample paths to get $x_{N,i}^{\Delta}$ for $i = 1, 2, \ldots, 10^7$. Then, we approximate $\mathrm{E}\left[x_N^{\Delta}\right]$ by

$$\mathrm{E}\left[x_N^{\Delta}\right] = \frac{\sum_{i=1}^{10^7} x_{N,i}^{\Delta}}{10^7}.$$

To find the expectation of the exact solution $\mathrm{E}\left[X_T\right]$, we just substitute $T = 1$ and all parameters which we already set to (3.7). Then, we receive $\log\left|\mathrm{E}\left[X_T\right] - \mathrm{E}\left[x_N^{\Delta}\right]\right|$ for each Δ. Therefore, for each scheme we can perform linear regression where $\log(\Delta)$ is the explanatory variable and $\log\left|\mathrm{E}\left[X_T\right] - \mathrm{E}\left[x_N^{\Delta}\right]\right|$ is the response variable to find weak orders of convergence β and intercepts $\log(C)$. We collect run time for each numerical scheme by the Matlab code `tic` and `toc`, shown in Appendix **A** to **H**.

CHAPTER IV

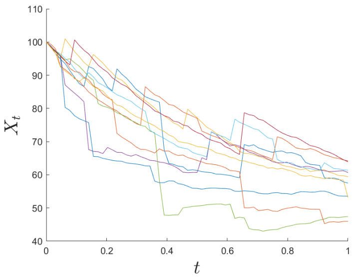
EXPERIMENTAL RESULTS

In this chapter, we show the positivity preserving result from Section 3.3, the regression result and run time from Section 3.5 for both JCIR and JCEV models.

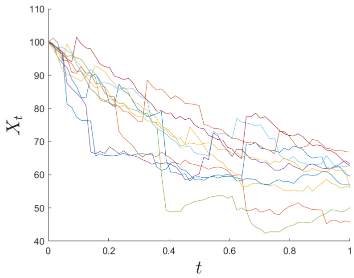
4.1 Positive sample paths

Recall from Section 3.3 that we set parameters $\kappa = 2$, $\theta = 50$, $\sigma = 0.30$, $\gamma = 0.80$, $\lambda = 5$, $X_0 = 100$ and $\mathcal{D} \sim SLog\mathcal{N}(0, 0.1, 1)$. We also set $T \in \{2, 4, 8\}$ and $\Delta \in \{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$. Thus, we have nine cases to simulate for each method. For each case, we simulate 10,000 sample paths to see if the numerical method provides any negative sample paths. We choose $\alpha = \frac{3}{4}$ for JCEV model and of course, $\alpha = \frac{1}{2}$ for JCIR model for all simulations. For all sixteen numerical methods for JCIR and JCEV models, every $T \in \{2, 4, 8\}$ and every $\Delta \in \{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$, all of the 10,000 sample paths yield positive numerical solutions. Therefore, we go on with these numerical methods to find weak orders of convergence by the procedure in Section 3.5.

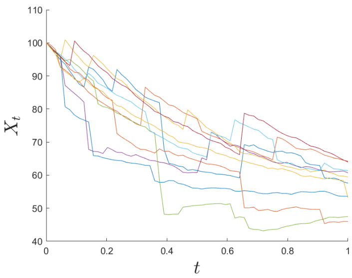
Recall from Section 3.5 that for each numerical method, we simulate 10^7 sample paths for each value of $\Delta \in \left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}\right\}$ to perform the linear regression to find weak orders of convergence. For a certain method, within these 5×10^7 sample paths, if numerical solutions became negative, we would get complex numbers which indicate invalidity of that numerical method. Fig. 4.1 shows the first 10 sample paths from EM method and the corresponding transformed method, TEM, with $T = 1$ and $\Delta = \frac{1}{64}$. Fig. 4.2 - 4.8 have similar explanation to Fig. 4.1 for SE, JE, JSE, JW, JSW, JD and JSD, respectively. For all 16 numerical methods and for all values of $\Delta \in \left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}\right\}$, every sample path is positive.



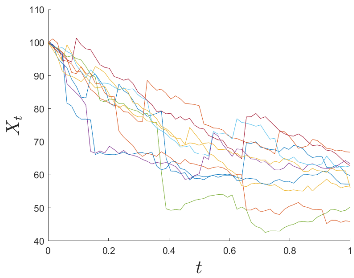
(a) EM with $\alpha = \frac{1}{2}$



(b) EM with $\alpha = \frac{3}{4}$

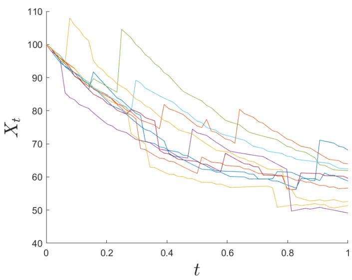


(c) TEM with $\alpha = \frac{1}{2}$

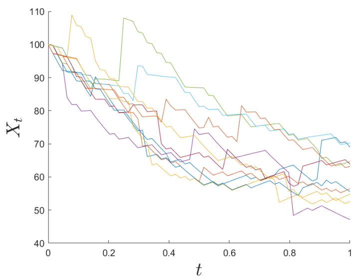


(d) TEM with $\alpha = \frac{3}{4}$

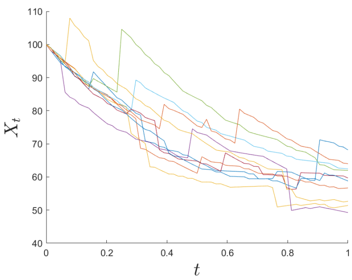
Figure 4.1: 10 sample paths of EM and TEM simulations



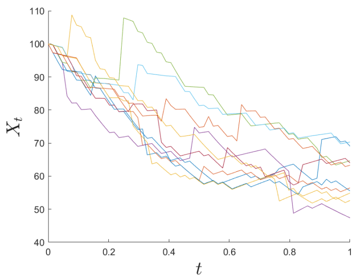
(a) SE with $\alpha = \frac{1}{2}$



(b) SE with $\alpha = \frac{3}{4}$

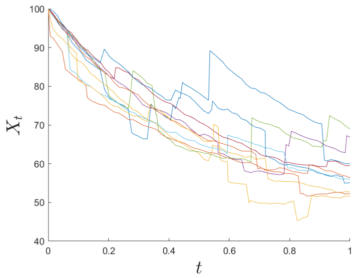


(c) TSE with $\alpha = \frac{1}{2}$

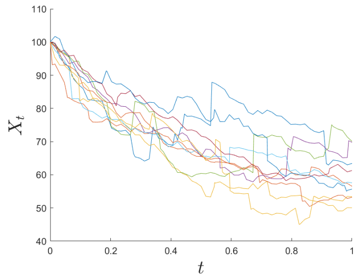


(d) TSE with $\alpha = \frac{3}{4}$

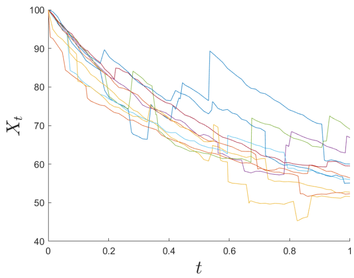
Figure 4.2: 10 sample paths of SE and TSE simulations



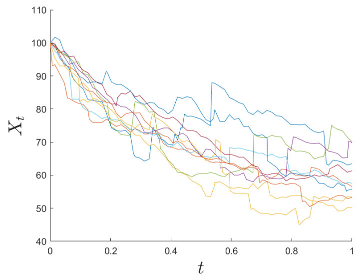
(a) JE with $\alpha = \frac{1}{2}$



(b) JE with $\alpha = \frac{3}{4}$

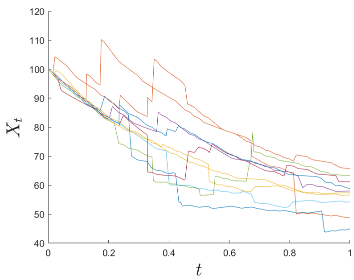


(c) TJE with $\alpha = \frac{1}{2}$

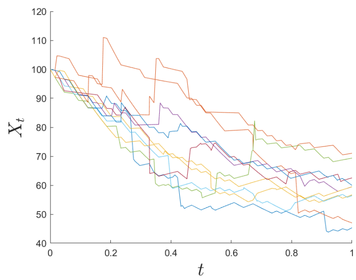


(d) TJE with $\alpha = \frac{3}{4}$

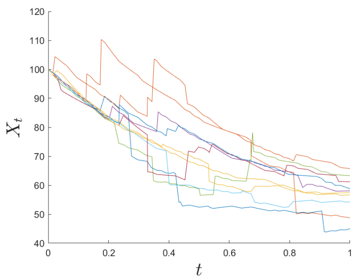
Figure 4.3: 10 sample paths of JE and TJE simulations



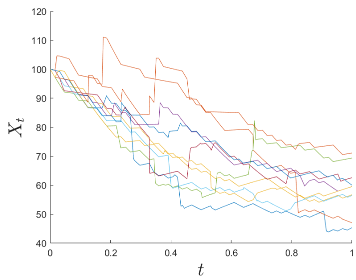
(a) JSE with $\alpha = \frac{1}{2}$



(b) JSE with $\alpha = \frac{3}{4}$

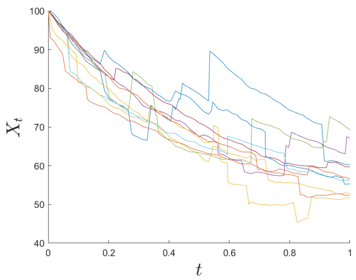


(c) TJSE with $\alpha = \frac{1}{2}$

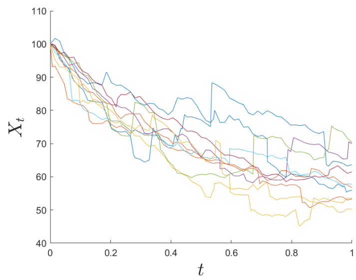


(d) TJSE with $\alpha = \frac{3}{4}$

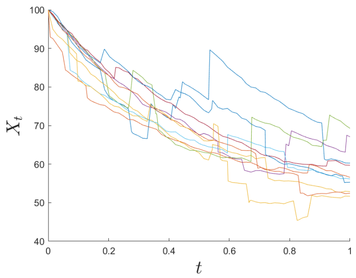
Figure 4.4: 10 sample paths of JSE and TJSE simulations



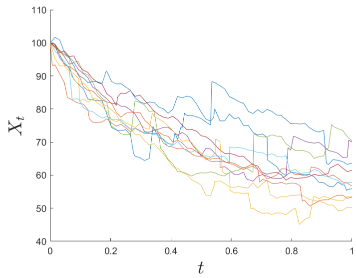
(a) JW with $\alpha = \frac{1}{2}$



(b) JW with $\alpha = \frac{3}{4}$

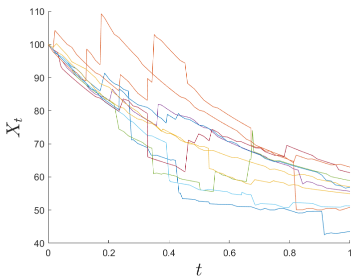


(c) TJW with $\alpha = \frac{1}{2}$

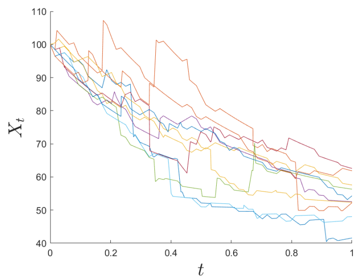


(d) TJW with $\alpha = \frac{3}{4}$

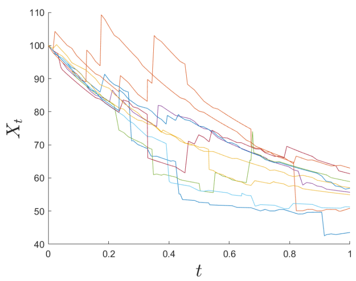
Figure 4.5: 10 sample paths of JW and TJW simulations



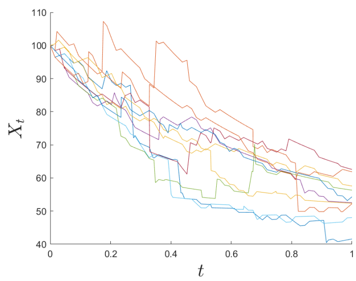
(a) JSW with $\alpha = \frac{1}{2}$



(b) JSW with $\alpha = \frac{3}{4}$

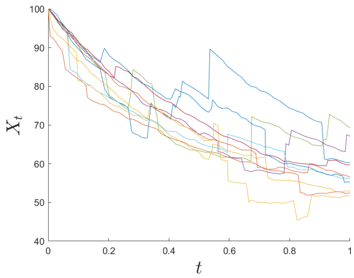


(c) TJSW with $\alpha = \frac{1}{2}$

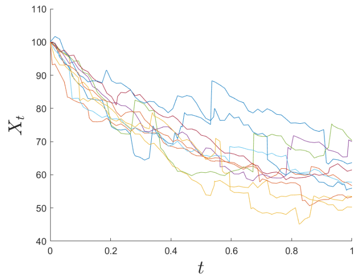


(d) TJSW with $\alpha = \frac{3}{4}$

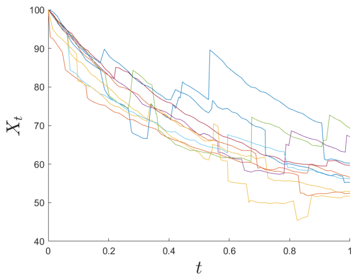
Figure 4.6: 10 sample paths of JSW and TJSW simulations



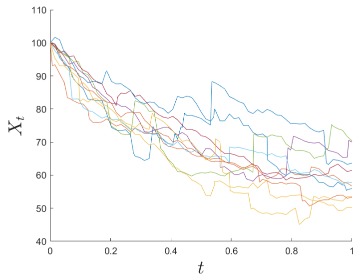
(a) JD with $\alpha = \frac{1}{2}$



(b) JD with $\alpha = \frac{3}{4}$

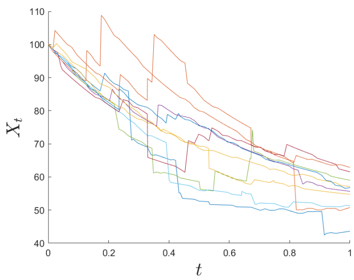


(c) TJD with $\alpha = \frac{1}{2}$

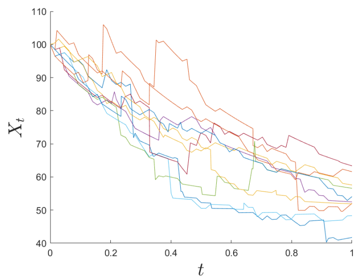


(d) TJD with $\alpha = \frac{3}{4}$

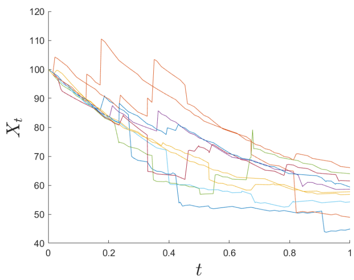
Figure 4.7: 10 sample paths of JD and TJD simulations



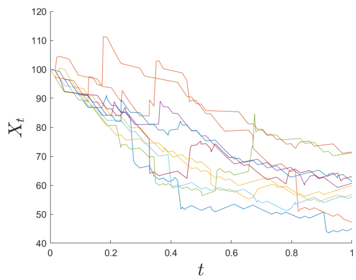
(a) JSD with $\alpha = \frac{1}{2}$



(b) JSD with $\alpha = \frac{3}{4}$



(c) TJSD with $\alpha = \frac{1}{2}$



(d) TJSD with $\alpha = \frac{3}{4}$

Figure 4.8: 10 sample paths of JSD and TJSD simulations

4.2 Weak order of convergence result

Recall from Section 3.5 that we set parameters $\kappa = 2$, $\theta = 50$, $\sigma = 0.30$, $\gamma = 0.80$, $\lambda = 5$, $X_0 = 100$ and $\mathcal{D} \sim SLog\mathcal{N}(0, 0.1, 1)$. We choose $\alpha = \frac{3}{4}$ for JCEV model and of course, $\alpha = \frac{1}{2}$ for JCIR model for all simulations. We use $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$ and final time $T = 1$ to perform linear regression in order to find weak order of convergence β and intercept $\log(C)$ for each method.

Fig. 4.9 and Fig. 4.10 show the regression results with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$ for JCIR and JCEV models, respectively. The X-axis represents $\log(\Delta)$ and the Y-axis represents $\log\left|E\left[X_T\right] - E\left[x_N^\Delta\right]\right|$. In these figures, red and green lines are the results for order-one numerical methods. Blue and magenta lines are the results for order-two numerical methods. The results from the direct approach are presented by solid lines, and the results from the transformed approach are presented by dash lines. The reference lines with slope one and two are also provided by the black solid line and the black dash line, respectively. From these figures, we regard the results for $\Delta = \frac{1}{64}$ as outliers because JW, TJW, JD and JTD do not perfectly fit with straight lines. Notice that JW, TJW, JD and JTD are order-two numerical methods. Therefore, for order-two numerical methods, we use the regression results only for $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$. We show the regression results for JCIR and JCEV models with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ in Fig. 4.11 and Fig. 4.12, respectively.

From Fig. 4.11 and Fig. 4.12, we can see that there are two separate groups of order-one methods and order-two methods. The group of order-two methods is lower and more steep than the group of order-one methods. Therefore, all of the order-two methods provide less weak error than all of the order-one methods.

When we exclude the outliers, the dash lines always be lower than their corresponding solid lines. Thus, the transformed methods can reduce weak error from their corresponding direct methods.

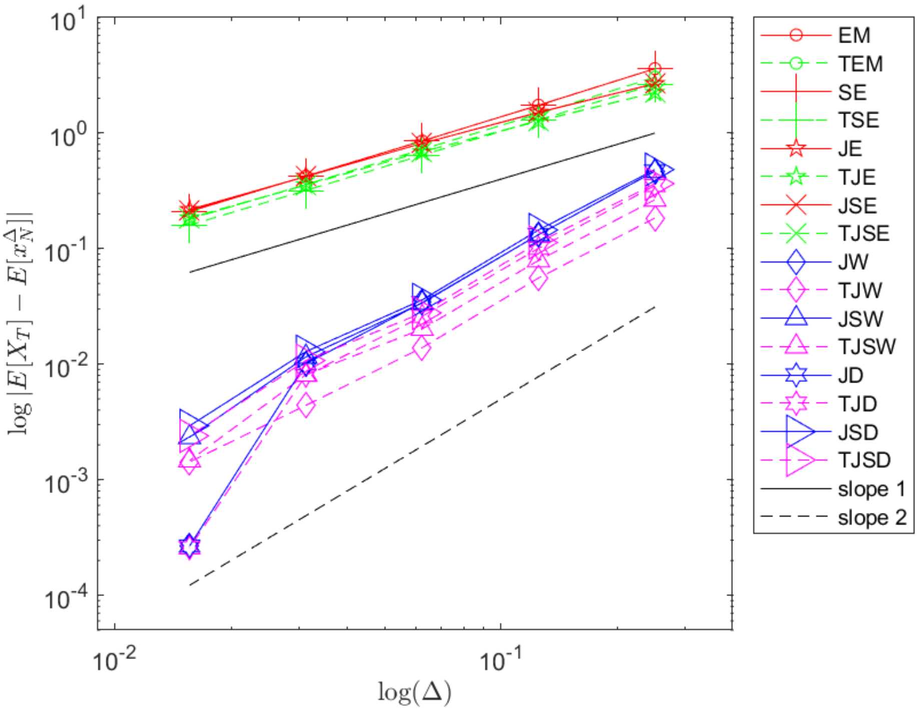


Figure 4.9: Regression results when $\alpha = \frac{1}{2}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$

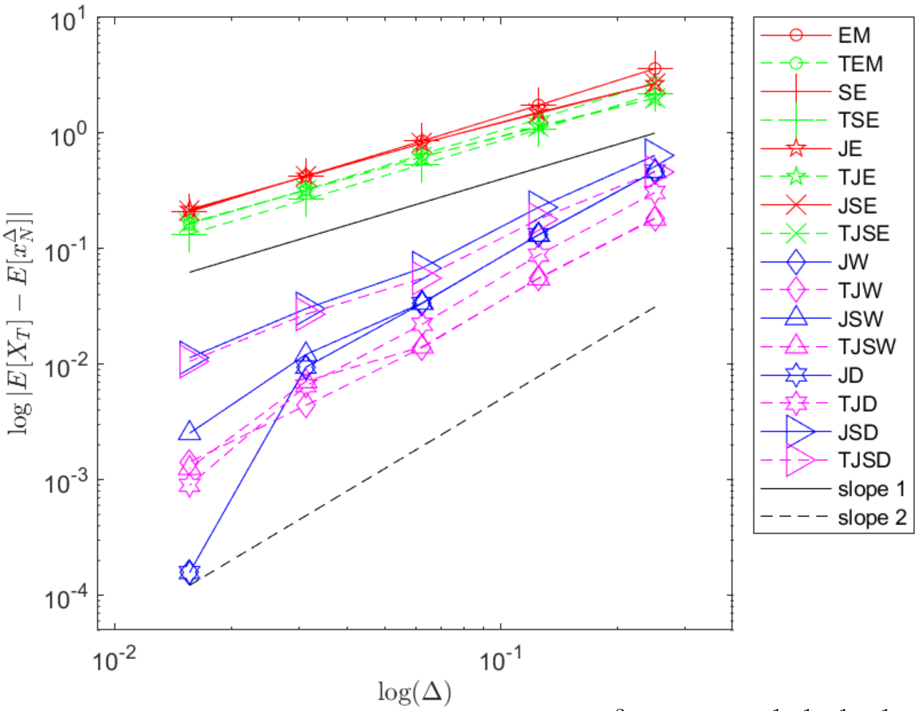


Figure 4.10: Regression results when $\alpha = \frac{3}{4}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$

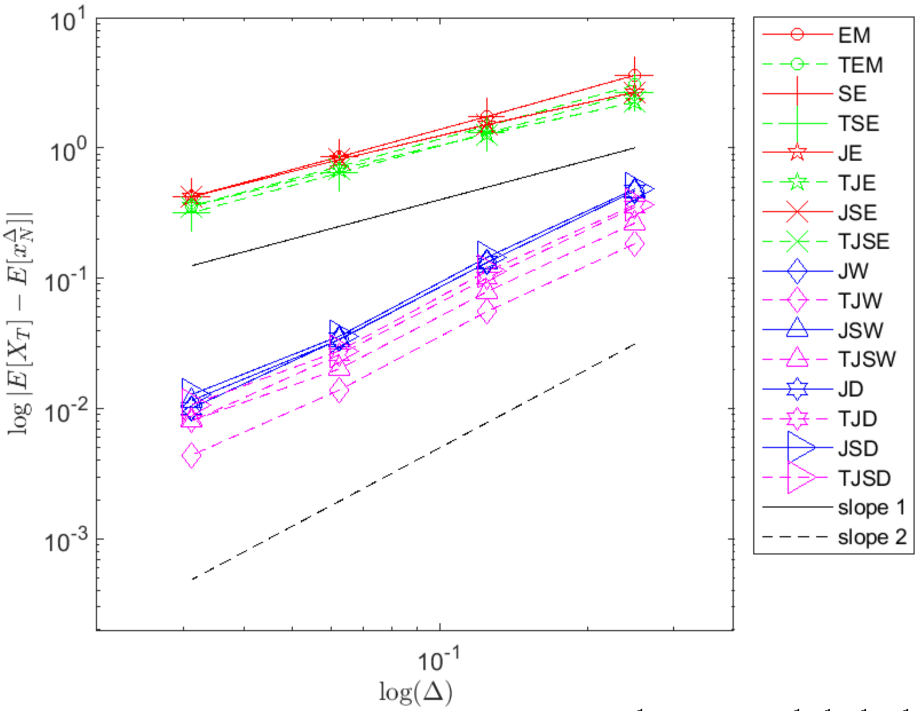


Figure 4.11: Regression results when $\alpha = \frac{1}{2}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$

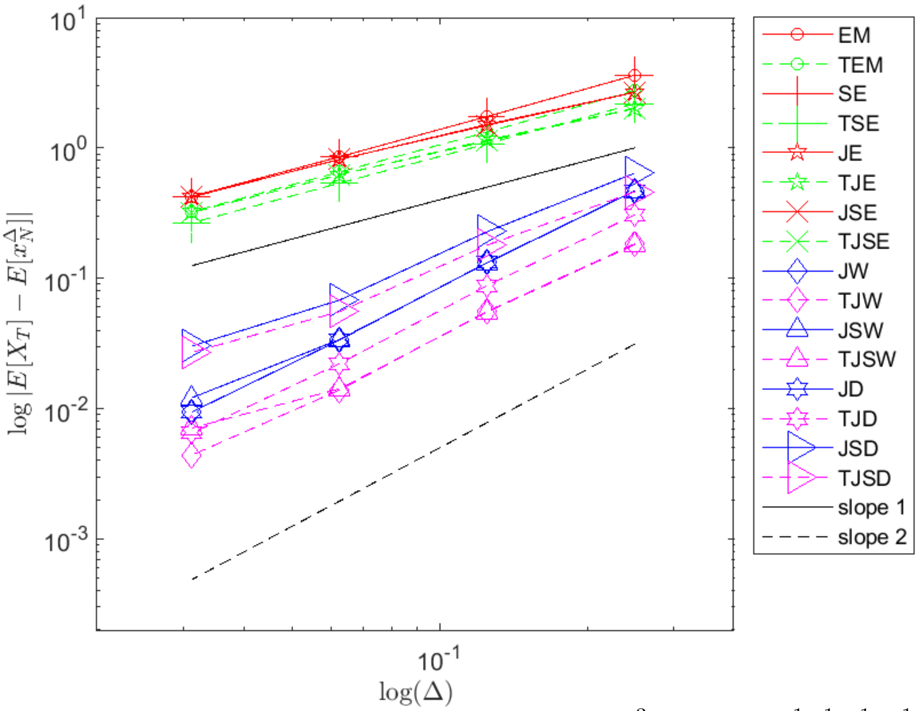


Figure 4.12: Regression results when $\alpha = \frac{3}{4}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$

4.2.1 Order-one regression results

Since there are no outliers in order-one methods, we use $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$ to find weak orders of convergence for JCIR and JCEV models. We show their regression results in Fig. 4.13 and Fig. 4.14, respectively. From Fig. 4.13 and Fig. 4.14, the green dash lines are lower than the red solid lines. This guarantees that for order-one methods, the transformed schemes provide less weak error than their corresponding direct schemes. Moreover, the red solid lines almost overlap with each other for each model. Therefore, EM ,SE, JE and JSE yield almost conformable weak errors. From Fig. 4.13 and Fig. 4.14, every regression result for order-one methods seems to have slope 1 when we compare them with the reference line.

Table 4.1 and Table 4.2 show the regression results for order-one numerical methods (EM, TEM, SE, TSE, JE, TJE, JSE and TJSE), which are weak orders of convergence β and intercepts $\log(C)$ where Δ are $\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ and $\frac{1}{64}$ for both JCIR and JCEV models, respectively. The computed weak orders of convergence from order-one numerical methods for JCIR and JCEV models are in the close interval $[0.9052, 1.0254]$ and $[0.9020, 1.0267]$, respectively. Therefore, these order-one numerical methods provide weak order of convergence 1 as the theory says. For JCIR and JCEV models, TJE provides the lowest weak order of convergence, and EM provides the highest weak order of convergence. The value of intercepts for JCIR and JCEV models are in the close interval $[2.1020, 2.6959]$ and $[1.9830, 2.6990]$, respectively. TJE provides the lowest intercept for JCIR model, and TJSE provides the lowest intercept for JCEV model. EM provides the highest intercept for both JCIR and JCEV models. From all of the regression results of order-one methods, the most accurate order-one methods for JCIR and JCEV model are TSE, TJE and TJSE.

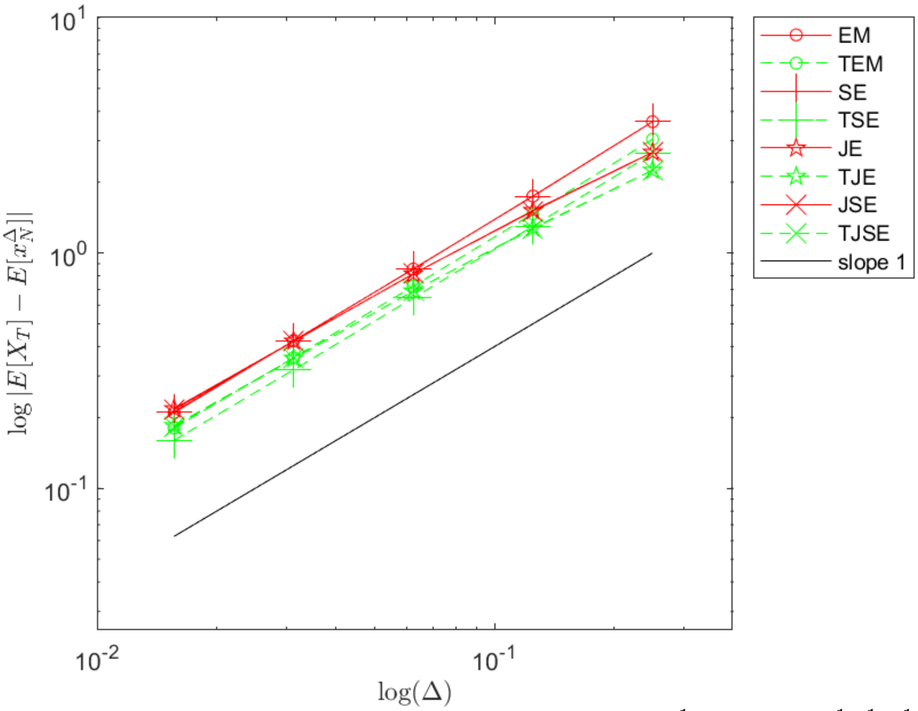


Figure 4.13: Order-one regression results when $\alpha = \frac{1}{2}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$

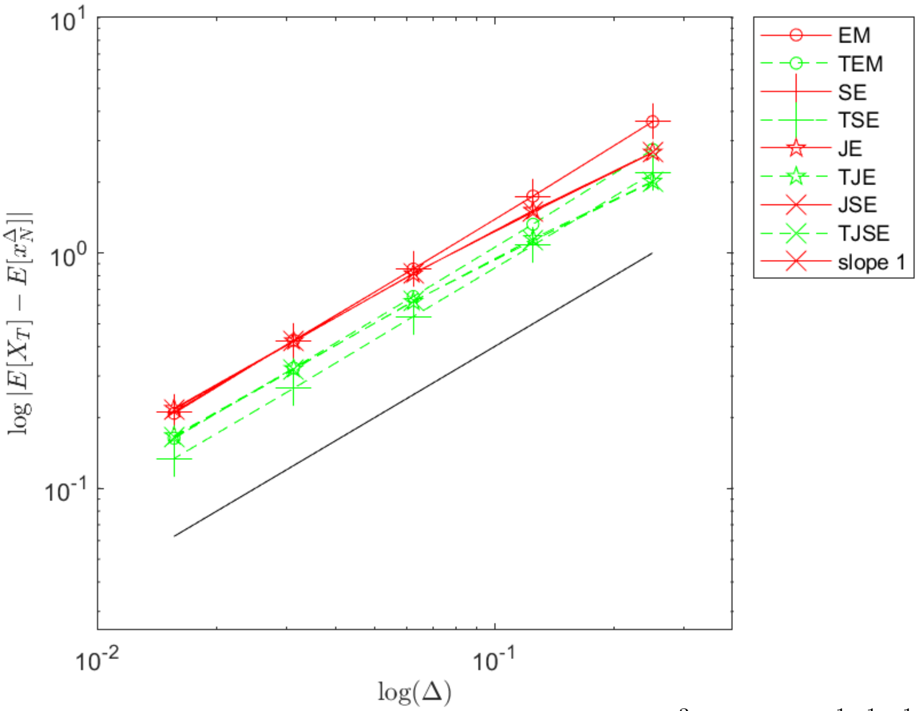


Figure 4.14: Order-one regression results when $\alpha = \frac{3}{4}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$

Numerical scheme	Weak orders of convergence β	Intercepts $\log(C)$
EM	1.0254	2.6959
TEM	1.0184	2.5105
SE	1.0232	2.6904
TSE	1.0124	2.3710
JE	0.9082	2.2801
TJE	0.9052	2.1020
JSE	0.9102	2.2822
TJSE	0.9057	2.1045

Table 4.1: The regression results of order-one schemes when $\alpha = \frac{1}{2}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$

Numerical scheme	Weak orders of convergence β	Intercepts $\log(C)$
EM	1.0267	2.6990
TEM	1.0179	2.4081
SE	1.0231	2.6906
TSE	1.0070	2.1717
JE	0.9074	2.2785
TJE	0.9020	1.9896
JSE	0.9077	2.2734
TJSE	0.9025	1.9830

Table 4.2: The regression results of order-one schemes when $\alpha = \frac{3}{4}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$

4.2.2 Order-two regression results

Since order-two methods provide outliers when $\Delta = \frac{1}{64}$, we choose $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ to find weak orders of convergence for order-two methods. Fig. 4.15 and Fig. 4.16 show order-two regression results with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ for JCIR and JCEV models, respectively. The magenta dash lines are lower than the blue solid lines. This guarantees that for order-two methods, the transformed schemes provide less weak error than their corresponding direct schemes. As for JCIR model, the blue solid lines almost overlap with each other. Therefore, JW, JSW, JD and JSD yield quite indistinguishable weak errors. We can see that TJW provides lowest weak error. As for JCEV model, the regression results from JSD and TJSD provide highest weak errors, and TJW and TJSW provide the lowest weak errors. However, the regression results from JW, JSW and JD are almost overlap with each other. Therefore, JW, JSW and JD yield quite indistinguishable weak errors.

Table 4.3 and Table 4.4 show the regression results for order-two numerical methods (JW, TJW, JSW, TJSW, JD, TJD, JSD and TJSD), which are weak orders of convergence β and intercepts $\log(C)$ where Δ are $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ and $\frac{1}{32}$ for both JCIR and JCEV models, respectively. From Table 4.3 and Table 4.4, there are no order-two methods whose weak orders of convergence reach 2. As for JCIR model, the computed weak orders of convergence from order-two methods are in the close interval $[1.7058, 1.8563]$. JW and JD provide the highest weak orders of convergence, and TJSW provides the lowest weak order of convergence. The intercepts are in the close interval $[0.9929, 1.8150]$. JW and JD provide the highest intercepts, and TJSW provides the lowest intercept. As for JCEV model, the weak orders of convergence from order-two methods are in the close interval $[1.3963, 1.8887]$. JW and JD provide the highest weak orders of convergence, and TJSD provides the lowest weak order of convergence. The intercepts are in the close interval $[0.4258, 1.8727]$. JW and JD provide the highest intercepts, and TJSW provides the lowest intercept.

For JCIR and JCEV models, JW and JD provide the highest weak order of convergence with the same regression result, but their corresponding transformed schemes, TJD and TJW provide difference regression result. TJD provides higher weak order of convergence than TJW, but TJW provides less weak error than TJD.

From all regression results of order-two methods, the most accurate order-two numerical method for JCIR model is TJW, and the most accurate order-two numerical methods for JCEV model are TJW and TJSW.

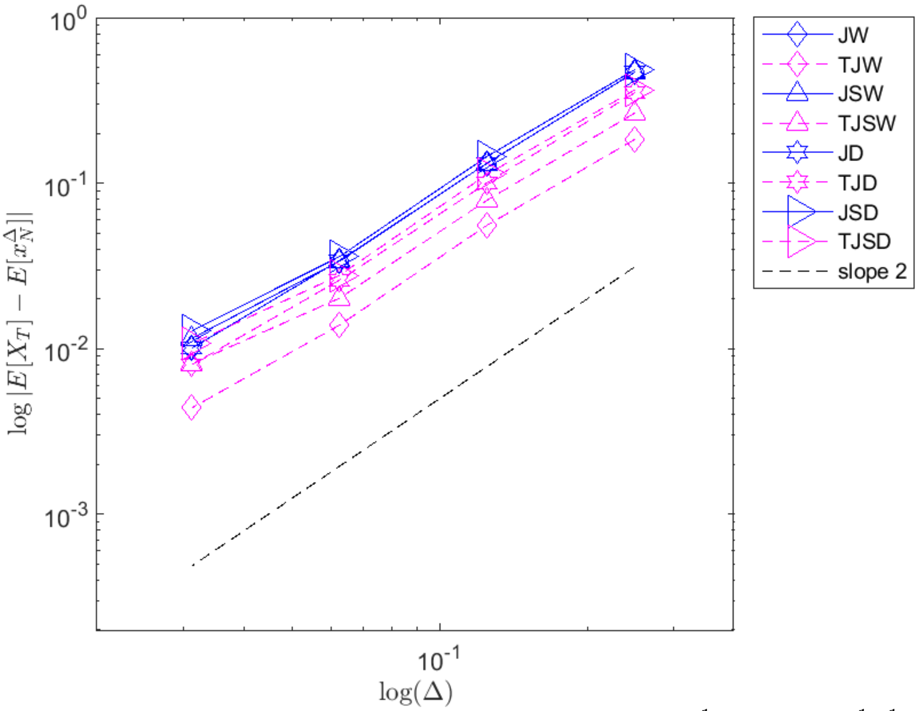


Figure 4.15: Order-two regression results when $\alpha = \frac{1}{2}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$

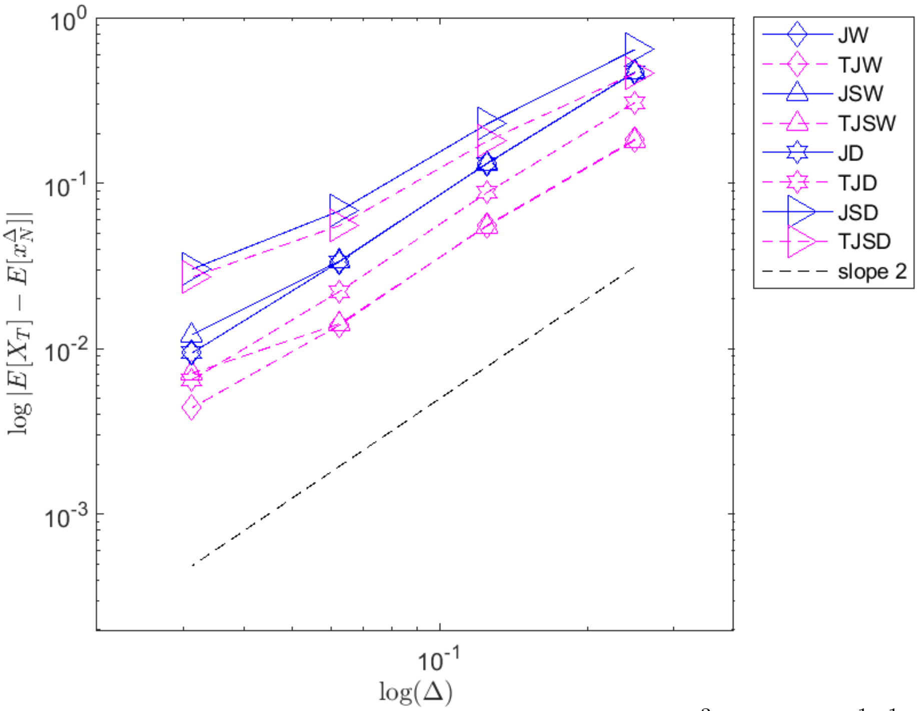


Figure 4.16: Order-two regression results when $\alpha = \frac{3}{4}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$

Numerical scheme	Weak orders of convergence β	Intercepts $\log(C)$
JW	1.8563	1.8150
TJW	1.7923	1.1571
JSW	1.7979	1.7049
TJSW	1.7058	0.9929
JD	1.8563	1.8150
TJD	1.8318	1.4896
JSD	1.7148	1.7718
TJSD	1.7290	1.3694

Table 4.3: The regression results of order-two methods when $\alpha = \frac{1}{2}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$

Numerical scheme	Weak orders of convergence β	Intercepts $\log(C)$
JW	1.8887	1.8727
TJW	1.8156	0.8344
JSW	1.7758	1.6616
TJSW	1.5983	0.4258
JD	1.8887	1.8727
TJD	1.8667	1.4138
JSD	1.6021	1.4962
TJSD	1.3963	1.1392

Table 4.4: The regression results of order-two schemes when $\alpha = \frac{3}{4}$ with $\Delta = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$

4.3 Run times

Table 4.5 and Table 4.6 show the run times of all numerical methods for both JCIR and JCEV models, respectively. Note that the unit of run times in this subsection is $\times 10^3$ seconds.

As for JCIR model, run times for $\Delta = \frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{32}$ and $\frac{1}{64}$ are in the close intervals $[0.3874, 1.5089]$, $[0.7702, 1.6628]$, $[1.3547, 2.7691]$, $[1.4258, 4.9013]$ and $[1.5860, 9.0855]$, respectively. TSE provides lowest run time for $\Delta = \frac{1}{4}$ and $\frac{1}{8}$. JE provides the lowest run time for $\Delta = \frac{1}{16}$ and $\frac{1}{32}$. TJSE provides the lowest run time for $\Delta = \frac{1}{64}$. JD provides the highest run time for $\Delta = \frac{1}{4}$. EM provides the highest run time for $\Delta = \frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{32}$ and $\frac{1}{64}$.

As for JCEV model, run times for $\Delta = \frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{32}$ and $\frac{1}{64}$ are in the close intervals $[0.4034, 1.5530]$, $[0.7813, 1.7240]$, $[1.3731, 2.8058]$, $[1.4747, 4.9687]$ and $[1.6223, 9.1591]$, respectively. TSE provides the lowest run time for $\Delta = \frac{1}{4}$. SE provides the lowest run time for $\Delta = \frac{1}{8}$. JE provides the lowest run time for $\Delta = \frac{1}{16}$. TJE provides the lowest run time for $\Delta = \frac{1}{32}$ and $\frac{1}{64}$. TJW provides the highest run time for $\Delta = \frac{1}{4}$. JSD provides the highest run time for $\Delta = \frac{1}{8}$. EM provides the highest run times for $\Delta = \frac{1}{16}$, $\frac{1}{32}$ and $\frac{1}{64}$.

Fig. 4.17 and Fig. 4.18 show run times of all 16 numerical schemes for both JCIR and JCEV models, respectively. For JCIR and JCEV models, the group of highest-run-time-schemes consists of EM, TEM, SE and TSE, and the group of the lowest-run-time-schemes consists of JE, TJE, JSE and TJSE.

From all of run times result for both JCIR and JCEV models, we can see that for the high values of Δ ($\Delta = \frac{1}{4}, \frac{1}{8}$), the non-jump-adapted schemes, which are EM, TEM, SE and TSE, tend to run faster than the jump-adapted schemes. However, for the value of small Δ ($\Delta = \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$), the jump-adapted schemes, which are JE, TJE, JSE, TJSE, JW, TJW, JSW, TJSW, JD, TJD, JSD and TJSD, tends to run faster than the non-jump-adapted schemes. When Δ become smaller, JE, TJS, JSE and TJSE tend to consume less time than the other schemes. Comparing run time between the simplified schemes

(JSE, JSW, JSD, TJSE, TJSW and TJSD) and the corresponding non-simplified schemes (JE, JW, JD, TJE, TJW and TJD), we find that simplified methods do not significantly reduce run time.

Numerical schemes	Run times ($\times 10^3$ seconds)				
	$\Delta = \frac{1}{4}$	$\Delta = \frac{1}{8}$	$\Delta = \frac{1}{16}$	$\Delta = \frac{1}{32}$	$\Delta = \frac{1}{64}$
EM	1.0682	1.6628	2.7691	4.9013	9.0855
TEM	1.0597	1.6604	2.7155	4.8078	8.8945
SE	0.4433	0.7910	1.5298	3.1057	6.2547
TSE	0.3874	0.7702	1.5486	3.0848	6.1581
JE	1.3226	1.3484	1.3547	1.4258	1.5945
TJE	1.3137	1.3381	1.3762	1.4405	1.5972
JSE	1.3296	1.3368	1.3593	1.4264	1.6445
TJSE	1.3075	1.3236	1.3752	1.4414	1.5860
JW	1.4692	1.5541	1.6931	1.9703	2.5309
TJW	1.5014	1.5843	1.7450	2.0689	2.6781
JSW	1.4892	1.5605	1.7111	2.0094	2.6256
TJSW	1.4312	1.5004	1.6280	1.8761	2.3540
JD	1.5089	1.5872	1.6811	1.9949	2.4479
TJD	1.3537	1.3855	1.4512	1.6093	1.8839
JSD	1.4438	1.5109	1.6549	1.9273	2.4821
TJSD	1.4526	1.4948	1.5996	1.8301	2.2603

Table 4.5: The run time of all 16 schemes when $\alpha = \frac{1}{2}$

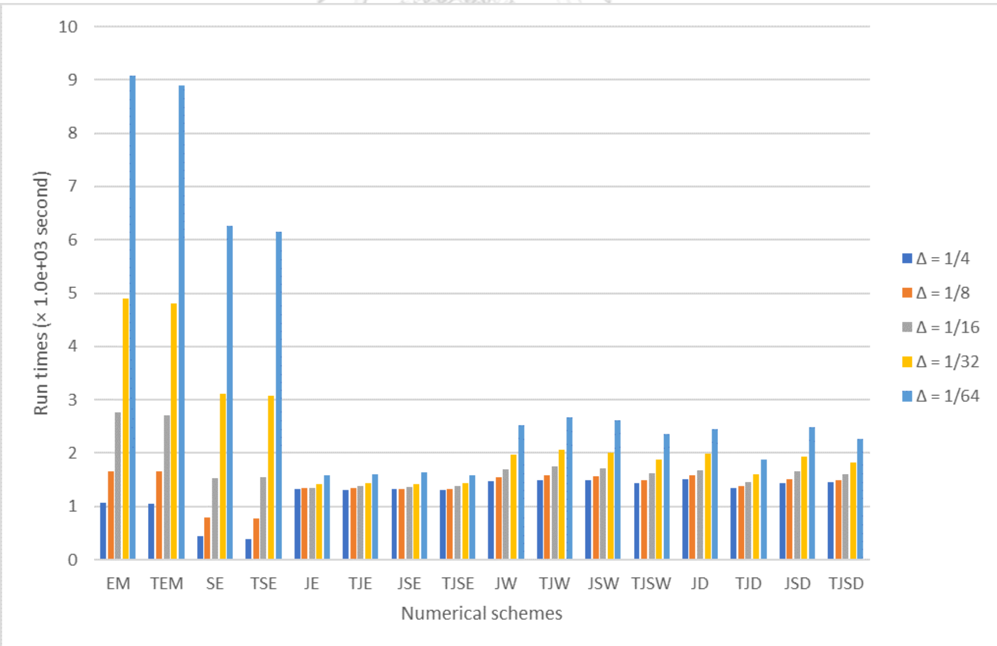


Figure 4.17: Run times when $\alpha = \frac{1}{2}$

Numerical schemes	Run times ($\times 10^3$ seconds)				
	$\Delta = \frac{1}{4}$	$\Delta = \frac{1}{8}$	$\Delta = \frac{1}{16}$	$\Delta = \frac{1}{32}$	$\Delta = \frac{1}{64}$
EM	1.1052	1.6744	2.8058	4.9687	9.1591
TEM	1.0828	1.6688	2.7531	4.8743	8.9742
SE	0.4662	0.7813	1.5661	3.1244	6.2211
TSE	0.4034	0.8037	1.5596	3.0964	6.2498
JE	1.3134	1.3276	1.3731	1.4803	1.7160
TJE	1.3285	1.3478	1.3900	1.4747	1.6223
JSE	1.3015	1.3645	1.3843	1.5107	1.7387
TJSE	1.3727	1.3816	1.4438	1.5520	1.7838
JW	1.5411	1.6190	1.8170	2.1941	2.9708
TJW	1.5530	1.6471	1.8405	2.2040	2.9889
JSW	1.5316	1.6469	1.8649	2.2882	3.1274
TJSW	1.4713	1.5516	1.6972	2.0012	2.6336
JD	1.5417	1.6722	1.8902	2.2681	3.2163
TJD	1.4190	1.4859	1.6026	1.8460	2.3258
JSD	1.0460	1.6100	1.7240	1.9070	2.6100
TJSD	1.4967	1.6609	1.7374	2.0668	2.7347

Table 4.6: The run time of all 16 schemes when $\alpha = \frac{3}{4}$

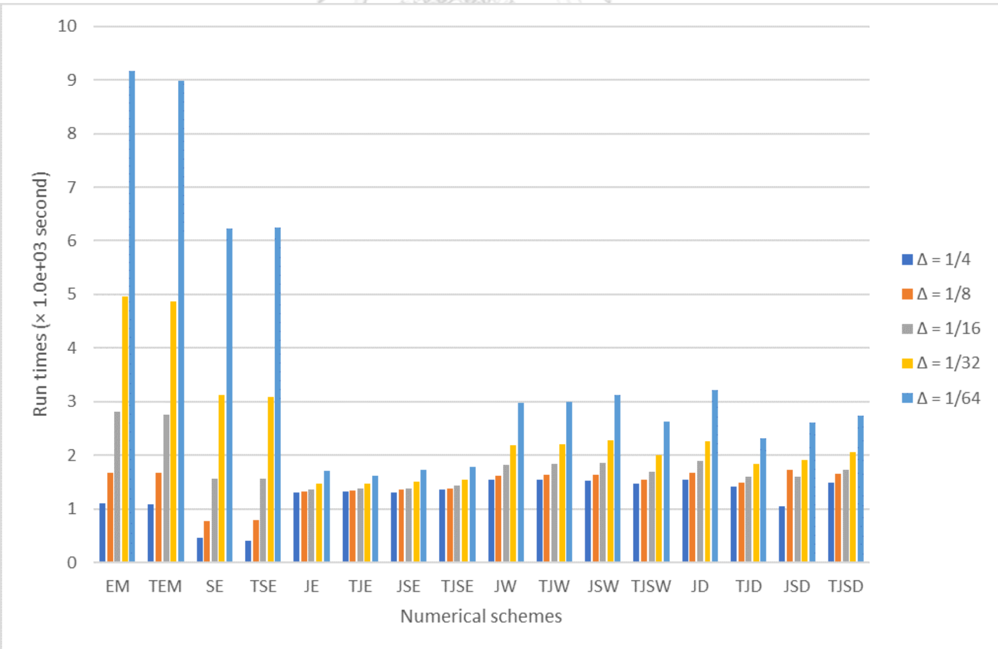


Figure 4.18: Run times when $\alpha = \frac{3}{4}$

CHAPTER V

CONCLUSION

In this chapter, we conclude the results from our simulation and suggest possible future work.

5.1 Conclusions

- The sixteen schemes are valid to approximate for numerical solutions of the JCIR and JCEV models.
- The transformation $f(X_t) = X_t^{1-\alpha}$ reduces the weak error from simulations.
- Simplified methods do not significantly reduce run time when compared with the corresponding non-simplified methods.
- For JCIR model, TJW tends to provide lower weak error than the other schemes.
- For JCEV model, TJW and TJSW tend to provide lower weak error than the other schemes.
- For both JCIR and JCEV model, JD and JW provide the same highest weak order of convergence.
- For both JCIR and JCEV model, JE, TJE, JSE and TJSE provide the lowest run times.
- We suggest TJW or TJSW to simulate numerical solutions for JCIR and JCEV models, because they tend to give lower weak errors than the other schemes and they consume reasonably low run time.

5.2 Future work

In this work, we focus on weak convergence of numerical methods. However, there is another type of convergence for numerical methods called “strong convergence”. To find a strong order of convergence, we need to find the exact solution of the SDE. Unfortunately, we do not have a closed form solution of the SDE (1.3). However, we may approximate the exact solution of the SDE (1.3) and numerically find the strong order of convergence. Moreover, higher order method can be consider to numerically solve the SDE (1.3). As for the model, to make it more realistic, the jump size distribution can be change and the Poisson process can be inhomogeneous, i.e., the intensity rate can be a function varying in time or another stochastic process. Also, other parameters in the model can be modeled by a system of SDE. In addition, the source of natural noises in the model can be changed. For example, a fractional Brownian motion can be used instead of the Wiener process.

REFERENCES

[1] N. Beliaeva and S. Nawalkha. Pricing american interest rate options under the jump-extended constant-elasticity-of-variance short rate models. *Journal of Banking and Finance*, 36:151–163, 2012.

[2] J. C. Cox, J. Ingersoll, and S. A. Ross. A theory of the term structure of interest rates. *Econometrica*, 53:385–406, 1985.

[3] J. C. Cox and S. A. Ross. The valuation of options for alternative stochastic processes. *Journal of Financial Economics*, 3:145–166, 1976.

[4] J. Goodman. *Lecture Note in Stochastic Calculus Notes, Lecture 7.* Department of Mathematics University of California at Davis, April 2007.

[5] M. Johannes. The statistical and economic role of jumps in continuous-time interest rate models. *The Journal of Finance*, 1:227–260, 2004.

[6] N. B. Liberati and E. Platen. Approximation of jump diffusions in finance and economics. *Computational Economics*, 29:283–312, 2007.

[7] R. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Economic Theory*, 3:125–144, 1976.

[8] K. Øksendal. *Stochastic Differential Equations, An Introduction with Applications.* Springer, Fifth Edition, Corrected Printing Springer-Verlag Heidelberg New York, 2003.

[9] E. Platen and D. Health. *Introduction to Quantitative Finance :A Benchmark Approach.* Springer Finance. (Springer, Forthcoming), 2006.

[10] N. Privault. *Notes on Stochastic Finance, Lecture Notes, Stochastic Calculus in Finance II FE6516.* Division of Mathematical Sciences School of Physical and Mathematical Sciences Nanyang Technological University, January 2020.

[11] S. Shkoller. *MAT125B Lecture Notes*. New York University Department of Mathematics, May 2011.

[12] X. Yang and X. Wang. A transformed jump-adapted backward euler method for jump-extended CIR and CEV models. *Numerical Algorithms*, 74:39–57, 2017.

APPENDIX

In appendix, we show Matlab code to simulate for test for positivity preserving and findind weak order of convergence of EM, JEM, TEM and TJE. For the other scheme, we simulate with the same structure of code but difference only in updated numerical schemes.

APPENDIX A : EM simulation for test positivity preserving matlab code

```

randn('state',100)
rand('state',100)

alpha = 0.5;     % 0.5 for CIR, 0.75 for CEV
kappa = 2 ; theta = 1 ; sigma = 1.5 ; gamma = 0.5;
Szero = 2;
lambda = 5; mu_ln = 0; sigma_ln = 0.1;

T = 1;
M = 10000;
mean_Xi = exp(mu_ln + 0.5*(sigma_ln^2)) - 1;

dt_step = [1/4, 1/8, 1/16]; %The size of each step size
%Defind function
a = @(x) kappa*(theta-x);
b = @(x) sigma*(x^alpha);
c = @(x) gamma*x;
Time = [2,4,8];
P_count = zeros(length(Time),length(dt_step));
for k=1:length(Time)
    T = Time(k);
    for p = 1:length(dt_step)
```

```
        dt = dt_step(p);
        N = T/dt;
        count = 0;  % the number of invalid paths (negative at least once)
        for i = 1:M
            Stemp = Szero;
            for j = 1:N
                Winc = sqrt(dt)*randn;
                poi = poissrnd(lambda*dt);
                if poi == 0
                   Xi = 0;
                else
                   Xi = sum(lognrnd(mu_ln,sigma_ln,[1,poi])-ones(1,poi));
                end
                Stemp = Stemp + a(Stemp)*dt + b(Stemp)*Winc + c(Stemp)*Xi;
                if Stemp < 0
                    count = count+1;
                    break;    % go to the next i
                end
            end
        end
        P_count(k,p) = count;
        k
        p
    end
end
disp(P_count)
```

```
eval(sprintf('save P1_EM_0.50_test P_count'));
```

APPENDIX B : JEM simulation for test positivity preserving matlab code

```
randn('state',100)
rand('state',100)

alpha = 0.75;     % 0.5 for CIR, 0.75 for CEV
kappa = 2 ; theta = 50 ; sigma = 0.3; gamma = 0.8;
Szero = 100;
lambda = 5; mu_ln = 0; sigma_ln = 0.1;

M = 10000;        % the number of sample paths

Time = [2,4, 8];
dt_step = [1/4, 1/8, 1/16];
P_count = zeros(length(Time),length(dt_step));
for l=1:length(Time)
    T = Time(l);
    for p = 1:length(dt_step)                    % take various timesteps
        dt = dt_step(p);
        count = 0;
        oldGrid = 0:dt:T;                 %% construct a jump-adapted time discretization
        for k = 1:M
            ei = [];                         %%
            t_jump = 0;                          %%
            while(1)                         %%
                eii = exprnd(1/lambda);      %% exponential random
```

```
            variable (jump length)
        t_jump = t_jump + eii;              %% increase distant of jump time t ????
        if(t_jump > T)                      %% determine that t from exponential is not more than T
            break;                          %%
        else
            ei = [ei eii];                  %% increase the list of jump time
        end
    end
    t_jump = cumsum(ei);                    %% at any t is increase
    newGrid=union(oldGrid,t_jump);          %% combine jump time and the time

    L = length(newGrid)-1;                  % L steps
    dgrid = diff(newGrid);                  %Differences and approximate derivatives
    Stemp = Szero;
    for j = 1:L
        %Winc = sqrt(dgrid(j))*randn();
        if Stemp < 0
            count=count+1;
            break;
        end
        Stemp = Stemp + kappa*(theta - Stemp)*dgrid(j) + sigma*(Stemp^alpha)*sqrt(dgrid(j))*randn();
        if any(t_jump - newGrid(j) == 0)
            Xi = lognrnd(mu_ln,sigma_ln)-1;
```

```
                    Stemp = Stemp + gamma*Stemp*Xi;
                end

            end
        end
        P_count(1,p)= count;

    end
end
disp(P_count)
eval(sprintf('save P3_JEM_0.75 P_count'));
```

APPENDIX C : EM simulation for finding weak order of convergence and running time matlab code

```
rand('state',100)
randn('state',100)
tic
alpha = 0.5;   % 0.5 for CIR, 0.75 for CEV
kappa = 2 ; theta = 50 ; sigma = 0.3; gamma = 0.8;
Szero = 100;
lambda = 5; mu_ln = 0; sigma_ln = 0.1;

T = 1;
M = 10^7;
mean_Xi = exp(mu_ln + 0.5*(sigma_ln^2)) - 1;

dt_step = [1/4,1/8,1/16,1/32,1/64]; %The size of each step size
%Defind function
a = @(x) kappa*(theta-x);
```

```
b = @(x) sigma*(x^alpha);
c = @(x) gamma*x;
S = zeros(length(dt_step),1);
for p = 1:length(dt_step)
    final_Stemp = zeros(M,1);
    dt = dt_step(p);
    N = T/dt;
    for i = 1:M
        Stemp = Szero;

        for j=1:N
            Winc = sqrt(dt)*randn();
            poi = poissrnd(lambda*dt);
            if poi == 0
                Xi = 0;
            else
                Xi = sum(lognrnd(mu_ln,sigma_ln,[1,poi])-ones(1,poi)
                    );
            end

            Stemp = Stemp + a(Stemp)*dt + b(Stemp)*Winc ...
                + c(Stemp)*Xi;
        end
        final_Stemp(i) = Stemp;
    end
    S(p) = mean(final_Stemp);
    p
end

%disp(S)
```

```

Ex_exact = (-kappa*theta + ...
    (kappa*theta + (-kappa+gamma*lambda*mean_Xi)*Szero)*exp((-kappa+gamma*lambda*mean_Xi)*T))...
    /(-kappa + gamma*lambda*mean_Xi);
Serr = abs(S - Ex_exact);
Dtvals = dt_step;
%%

A = [ones(length(dt_step),1), log(Dtvals)']; rhs = log(Serr);
sol = A\rhs;
q = sol(2);
resid = norm(A*sol - rhs);
time = toc;

figure(1)
loglog(Dtvals,Serr,'-*')
saveas(gcf,'G_01_EM_50_10_new.png')
eval(sprintf('save 01_EM50_10_new Serr Dtvals q sol resid time'));
rand('state',100)
randn('state',100)
tic
alpha = 0.5;     % 0.5 for CIR, 0.75 for CEV
kappa = 2 ; theta = 50 ; sigma = 0.3; gamma = 0.8;
Szero = 100;
lambda = 5; mu_ln = 0; sigma_ln = 0.1;

T = 1;
M = 10^7;
mean_Xi = exp(mu_ln + 0.5*(sigma_ln^2)) - 1;
```

```

dt_step = [1/4,1/8,1/16,1/32,1/64]; %The size of each step size
%Defind function
a = @(x) kappa*(theta-x);
b = @(x) sigma*(x^alpha);
c = @(x) gamma*x;
S = zeros(length(dt_step),1);
for p = 1:length(dt_step)
    final_Stemp = zeros(M,1);
    dt = dt_step(p);
    N = T/dt;
    for i = 1:M
        Stemp = Szero;

        for j=1:N
            Winc = sqrt(dt)*randn();
            poi = poissrnd(lambda*dt);
            if poi == 0
                Xi = 0;
            else
                Xi = sum(lognrnd(mu_ln,sigma_ln,[1,poi])-ones(1,poi)
                    );
            end

            Stemp = Stemp + a(Stemp)*dt + b(Stemp)*Winc ...
                + c(Stemp)*Xi;
        end
        final_Stemp(i) = Stemp;
    end
    S(p) = mean(final_Stemp);
```

```
    p
end

%disp(S)

Ex_exact = (-kappa*theta + ...
    (kappa*theta + (-kappa+gamma*lambda*mean_Xi)*Szero)*exp((-
        kappa+gamma*lambda*mean_Xi)*T))...
    /(-kappa + gamma*lambda*mean_Xi);
Serr = abs(S - Ex_exact);
Dtvals = dt_step;
%%

A = [ones(length(dt_step),1), log(Dtvals)']; rhs = log(Serr);
sol = A\rhs;
q = sol(2);
resid = norm(A*sol - rhs);
time = toc;

figure(1)
loglog(Dtvals,Serr,'-*')
saveas(gcf,'G_01_EM_50_10_new.png')
eval(sprintf('save 01_EM50_10_new Serr Dtvals q sol resid time'));
```


APPENDIX D : JEM simulation for finding weak order of convergence and running time matlab code

```
randn('state',100)
rand('state',100)
tic
```

```
alpha = 0.5;     % 0.5 for CIR, 0.75 for CEV
kappa = 2 ; theta = 50 ; sigma = 0.3; gamma = 0.8;
Szero = 100;
lambda = 5; mu_ln = 0; sigma_ln = 0.1;
T = 1;
M = 10^7;        % the number of sample paths

mean_Xi = exp(mu_ln + 0.5*(sigma_ln^2)) - 1;

dt_step = [1/4,1/8,1/16,1/32,1/64]; %The size of each step size
S = zeros(length(dt_step),1); %mean of final Stemp
%Defind function
a = @(x) kappa*(theta-x);
b = @(x) sigma*(x^alpha);
c = @(x) gamma*x;

for p = 1:length(dt_step)                      % take various timesteps
    SM_final = zeros(M,1);
    dt = dt_step(p);

    oldGrid = 0:dt:T;                 %% construct a jump-adapted time discretization
    for k = 1:M
        ei = [];                      %%
        t_jump = 0;                      %%
        while(1)                      %%
            eii = exprnd(1/lambda);   %% exponential random variable (jump length)
            t_jump = t_jump + eii;              %% increase distant
```

```
            of jump time t ????
        if(t_jump > T)                      %% determine that t from exponential is not more than T
            break;                          %%
        else
            ei = [ei eii];                  %% increase the list of jump time
        end
    end
    t_jump = cumsum(ei);                    %% at any t is increase
    newGrid=union(oldGrid,t_jump);          %% combine jump time and the time

    L = length(newGrid)-1;                  % L steps
    dgrid = diff(newGrid);                  %Differences and approximate derivatives
    Stemp = Szero;
    for j = 1:L
        %Winc = sqrt(dgrid(j))*randn();

        Stemp = Stemp + a(Stemp)*dgrid(j) + b(Stemp)*sqrt(dgrid(j))*randn();
        if any(t_jump - newGrid(j+1) == 0)
            Xi = lognrnd(mu_ln,sigma_ln)-1;
            Stemp = Stemp + c(Stemp)*Xi;
        end
    end
    SM_final(k) = Stemp;
end
S(p) = mean(SM_final);
```

```
    p
end
Ex_exact = (-kappa*theta + ...
            (kappa*theta + (-kappa+gamma*lambda*mean_Xi)*Szero)*exp
                ((-kappa+gamma*lambda*mean_Xi)*T))...
           /(-kappa + gamma*lambda*mean_Xi);
Serr = abs(S - Ex_exact);
Dtvals = dt_step;
%%

A = [ones(length(dt_step),1), log(Dtvals)']; rhs = log(Serr);
sol = A\rhs;
q = sol(2);
resid = norm(A*sol - rhs);
figure(3)
loglog(Dtvals,Serr,'-*')
saveas(gcf,'G_03_JEM_50_10_IM.png')
eval(sprintf('save 03_JEM50_10_IM Serr Dtvals q sol resid time '))
    ;
```


APPENDIX E : TEM simulation for test positivity preserving matlab code

```
randn('state',100)
rand('state',100)

alpha = 0.5;     % 0.5 for CIR, 0.75 for CEV
kappa = 2 ; theta = 50 ; sigma = 0.3; gamma = 0.8;
Szero = 100;
lambda = 5; mu_ln = 0; sigma_ln = 0.1;

```

```
M = 10000;        % the number of sample paths

Time = [2,4, 8];
dt_step = [1/4, 1/8, 1/16];
P_count = zeros(length(Time),length(dt_step));

for k = 1:length(Time)
    T = Time(k);
    for p = 1:length(dt_step)
        dt = dt_step(p);
        N = T/dt;
        count = 0;
        for i = 1:M
            Stemp = Szero^(1-alpha);
            for j=1:N
                if Stemp<0
                    count=count+1;
                    break;
                end
                Winc = sqrt(dt)*randn();
                poi = poissrnd(lambda*dt);
                if poi == 0
                    Xi = 0;
                else
                    Xi = sum(lognrnd(mu_ln,sigma_ln,[1,poi])-ones(1,
                        poi));
                end
                Stemp = Stemp+(0.5*(1-alpha)*(-alpha)*(Stemp^((-
                    alpha-1)/(1-alpha))*abs(sigma*Stemp^(alpha/(1-
                    alpha)))^2 ...
```

```
                    +kappa*(theta-Stemp^(1/(1-alpha)))*(1-alpha)*(
                        Stemp^((-alpha)/(1-alpha))))*dt ...
                    + sigma*(1-alpha)*Winc;
                if poi  = 0
                    Stemp = Stemp +((Stemp^(1/(1-alpha))+gamma*Stemp
                        ^((1/(1-alpha)))*Xi)^(1-alpha)-Stemp)*poi;
                end

            end
        end
        P_count(k,p)=count;
    end
end
disp(P_count)
eval(sprintf('save PT1_TEM_0.5 P_count'));
```

APPENDIX F : TJE simulation for test positivity preserving matlab code

```
randn('state',100)
rand('state',100)

alpha = 0.75;     % 0.5 for CIR, 0.75 for CEV
kappa = 2 ; theta = 50 ; sigma = 0.3; gamma = 0.8;
Szero = 100;
lambda = 5; mu_ln = 0; sigma_ln = 0.1;

M = 10000;        % the number of sample paths

Time = [2,4,8];
dt_step = [1/4, 1/8, 1/16];
```

```
P_count = zeros(length(Time),length(dt_step));
a = @(y) 0.5*(alpha^2 - alpha)*sigma^2/y ...
    + kappa*(theta- y^(1/(1-alpha)))*(1-alpha)*y^(-alpha/(1-alpha));
b = sigma*(1-alpha);
c = @(y,Xi) ((y^(1/(1-alpha))*(1+gamma*Xi)).^(1-alpha)-y);

for l = 1:length(Time)
    T = Time(l);
    for p = 1:length(dt_step)                              % take various timesteps
        dt = dt_step(p);
        count = 0;
        oldGrid = [0:dt:T];                  %% construct a jump-adapted time discretization
        for k = 1:M
            ei = [];                         %%
            t_jump = 0;                          %%
            while(1)                         %%
                eii = exprnd(1/lambda);      %% exponential random variable (jump length)
                t_jump = t_jump + eii;                %% increase distant of jump time t ????
                if(t_jump > T)                    %% determine that t from exponential is not more than T
                    break;                   %%
                else
                    ei = [ei eii];           %% increase the list of jump time
                end
```

```
            end
            t_jump = cumsum(ei);              %% at any t is increase
            newGrid=union(oldGrid,t_jump);       %% combine jump time and the time

            L = length(newGrid)-1;          % L steps
            dgrid = diff(newGrid);          %Differences and approximate derivatives
            Stemp = Szero^(1-alpha);
            for j = 1:L
                Winc = sqrt(dt)*randn();
                Stemp = Stemp + a(Stemp)*dt + b*Winc;
                if any(t_jump - newGrid(j) == 0)        %% add a jump if step j is a jumpt time
                    Xi = lognrnd(mu_ln,sigma_ln)-1;
                    Stemp = Stemp + c(Stemp,Xi);
                end
                if Stemp^(1/(1-alpha)) < 0
                    count=count+1;
                    break;
                end
            end

        end
        P_count(1,p) = count;
    end
end
eval(sprintf('save PT3_TJEM_0.75 P_count'));
disp(P_count)
```

APPENDIX G : TEM simulation for finding weak order of convergence and running time matlab code

```
randn('state',100)
rand('state',100)
tic
alpha = 0.5;     % 0.5 for CIR, 0.75 for CEV
kappa = 2 ; theta = 50 ; sigma = 0.3; gamma = 0.8;
Szero = 100;
lambda = 5; mu_ln = 0; sigma_ln = 0.1;

T = 1;
M = 10^7;
mean_Xi = exp(mu_ln + 0.5*(sigma_ln^2)) - 1;

dt_step = [1/4, 1/8, 1/16, 1/32, 1/64]; %The size of each step size
%Defind function
a = @(y) 0.5*(alpha^2 - alpha)*(sigma^2)/y ...
        + kappa*(theta- (y^(1/(1-alpha))))*(1-alpha)*(y^(-alpha/(1-alpha)));
b =  sigma*(1-alpha);
c = @(y) y;
S = zeros(length(dt_step),1);
for p = 1:length(dt_step)
    final_Stemp = zeros(M,1);
    dt = dt_step(p);
    N = T/dt;
    for i = 1:M
        Stemp = Szero^(1-alpha);
```

```
        for j=1:N
            Winc = sqrt(dt)*randn();
               poi = poissrnd(lambda*dt);
               if poi == 0
                   Xi = 0;
               else
                   Xi = lognrnd(mu_ln,sigma_ln,[1,poi])-ones(1,poi);
                   Xi_sum = 0;
                   for n = 1:length(Xi)
                       Xi_sum = Xi_sum + ((1+gamma*Xi(n))^(1-alpha)-1);
                   end
               end
               Stemp = Stemp + a(Stemp)*dt + b*Winc + c(Stemp)*Xi_sum;
        end
        final_Stemp(i) = Stemp^(1/(1-alpha));
    end
    S(p) = mean(final_Stemp);
    p
end
%disp(S)

Ex_exact = (-kappa*theta + ...
            (kappa*theta + (-kappa+gamma*lambda*mean_Xi)*Szero)*exp((-kappa+gamma*lambda*mean_Xi)*T))...
           /(-kappa + gamma*lambda*mean_Xi);
Serr = abs(S - Ex_exact);
Dtvals = dt_step;
```

```
%%

A = [ones(length(dt_step),1), log(Dtvals)']; rhs = log(Serr);
sol = A\rhs;
q = sol(2);
resid = norm(A*sol - rhs);
time = toc;

figure(11)
loglog(Dtvals,Serr,'-*')
saveas(gcf,'G_OT1_TEM_0.50_10_IM1.png')
eval(sprintf('save OT1_TEM50_10_IM1 Serr Dtvals q sol resid time')
    );
```

APPENDIX H : TJE simulation for finding weak order of convergence and running time matlab code

```
randn('state',100)
rand('state',100)

tic
alpha = 0.5;     % 0.5 for CIR, 0.75 for CEV
kappa = 2 ; theta = 50 ; sigma = 0.3; gamma = 0.8;
Szero = 100;
lambda = 5; mu_ln = 0; sigma_ln = 0.1;
M = 10^7; T =1;
mean_Xi = exp(mu_ln + 0.5*(sigma_ln^2)) - 1;

dt_step = [1/4, 1/8, 1/16, 1/32, 1/64]; %The size of each step
    size
```

```
S = zeros(length(dt_step),1); %mean of final Stemp
%Defind function
a = @(x) 0.5*alpha*(alpha-1)*(sigma^2)/x + kappa*(1-alpha)*(theta*(x^(-alpha/(1-alpha))) - x);
b = @(x) sigma*(1-alpha);
c = @(y,Xi) (y^(1/(1-alpha))*(1+gamma*Xi))^(1-alpha)-y;

for p = 1:length(dt_step)                          % take various timesteps
    SM_final = zeros(M,1);
    dt = dt_step(p);

    oldGrid = 0:dt:T;                    %% construct a jump-adapted time discretization
    for k = 1:M
        ei = [];                         %%
        t_jump = 0;                          %%
        while(1)                         %%
            eii = exprnd(1/lambda);      %% exponential random variable (jump length)
            t_jump = t_jump + eii;               %% increase distant of jump time t ????
            if(t_jump > T)                   %% determine that t from exponential is not more than T
                break;                   %%
            else
                ei = [ei eii];           %% increase the list of jump time
            end
        end
```

```
        t_jump = cumsum(ei);                %% at any t is increase
        newGrid=union(oldGrid,t_jump);      %% combine jump time and the time

        L = length(newGrid)-1;          % L steps
        dgrid = diff(newGrid);          %Differences and approximate derivatives
        Stemp = Szero^(1-alpha);
        for j = 1:L
            Winc = sqrt(dgrid(j))*randn();
            Stemp = Stemp + a(Stemp)*dgrid(j) + b(Stemp)*Winc;
            if any(t_jump - newGrid(j+1) == 0)
                Xi = lognrnd(mu_ln,sigma_ln)-1;
                Stemp = Stemp + c(Stemp,Xi);
            end
        end
        SM_final(k) = Stemp^(1/(1-alpha));
    end
    S(p) = mean(SM_final);
    p
end
Ex_exact = (-kappa*theta + ...
            (kappa*theta + (-kappa+gamma*lambda*mean_Xi)*Szero)*exp((-kappa+gamma*lambda*mean_Xi)*T))...
           /(-kappa + gamma*lambda*mean_Xi);
Serr = abs(S - Ex_exact);
Dtvals = dt_step;
%%

A = [ones(length(dt_step),1), log(Dtvals)']; rhs = log(Serr);
```

```
sol = A\rhs;
q = sol(2);
resid = norm(A*sol - rhs);
time = toc;

figure(14)
loglog(Dtvals,Serr,'-*')
saveas(gcf,'G_OT3_TJEM_50_10_IM_new.png')
eval(sprintf('save OT3_TJEM50_10_IM_new Serr Dtvals q sol resid
    time'));
```

BIOGRAPHY

Name	Mr. Purin Klunklar
Date of Birth	2 November 1994
Place of Birth	Chonburi, Thailand
Education	B.Sc. (Mathematics), Mahidol University, 2016