2015-02-03

European and American Option Pricing with a Geometric Markov Renewal Process Underlying Asset Model

Moyer, Zachary

UNIVERSITY OF CALGARY

European and American Option Pricing with a Geometric Markov Renewal Process Underlying Asset Model

by

Zachary Moyer

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN MATHEMATICS AND STATISTICS

CALGARY, ALBERTA

JANUARY, 2015

Abstract

We present methods for pricing of American and European options under a Geometric Markov Renewal Process (GMRP) as the underlying asset model. We provide a detailed overview of the GMRP. Discussions of Markov processes, Geometric Brownian Motion, and GMRP approximation techniques are presented. We discuss the Aase trading model, with a MATLAB implementation. We discuss the Black-Scholes and binomial Cox-Ross-Rubinstein formulas for European and American options. We present results on Fixed Time Increments GMRP, with a derivation of a method for a limiting case of Fixed Time Increments GMRP (applicable to perpetual American options), complete with MATLAB implementations. We also present a MATLAB implementation for the pricing of European options under GMRP with an arbitrary jump distribution. We discuss diffusion and normal deviated approximations of a GMRP, and present MATLAB implementations for pricing American and European options. We follow this with a discussion of a Poisson approximation of a security market. A literature review is presented, together with an appendix including our MATLAB implementations. Conclusions and recommendations for future research directions conclude the paper.

Acknowledgments

I would like to express my deep gratitude to Dr. Anatoliy Swishchuk for his excellent instruction and supervision, not only in my Master's thesis and research, but in his very well-taught and well-organized courses in the University of Calgary's program in Mathematical Finance and Risk Management. His enthusiasm was always inspirational and always motivated me to do my best. Also, I must thank his lovely wife Dr. Mariya Swishchuk for accepting my numerous late-night phone calls in order to speak with Dr. Anatoliy Swishchuk.

I would also like to give my thanks to Dr. Yuriy Zinchenko for his excellent summary of optimization techniques in MATLAB, which proved very valuable to my research. Also, Dr. Tony Ware, Dr. Deniz Sezer, and Dr. Larry Bates were always available for excellent discussions on financial asset models. I also give my thanks to my excellent professors Dr. Keith Nicholson, Dr. Mohammed Aiffa, Dr. Paul Binding, Dr. Eugene Couch, Dr. Clifton Cunningham, Dr. Elena Braverman, Dr. Viena Stastna, Dr. Gordana Dmitrasinovic-Vidovic, Dr. Alex Brunyi, Dr. Chao Qui, Dr. Dandong Feng, Dr. Wenyuan Liao, Dr. Miloslav Nosal, Dr. Murray Burke, Dr. Bingrui Sun, and the late Dr. Richard Mollin, who inspired me from my first year to follow a path in Mathematics. My preparation of this thesis was also greatly aided by the constant kind and helpful reminders and excellent secretarial skills provided by Mrs. Yanmei Fei, to whom I give much thanks! Also the Dean of Science Dr. Bart Hicks for his excellent advise during my selection of Mathematical Finance and Risk Management when I entered my program, the Dean of Science Dr. Ken Barker for his guidance through the regulations of my degree stream, and Dr. Edward Lozowski for our many interesting conversations on statistical methods.

I also would like to thank Portfolio Manager Ken Corner and CIO Tim Pickering of Auspice Capital Advisors for the extremely valuable lessons I learned in practical financial asset modelling during my internship there. Mr. Corner's excellent advise and guidance served to direct both my mathematical focus and programming techniques to a new level of practicality, efficiency, and applicability. Thank you also to the late Hugo Graumann of the Department of Physics and Astronomy, who taught me everything I needed to know about computers.

In addition, I greatly appreciate the ever-present, ever-engaging mathematical discussions from my friends and fellow students Michelle, Alihan, Heng, Angela, Lucy, Shixian, Siying, Qicheng,

Ji, Vladimir, Hongyan, Jialin, Yuan, Wanqing, Majid, Hui, Wenyan, Kunlin, Chihsien, and many others. Also, for always being there during my valuable breaks from my research, Chihiro, Irene, Yuki, Yoko, Ihiro, Ai, Airi, Reiko, Akari, Lin, and my other dear friends.

I also express my great appreciation for the financial support from the University of Calgary in the form of scholarships. These were not only financially helpful, but were extremely rewarding and encouraging along my path.

I also wish to express my sincere thanks to my family who has provided constant support, in particular my great aunt Mabel, for her help in everything; in countless little things day by day.

Dedication

To my great-grandmother Annie and my great aunt Mabel.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The Geometric Markov Renewal Process (GMRP) is a recent and novel model for the behavior of assets in the world of finance. Its strength lies in the fact that it utilizes an underlying Markov chain to simulate the effect of a market being in various "states". For example, a 3-state GMRP model may simulate the market being in 3 distinct behavior patters: a "good" state, where prices are rising, a "normal" state, where prices are stable, and a "poor" state, when prices are falling. Indeed, regime-switching models are currently used extensively in modeling and simulating the dynamics of energy prices, especially in creating models of volatility. Volatility swaps with energy-based or highly energy-correlated stocks as the underlying asset often use a 2-state regime-switching model in simulation attempts. Such models are invaluable not only to volatility swap traders, but also to risk managers who wish to have a grasp on profitability arising from volatility changes, and to guard against any negative results in portfolio management arising from such behavior.

Trend analysts have also expressed interest in GMRP modeling, as it has an ability to at least exhibit, if not fully capture, trends arising from phenomena like buying or selling sprees. Such phenomena are possible to arise using a GMRP model by the proper choice of states and parameters.

1.2 Jump-Based Asset Models

In this section we outline the three principle models of jump processes examined in this thesis. We state them in increasing order of generality.

A commonly used model for an asset price is the Binomial Model used by Cox, Ross, and Rubinstein [5], with a process defined as such:

$$S_n = S_0 \prod_{k=1}^{n} (1+\rho_k) \tag{1.2.1}$$

where we have $S_0 > 0$ as the starting stock price, and

$$\rho_k = \begin{cases} u & with\ probability\ \ p \\ d & with\ probability\ \ (1-p) \end{cases} \quad \text{and } -1 < d < u, 0 < p < 1$$

One potentially unrealistic of aspect of the above model is that at each point, the probability of the "up" state "u" is always the same. We can capture varying probabilities going "up" or "down" by introducing states to the market, where those up or down probabilities vary among these states. For the Binomial Model we have established results for the pricing of (dividend-free) European and American calls and puts.

Another jump asset model is the Geometric Compound Poisson Process (GCPP) [1]:

$$S_t = S_0 \prod_{k=1}^{N(t)} (1+Y_k) \tag{1.2.2}$$

where we have $N(t)$ is a Poisson process, Y_k are i.i.d. random variables independent of $N(t)$, and S_0>0 is again the starting stock price. This model is more general than the previous in the following ways:

1. We have the ability to model the jump times with a Poisson distribution, eliminating the unrealistic rigidity of known jump times from the Binomial Model.

2. The i.i.d. random variables Y_k allow for greater flexibility of the percentage gain or loss at each jump point.

Because of the power to make the jump times Poisson distributed and the percentage gain/loss flexibility, the GCPP is considered an improvement over the Binomial Model. For the GCPP, we have risk-neutral pricing formulas for (dividend-free) European options. No formulas for American options are presented in the Aase paper [1].

An even more general model than the previous GCPP is the Geometric Markov Renewal Process (GMRP), which is defined as below [21]:

$$S_t = S_0 \prod_{k=0}^{\nu(t)} (1 + \rho(x_k)) \tag{1.2.3}$$

where $\nu(t)$ is a counting process, $x(t) := x_{\nu(t)}$ is a semi-Markov process, S_0 is the starting stock price, $\rho(x)$ is a bounded, continuous function on states of the Markov chain X, and we have that $\rho(x) > -1$. The GMRP is more general than the previous model in the following key ways:

1. The embedded Markov chain introduces dependence on the present state, something lacking in both the GCPP and the Binomial model.

2. The counting process $\nu(t)$ is general; it need not be the Poisson model as is the case with GCPP.

3. There exist pricing schemes for (dividend-free) European calls and puts and American calls and puts (via GBM approximations, see Sections 8.5 and 8.6).

As we can see, the GCPP is trivially a special case of the GMRP, as any i.i.d. sequence of random variables (in the case of GCPP, the sequence Y_k) is trivially a Markov chain. The GMRP is very useful powerful in modelling not only asset prices, but even underlying aspects of markets, such as trends. For example, a 3-state Markov chain may represent good, neutral, and poor states of the market for an asset. The power of the GMRP lies in the Markov chain's ability to assign probabilities based on the current state, which allows for easy but practical calculations for events such as (in a good-neutral-poor market model) the chance of the market switching out of its current state into another state.

We will elaborate on the GMRP in Chapter 7.

Outline of the Thesis

In Chapter 3, we will cover necessary information and concepts required for the study of the models presented in this thesis. A literature review is in Chapter 2. Chapter 6 will outline the Aase model of continuous trading, and also present some numerical results. Chapter 4 will examine methods of options pricing for European-style options modelled with a Geometric Brownian Motion (GBM) via the celebrated Black-Scholes formulas. Chapter 5 will present the first of the jump asset models we discuss: the Binomial Model, complete with the Cox-Ross-Rubenstein method of option pricing using this model, with applications to American-style options; in particular, how to convert from a GBM to the Binomial scheme in order to use the Cox-Ross-Rubinstein formulas. Chapter 7, as mentioned earlier in this chapter, will address discuss details on the Geometric Markov Renewal Process. Additionally, numerical results in the form of MATLAB codes are presented, as well as new, original research into solving an integral for options valuation in a certain case, together with a MATLAB implementation of this method. The following chapter, Chapter 8, outlines details on GMRP approximations necessary for upcoming chapters. Within this Chapter, the sections 8.5 and 8.6 discuss and present numerical results on, respectively, the Diffusion and Normal Deviated GMRP approximation schemes. In both cases we present numerical results for the Ergodic case and the Double Averaged Markov Process scenarios in the form of MATLAB codes implementing the approximation methods and using aforementioned schemes for both European-style and American-style options pricing. In Chapter 9 we will see related results for the Poisson approximation, a presentation of a Levy process to describe an asset in this formulation, and in particular, how it relates to market incompleteness. An overview of possible future paths for research can be found in Chapter 10. In the Appendices, one may find a detailed description of how to calculate a Markov potential operator, together with MATLAB codes for the implementation of many options pricing algorithms presented. These implementations and any theory that is not cited which is used in setting up these implementations in this thesis are original work.

Chapter 2

Literature Review

On the general issue of pricing of options, there are two important results; the Black-Scholes formulas (for a Geometric Brownian Motion underlying asset model) and the Cox-Ross-Rubenstein formulas (for a binomial underlying asset model). In 1973, Black and Merton presented one of the most famous papers of all in the history of financial mathematics. Using the No-Arbitrage Principle, they derive a theoretical valuation formula for options. Applications to corporate liabilities such as common stock, corporate bonds, and warrants are included. Also, they present a usage of the formula to find the discount that should be applied to a corporate bond due to the possibility of default [3]. Robert Merton published a paper in 1976 outlining specific and important details for the construction of a replicating portfolio in the 1973 Black/Merton paper. Furthermore, he presents an option pricing formula for the more general case where the underlying stock may have jumps as well as a continuous part; this formula retains the properties of independence from investor preference of knowledge of the expected return of the asset. Also, results for corporate liabilities are included [13]. In their influential 1979 paper, Cox, Ross, and Rubinstein presented a binomial model based approach to the valuation of options. The paper contains the well-known Black-Scholes model as a special limiting case utilizing a straight-forward derivation. The well-known Cox-Ross-Rubinstein formulas for pricing of options are presented. Methods of generalization of the model are presented. A simple procedure for valuing American-style options is presented, namely, the accredited method of "backwardation" valuation for American options[5]. In 2003, Sick and Elliot presented results for a regime-switching model for modelling electricity price risk

by using a similar concept to the Geometric Markov Renewal Process. They used a Markov process to model the number of large generators of electricity on line in the Alberta electricity pool [6].

On the more specific matter relevant to the Geometric Markov Renewal Process (GMRP) formulation dealt with in this thesis, we will now examine the literature relevant to Markov processes and the research into the GMRP, as well as the pricing of options under this model. Pyke published the first paper on Markov renewal processes in 1961 in his paper dealing with basic definitions and some preliminary results on them [15]. Pyke followed this up with another paper investigating more accurately the case for Markov renewal processes having a finite number of states. He presents explicit expressions for the distribution functions of first passage times, and also for the marginal distribution function of the corresponding Semi-Markov process. He presents double generating functions are obtained for the distribution functions of the associated Nj-processes. Pyke discusses the limiting behavior of a Markov Renewal process and completely derives the stationary probabilities. He provides several examples [16]. In a 1963 paper, Jewell investigates programming over a Markov-renewal process - in which the intervals between transitions of a system from state i to state j are independent samples from a distribution that may depend upon both i and j. With a given reward structure and decision mechanism that influences both rewards and the Markov-renewal process itself, Jewell formulates the problem as follows: to select alternatives at each transition so as to maximize total expected reward. He investigates both finite and infinite-return models. Jewell indicates extensions to non-ergodic processes. Also, he presents special results for the two-state process. He concludes with an example of machine maintenance and repair is used to illustrate the generality of the models and special problems that may arise. [7]. In 1964, Pyke and Schaufele present a paper dealing with the Doeblin Ratio limit laws, the weak and strong laws of large numbers, and the Central Limit theorem for Markov Renewal processes. They provide a general definition of these processes, together with means and variances of random variables associated with recurrence times in computational examples, with stronger results in the special case of Markov chains [17]. Moore and Pyke presented a paper in 1968 dealing with the estimation of the transition distributions of a Markov renewal process with finitely many states. They use a natural estimator for the transition distributions is defined and show that it is consistent. They also derive limiting distributions for this estimator. They also develop a density for a Markov

renewal process in order to permit the definition of maximum likelihood estimators for a renewal process and for a Markov renewal process in particular [14].

Anatoliy Swishchuk and Shafiqul Islam formally presented the Geometric Markov Renewal Process in their 2010 paper. They present the geometric Markov renewal processes as a model for a security market and study this processes in a diffusion approximation scheme. Weak convergence analysis and rates of convergence of ergodic geometric Markov renewal processes in diffusion scheme are investigated, with detailed results presented. They provide results for European call option pricing formulas in the case of ergodic, merged, and double-averaged diffusion geometric Markov renewal processes. [21]. In their 2011 follow-up paper, Swishchuk and Islam present extended results for averaged, merged, and double averaged normal deviated geometric Markov renewal processes. Weak convergence analysis and rates of convergence of ergodic Geometric Markov Renewal Processes are also presented, together with Martingale properties and infinitesimal operators of geometric Markov renewal processes. Also, they derive a Markov renewal equation for expectation. They include an example in the case of two ergodic classes. In this paper, they also consider a generalized binomial model for a security market induced by a position dependent random map, viewed as a special case of a Geometric Markov Renewal Process [21].

In 2013, Limnios, Swishchuk, et. al. introduce discrete-time semi-Markov random evolutions (DTSMREs), and study the asymptotic properties, namely, averaging, diffusion approximation, and diffusion approximation with equilibrium by the martingale weak convergence method associated with this formulation. Controlled DTSMREs are introduced and the authors derive Hamilton–Jacobi–Bellman equations for them. Applications are also presented [12]. Islam and Swishchuk published another paper in 2013 introducing the Poisson averaging scheme for the Geometric Markov Renewal Process, and they present European call option pricing formulas [22]. Finally, the accumulated results of Swishchuk, Islam, Limnios, and others on the Geometric Markov Renewal Process are accumulated and presented in a unified format in a 2013 book authored by Swishchuk and Islam [23].

Chapter 3

Necessary Preliminaries

3.1 Markov Processes

3.1.1 Markov Processes and Markov Chains [8, 18]

A Markov chain, named after the Russian mathematician Andrei Markov, is random evolution model that undergoes transitions from one state to another on a "state space". It is a random process exhibiting the so-called memoryless property; i.e., the next state depends only on the current state and not on the sequence of events that preceded it. This is called the Markov property. Markov chains have numerous applications as statistical models of real-world processes, and extend far beyond financial mathematics in their usage.

Definition 3.1. (Markov Process).

A process is said to be Markov if

$$P\{a < X_t \leq b | X_{t_1} = x_1, X_{t_2} = x_2, ..., X_{t_n} = x_n\} = P\{a < X_t \leq b | X_{t_n} = x_n\} \tag{3.1.1}$$

for $t_1 < t_2 < ... < t_n < t$. Here we have denoted by X the state space of the Markov process.

The following probability is called the transition probability function.

$$P(x, s; t, A) = P\{X_t \in A | X_s = x\},\ t > s \tag{3.1.2}$$

Definition 3.2. (Markov Chain)

A Markov chain is a a Markov process where we have that the number of states is countable or finite. In the case of a finite state space, the information about transition probabilities of the Markov chain can be stored in a matrix P.

$$P = (p_{ij})_{i,j \in X} \tag{3.1.3}$$

Here $0 \leq p_{ij} \leq 1$ for each i, j and $P(i,A) = \sum_{j \in A} p_{ij}$for and $A \subseteq X$.

The n-step stochastic kernel P^n is defined as follows:

$$P^n(x,A) = \int_X \int_X ... \int_X P(x_0, dx_1)P(x_1, dx_2)...P(x_{n-2}, dx_{n-1})P(x_{n-1}, A) \tag{3.1.4}$$

Here there are exactly n integrals $\int_X$ for all n of the states, $x = x_0$.

Definition 3.3. (Time-homogeneous Markov process)

A stochastic process $x(t)$ is called a time-homogenous Markov process, if, for any fixed $s,t \in \mathbb{R}_+$, $B \in \varepsilon$, we have that the following holds almost surely:

$$P(x(t+s) \in B|F_s) = P(x(t+s) \in B|x(s)) = P_t(x(s), B) \tag{3.1.5}$$

When 3.1.5 holds for any F -stopping time τ, instead of a deterministic time, the Markov process $x(t)$ is called a strong Markov process.

3.1.2 Markov Renewal Processes [21]

Definition 3.4. (Semi-Markov Kernel)

Given a probability space (X, χ, μ), a function $Q : X \times \chi, R^+ \to [0, \infty)$ is called a semi-Markov kernel if it has these properties:

1. $Q(x, ., f) : X \times R^+ \to [0, \infty)$ is a measurable function

2. $Q(.,.,t) : X \times \chi \to [0,\infty)$ is a semi-stochastic kernel in (x,A) and $Q(.,.,t) \leq 1$;

3. $Q(.,.,t) : X \times \chi \to [0,\infty)$ is a non-decreasing function, continuous from the right, and $Q(x,A,0) = 0$;

4. $Q(X,A,\infty+) := P(x,A) : X \times \chi \to [0,\infty)$ is a stochastic kernel;

5. Given $x \in X$ fixed, the function $Q(x,X,t)$ is a distribution function with respect to $t \geq 0$;

Definition 3.5. We call a process $(x_n, \theta_n)_{n\in\mathbb{Z}^+}$ on the phase space $X \times R^+$ a Markov renewal process (MRP) if the semi-Markov kernel below gives the transitional probabilities:

$$Q(x,A,t) = P\{x_{n+1} \in A, \theta_{n+1} \leq t | x_n = x\}, \forall x \in X, A \in \chi, t \in R^+ \tag{3.1.6}$$

It is clear that we have

$$\begin{aligned} Q(x,A,t) &= P(x,A)G_x(t) \\ P(x,A) := Q(x,A,+\infty) &= P\{x_{n+1} \in A | x_n = x\} \\ G_x(t) &:= P\{\theta_{n+1} < t | x_n = x\} \end{aligned} \tag{3.1.7}$$

where in the above $x \in X, A \in \chi, t \in R^+$.

The following process is called a semi-Markov process, where $\nu(t)$ is a counting process:

$$x(t) := x_{\nu(t)} \tag{3.1.8}$$

3.1.3 Stationary Distributions of Markov Chains [21]

Definition 3.6. (Stationary Probability Measures)

For the Markov chain $x_n, n \geq 0$, ρ is called a stationary (or, sometimes, invariant) probability measure on (E,ε), if for any $A \in \varepsilon$

$$\rho(A) = \int_E \rho(dx) P(x,A) \tag{3.1.9}$$

An initial probability measure μ on χ is called initial probability if it satisfies

$$P\{\omega : x(\omega_0) \in A\} = P(x_0 \in A) := \mu(A), A \in \chi \tag{3.1.10}$$

All the probabilities related to the Markov process $(x_n)_{n\in\mathbb{Z}^+}$ may be defined using μ and P as follows:

$$\begin{aligned} P(x_0 \in A) &= \mu(A) \\ P(x_1 \in A | x_0 = x) &= P(x, A) \\ P(x_1 \in A) &= \int_X P(x, A) d\mu(x) \end{aligned} \tag{3.1.11}$$

Indeed, a Markov chain is completely described by its one-step transition probability matrix, as well as the additional information about the specification of a probability distribution on the state of the process at time 0. Analysis of a Markov chain concerns itself mostly with the calculation of the probabilities of the possible paths of the process. The n-step transition probability matrices, $P^{(n)} = ||P_{ij}^n||$ where P_{ij}^n denotes the probability of the process going from state i to state j in n transitions are fundamental to these calculations.

Theorem 3.7. *(Stationary Distribution with Matrices)*

If we have a Markov chain with N states (N is finite), then the 1xN vector π of stationary probabilities can be found by solving the following equation, where we use P, the one-step transition probability matrix:

$$\pi \cdot P = \pi \tag{3.1.12}$$

In the above, $\cdot$ denotes standard matrix multiplication.

Theorem 3.8. *(Markov Matrix Multiplication)*

If $r + s = n$; $r, s \geq 0$ are integers, and P is the one-step transition probability matrix, we have that

$$P_{ij}^n = \sum_{k=0}^{\infty} P_{ik}^r P_{kj}^s \tag{3.1.13}$$

$$P_{ij}^n = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{3.1.14}$$

It is clear that Equation 3.1.13 denotes simply the i, j^{th} entry of P^n, the n^{th} power of P; this is standard matrix entry indexing.

3.1.4 Concepts for Ergodic Markov Chains [11]

Definition 3.9. (Ergodic Markov Chains.)

If $x_n, n \geq 0$, is an ergodic Markov chain, then we have the following three properties:

1. Given any probability measure α on (E, ξ), the following holds:

$$||\alpha P^n - \rho|| \to 0, n \to \infty \tag{3.1.15}$$

2. Given any $\varphi \in B$, the following holds for any probability measure μ on (E, ξ):

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} \varphi(x_k) = \int_E \rho(dx)\varphi(x) \tag{3.1.16}$$

For a homogenous ergodic Markov chain $(x_n)_{n\in\mathbb{Z}^+}$ where the stationary probability vector π, we have:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} p_{ij}^{(k)} = \pi_j \tag{3.1.17}$$

Denoting with P the operator of transition probabilities on B defined by

$$P\varphi(x) = \mathbb{E}[\varphi(x_{n+1})|x_n = x] = \int_E P(x, dy)\varphi(y) \tag{3.1.18}$$

and denoting by P^n the n-step transition operator corresponding to $P^n(x, A)$, the essential Markov property of the Markov chain is stated below:

$$\mathbb{E}[\varphi(x_{n+1})|F_n] = P\varphi(x_n) \tag{3.1.19}$$

Definition 3.10. (Stationary Projector)

Denoting with Π the stationary projector from the stationary distribution $\rho(B), B \in \varepsilon$ of a Markov chain (x_n) we have that:

$$\Pi\varphi(x) := \int_E \rho(dy)\varphi(y)\mathbf{1}(x) = \hat{\varphi}\mathbf{1}(x), \quad \hat{\varphi} := \int_E \rho(dx)\varphi(x) \tag{3.1.20}$$

Furthermore, as a potential of the Markov chain, the following series converges:

$$R_0 := \sum_{n=0}^{\infty} [P^n - \Pi] \tag{3.1.21}$$

Finally, we have the following, which is an important property of ergodic Markov chains:

Definition 3.11. (Continuous-time ergodic characterization)

In the case of continuous-time, the Markov process $x(t), t \geq 0$ is called ergodic with stationary probability π, if for every $\varphi \in B$, and for any probability measure π on (E, ξ), the following holds:

$$\lim_{t\to\infty} \frac{1}{t} \int_0^t \varphi(x(s))ds = \int_E \pi(dx)\varphi(x) \tag{3.1.22}$$

3.2 Geometric Brownian Motion [2]

A stochastic process S_t is said to follow a Geometric Brownian Motion (GBM) if it satisfies the following stochastic differential equation (SDE), where W_t is a Wiener process (defined below):

$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t \tag{3.2.1}$$

Definition 3.12. (Wiener process)

A Wiener process is a stochastic process is defined as follows:

- $W_0 = 0$ with probability 1
- The function $t \to W_t$ is continuous almost surely

- W_t has independent increments; in particular, $W_t - W_s \sim N(0, t-s)$ (for $0 \leq s < t$), where $N(\mu, \sigma^2)$ denotes the normal distribution with mean μ and variance σ^2.

This means that for $0 \leq s_1 < t_1 \leq s_2 < t_2$, $W_{t_1} - W_{s_1}$ and $W_{t_2} - W_{s_2}$ are independent random variables. This extends to n increments.

Alternatively, we have the following Levy characterization for a Wiener process: A Wiener process is an almost surely continuous martingale with $W_0 = 0$ with quadratic variation $[W_t, W_t] = t$. This means that $W_t^2 - t$ is a martingale).

Definition 3.13. (Levy process)

Assume $(\Omega, F, \mathbf{F}, P)$ is a filtered probability space, $F = F_T$, and the filtration $\mathbf{F} = (F_t)_{t \in [0,T]}$ satisfies the usual conditions. $T \in [0, \infty)$ denotes the time horizon (this can be infinite). A cadlag (continuous from the right, limit exists on the left), F_t adapted, real valued stochastic process $L = (L_t)_{0 \leq t \leq T}$ with $L_0 = 0$ a.s. is called a Levy process if we have the following properties:

- L has independent increments: $L_t - L_s$ is independent of F_s for any $0 \leq s < t \leq T$.
- L has stationary increments: $0 \leq s, t \leq T$, the distribution of $L_{t+s} - L_t$ is t-independent.
- L is stochastically continuous: For all $0 \leq t \leq T$ and $\varepsilon > 0$, $\lim_{s \to t} P(|L_t - L_s| > \varepsilon) = 0$.

It is important to note that Brownian motion is the only non-deterministic Levy process that has continuous sample paths. Also, the sum of a linear drift, a Brownian motion, and a compound Poisson process is again a Levy process. This is called a jump-diffusion process.

The paths of $L = (L_t)_{0 \leq t \leq T}$, a Levy jump-diffusion (a Brownian motion plus a compensated compound Poisson process), can be described by:

$$L_t = bt + \sigma W_t + \left(\sum_{k=1}^{N_t} J_k - t\lambda\kappa \right) \tag{3.2.2}$$

Here $b \in \mathbb{R}$, $\sigma \in \mathbb{R}^+$, $W = (W_t)_{0 \leq t \leq T}$ is a Wiener process, $N = (N_t)_{0 \leq t \leq T}$ is a Poisson process with intensity parameter λ, and $J = (J_k)_{k \geq 1}$ is an i.i.d. sequence of random variables with probability distribution F and $\mathbb{E}[J] = \kappa < \infty$. Thus, F describes the distribution of the jumps. The characterization function of L_t is given by the following:

$$\mathbb{E}[e^{iuL_t}] = \exp[t(iub - \frac{\mu^2\sigma^2}{2} - \int_{\mathbb{R}}(e^{iux} - 1 - iux)\lambda F(dx))] \tag{3.2.3}$$

3.3 Approximation Techniques [11]

3.3.1 Ergodic Approximation

The results on ergodic Markov chains from section 3.1.4 will prove to be very useful in the manipulation of GMRP methods where the underlying Markov chain is ergodic. As we will see in 8.5 and 8.6, it will allow us to approximate a GMRP - modulated asset price with a Geometric Brownian Motion (GBM). There is an easy way to determine the ergodicity of a Markov chain by looking at its one-step transition matrix.

Definition 3.14. (Communicating states)

Two states x_1 and x_2 of a Markov chain are said to communicate if there exists a non-zero probability of starting in x_1 at time t and reaching x_2 at some time $s \geq t$ and vice versa from x_2 to x_1.

Theorem 3.15. *(Ergodicity and Communicating States)*

A Markov chain is ergodic if and only if all states communicate.

It is easy to check by examination of the rows of the Markov transition matrix whether all states communicate. Thus is is straightforward to check if a (relatively small-sized!) Markov transition matrix is that of an ergodic Markov chain.

3.3.2 Merged Markov Processes

A merged Markov process can be thought of as a Markov chain embedded within a Markov chain; it is used to approximate the behavior of the larger state-space Markov chain. For each embedded state, we can apply simplifying assumptions due to each embedded state's ergodicity and average over the embedded states. We present the formal definition below:

Definition 3.16. (Merged Markov Process)

If the Markov state space $X = \{1,2,...,N\}$ consists of r ergodic classes $X_i, i = 1,2,...,r$, with stationary distribution $p_i(dx)$ in each class, then the Markov chain x_k is merged to the Markov chain $\hat{x}_k$ in the merged phase space $\hat{X} = \{1,2,...,r\}$. $\hat{x}(t)$ is a merged Markov process in phase states of space $\hat{X}$.

Definition 3.17. (Double Averaged Markov Process)

If we have, as in the above case, r ergodic classes, and these form a Markov chain, we can then examine the behavior of this new Markov chain. By finding the stationary probabilities of this ergodic chain, we can use these probailities to approximate the behavior of this chain. This is called double averaging of a Markov process, and it is a powerful tool for simplifying the behavior of a Markov chain with multiple ergodic states.

3.4 No-Arbitrage Principle [4]

Definition 3.18. (No-Arbitrage Principle)

There is no such portfolio (starting set of positions, i.e. holdings in financial assets) such that the initial value of the portfolio is equal to zero, but at some time $T > 0$ the value is positive, with nonzero probability.

For an example of arbitrage, consider the following situation: The market consists of a bond (or bank) with fixed interest rate $r \geq 0$ and an one asset (a stock). Suppose the interest rate on loans is $r = 10\%$. The price of the stock at time t is denoted $S(t)$. We have $S(0) = \$100$ and

$$S(1) = \begin{cases} \$120 & \text{with probability } 0.5 \\ \$90 & \text{with probability } 0.5 \end{cases}$$

Suppose a contract which allows the owner to buy the stock at time $t = 1$ for \$100 is sold at time 0 for \$100. Then an investor with no money at could borrow \$100 to purchase the contract. At time 1, in the case $S(1) = \$120$, he could exercise the contract and buy the stock for \$100 and sell

it for \$120 immediately, then pay $\$100 \times 10\% = \$110$ back to the bank. Thus this investor has a profit of $\$120 - \$100 = \$10$, and this scenario has a probability of $0.5 \neq 0$. Thus this violates the No-Arbitrage Principle.

Chapter 4

European Option Pricing with a GBM Model

4.1 Definition of European Options [4]

A financial option on an underlying asset $S(t)$ is called a European Call Option with strike price E and expiry time T if it has a payoff function $\max(S(T)-E,0)$.
A financial option on an underlying asset $S(t)$ is called a European Put Option with strike price E and expiry time T if it has a payoff function $\max(E-S(T),0)$.

4.2 Black-Scholes option pricing formulas for European-style options under GBM[4]

Assume the underlying asset price follows a Geometric Brownian Motion (GBM), where $W(t)$ is a Wiener process, and m is the drift parameter. Here $S(t)$ has a log-normal distribution.

$$S(t) = S(0)e^{mt+\sigma W(t)} \tag{4.2.1}$$

For a generalized European option on the asset $S(t)$ with payoff function $f(S(T))$, we have the following time-zero value, where E^* denotes expectation under a risk-neutral probability measure

P_*, with interest rate r:

$$D(0) = E^*(e^{-rT} f(S(T))) \tag{4.2.2}$$

Stochastic calculus gives us the martingale property:

$$E^*(e^{-rt} S(t)) = S(0) \tag{4.2.3}$$

Write the time-zero European call option value as $C^E(0)$ (where the strike price is X and expiry is T). Then we have:

$$C^E(0) = E^*(e^{-rT} \max(S(T) - X, 0)) \tag{4.2.4}$$

Noting that $V(t) = W(t) + (m - r + \frac{1}{2}\sigma^2)\frac{t}{\sigma}$ is a Wiener process under the risk-neutral probability measure P^* (with $t \geq 0$), the above becomes

$$C^E(0) = E^*(e^{-rT} \max(S(T) - X, 0)) = E^*(\max(S(0)e^{\sigma V(t) - \frac{1}{2}\sigma^2 T} - Xe^{-rT}, 0))$$

$$= \int_{d_2\sqrt{T}}^{\infty} (S(0)e^{\sigma x - \frac{1}{2}\sigma^2 T} - Xe^{-rT}) \frac{1}{\sqrt{2\pi T}} e^{-x^2/(2T)} dx$$

$$= S(0) \int_{-d_1}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy - Xe^{-rT} \int_{-d_2}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy$$

$$= S(0)N(d_1) - Xe^{-rT} N(d_2)$$

So we have

$$C^E(0) = S(0)N(d_1) - Xe^{-rT} N(d_2) \tag{4.2.5}$$

where we have used the notation

$$d_1 = \frac{\ln \frac{S(0)}{X} + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \quad d_2 = \frac{\ln \frac{S(0)}{X} + (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}} \tag{4.2.6}$$

and

$$N(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy = \int_{-x}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy \tag{4.2.7}$$

Analogously, we may derive the formula for European put options:

$$P^E(t) = Xe^{-r(T-t)}N(-d_2) - S(t)N(-d_1) \tag{4.2.8}$$

where d_1 and d_2 are as given in Equation (4.2.6).

Chapter 5

The Binomial Model and Pricing of American Options

5.1 Definition of American Options [4]

The main apparent difference between American options from European options (see 4.1) is that American options can be exercised at any time the option holder between time $t = 0$ and the time of expiry $t = T$.

A financial option on an underlying asset $S(t)$ is called a American Call Option with strike price E and expiry time T if, for any time t such that $0 < t < T$ it has a payoff function $\max(S(t) - E, 0)$.

A financial option on an underlying asset $S(t)$ is called a American Put Option with strike price E and expiry time T if, for any time t such that $0 < t < T$ it has a payoff function $\max(E - S(t), 0)$.

5.2 Binomial Model of an Asset Price (Cox-Ross-Rubinstein) [4, 5]

5.2.1 One-Step Binomial Model

Assume the asset price $S(1)$ at time 1 may take two values, with p and $1 - p$ representing the probabilities, $S(0)$ is the asset price at time 0, and $-1 < d < u < 1$:

$$\begin{cases} S^u = S(0)(1+u) & \text{with probability } p \\ S^d = S(0)(1+d) & \text{with probability } (1-p) \end{cases} \tag{5.2.1}$$

Using a replicating portfolio strategy, we must solve the system of equations, where $f(S(N))$ is the payoff function of the option:

$$\begin{cases} x(1)S^u + y(1)(1+r) = f(S^u) \\ x(1)S^d + y(1)(1+r) = g(S^d) \end{cases} \tag{5.2.2}$$

We get $x(1)$ and $y(1)$ as follows, and use these numbers to find the value of the replicating portfolio at time 0, thus finding the value of the option.

$$x(1) = \frac{f(S^u) - f(S^d)}{S^u - S^d} \; y(1) = (-1)\frac{(1+d)f(S^u) - (1+u)f(S^d)}{(u-d)(1+r)} \tag{5.2.3}$$

We have the result that the risk-neutral probability in this model is given by

$$p^* = \frac{r-d}{u-d} \tag{5.2.4}$$

It turns out that the discounted payoff computed with respect to the risk-neutral probability is equal to the present value of the contingent claim:

$$D(0) = E^*((1+r)^{-1} f(S(1)) \tag{5.2.5}$$

5.2.2 One-Step Model Extended to N Steps

Clearly, we can extend the one-step model (Equation 5.2.1) naturally to N steps, as follows: If the starting stock price is S_0, the the stock price at time n, S_n, is represented as

$$S_n = S_0 \prod_{i=1}^{n} (1+\rho_i) \tag{5.2.6}$$

where

$$\begin{cases} \rho_i = u & \text{with probability } p \\ \rho_i = d & \text{with probability } (1-p) \end{cases} \tag{5.2.7}$$

The possible paths of the asset create what is referred to as a binomial lattice; a recombinant binomial tree.

5.2.3 Two-Step Binomial Model

In the two-step model, extending Equation (5.2.1), we have the two values for $S(1)$ as $S^u = S(0)(1+u)$ and $S^d = S(0)(1+d)$, and the following possibilities for $S(2)$:

$$S^{uu} = S(0)(1+u)^2 \quad S^{ud} = S(0)(1+u)(1+d) \quad S^{dd} = S(0)(1+d)^2$$

Following an analogous strategy to the one-step method, and using p^* as in Equation (5.2.4), since the risk-neutral probability is the same as in the one-step case, we find that:

$$D(0) = \frac{1}{(1+r)^2}\left[(p^*)^2 f(S^{uu}) + 2p^*(1-p^*)f(S^{ud}) + (1-p^*)^2 f(S^{dd})\right] \tag{5.2.8}$$

Extending to 3 steps we have

$$D(0) = \frac{1}{(1+r)^3}\left[(p^*)^3 f(S^{uuu}) + 3(p^*)^2(1-p^*)f(S^{uud}) + 3(p^*)(1-p^*)^2 f(S^{udd}) + (1-p^*)^3 f(S^{ddd})\right] \tag{5.2.9}$$

5.2.4 General N-step Binomial Model

In the N-step generalization of the previous pricing schemes, we have, with $-1 < d < u$:

$$S_n = S_0 \prod_{i=1}^{n}(1+\rho_i) \quad \text{where} \quad \begin{cases} \rho_i = u & \text{with probability } p \\ \rho_i = d & \text{with probability } (1-p) \end{cases} \tag{5.2.10}$$

Extrapolating from previous results and using induction, we have the following option pricing formula for a payoff $f(S(N)$ (this is a risk-neutral valuation under the measure p^*):

$$D(0) = E^*\left((1+r)^{-N} f(S(N))\right) = \frac{1}{(1+r)^N} \sum_{k=0}^{N} \binom{N}{k} (p^*)^k (1-p^*)^{N-k} f\left(S(0)(1+u)^k (1+d)^{N-k}\right) \tag{5.2.11}$$

5.2.5 Cox-Ross-Rubinstein Formulas for the European Options

Using Equation (5.2.11), the prices of a European call $C^E(0)$ and a European put $C^P(0)$ with strike price X to be exercised after N steps are given as follows in the well-known Cox-Ross-Rubinstein equations, with p^* the risk-neutral probability as in Equation (5.2.4), and $q = p^* \cdot \left(\frac{1+u}{1+d}\right)$

$$\begin{aligned} C^E(0) &= S(0)[1-\Phi(m-1,N,q)] - (1+r)^{-N} X[1-\Phi(m-1,N,p^*)] \\ P^E(0) &= -S(0)\Phi(m-1,N,q) + (1+r)^{-N} X\Phi(m-1,N,p^*) \end{aligned} \tag{5.2.12}$$

where we denote

$$\Phi(m,N,p) = \sum_{k=0}^{m} \binom{N}{k} p^k (1-p)^{N-k}$$

5.3 Definition of American Options [4]

The main apparent difference between American options from European options (see 4.1) is that American options can be exercised at any time the option holder between time $t = 0$ and the time of expiry $t = T$.

A financial option on an underlying asset $S(t)$ is called a American Call Option with strike price E and expiry time T if, for any time t such that $0 < t < T$ it has a payoff function $\max(S(t) - E, 0)$.

A financial option on an underlying asset $S(t)$ is called a American Put Option with strike price E and expiry time T if, for any time t such that $0 < t < T$ it has a payoff function $\max(E - S(t), 0)$.

5.4 Pricing of American Puts via the Binomial Model Backwardation Method [4, 5]

By a famous result of Stephen Ross, we have that the price of an American call must be the same as that of a European call. We now concentrate on determining the value of an American put in the Binomial formulation.

5.4.1 American Options Under the Binomial Method

Consider an American option expiring at time 2. Unless the option has already been exercised, it will be worth $D^A(2) = f(S(2))$ at expiry. At time 1, the option holder has the choice to exercise immediately with payoff $f(S(1))$ or wait until time 2. The value of waiting to exercise the American option is exactly the same problem as a one-step European option to be priced at time 1 (see Equation 5.2.1), and the value of the American option should be the maximum of this value and of exercising the option immediately. Hence the value is

$$D^A(1) = \max\left\{f(S(1)), \frac{1}{1+r}\left[p^* f(S(1)(1+u)) + (1-p^*) f(S(1)(1+d))\right]\right\} = f_1(S(1)) \quad (5.4.1)$$

$f_1(S(1))$ is a random variable with two values: $f_1(x) = \max\left\{f(x), \frac{1}{1+r}[p^* f(x(1+u)) + (1-p^*) f(x(1+d))]\right\}$. So the time 0 value of the American option is

$$D^A(0) = \max\left\{f(S(0)), \frac{1}{1+r}[p^* f_1(S(0)(1+u)) + (1-p^*) f_1(S(0)(1+d))]\right\} \quad (5.4.2)$$

Induction to the N- step binomial case gives us:

$$\begin{aligned} D^A(N) &= f(S(N)) \\ D^A(N-1) &= \max\left\{ f(S(N-1)), \tfrac{1}{1+r}[p^* f(S(N-1)(1+u)) + (1-p^*) f(S(N-1)(1+d))] \right\} \\ &=: f_{N-1}(S(N-1)) \\ &\vdots \\ D^A(0) &= \max\left\{ f(S(0)), \tfrac{1}{1+r}[p^* f_1(S(0)(1+u)) + (1-p^*) f_1(S(0)(1+d))] \right\} \end{aligned} \tag{5.4.3}$$

This pricing method is referred to as "backwardation" as the time-zero price is found by working backwards through the binomial tree.

5.4.2 American Option Valuation Under the Binomial Scheme With GBM Model

Given a Geometric Brownian Motion asset price model as in Equation (3.2.1), we have the following possibilities for the logarithm of the asset return, using $p = 1/2$, in the analogous binomial model as in Equation (5.2.6), where $\tau = \frac{1}{N}$, where N is the number of steps:

$$\begin{cases} \ln(1+u) = m\tau + \sigma\sqrt{\tau} & \text{with probability } 1/2 \\ \ln(1+d) = m\tau + \sigma\sqrt{\tau} & \text{with probability } 1/2 \end{cases} \tag{5.4.4}$$

Equations (5.4.4) can be re-written:

$$\begin{cases} u = e^{m\tau + \sigma\sqrt{\tau}} - 1 \\ d = e^{m\tau - \sigma\sqrt{\tau}} - 1 \end{cases} \tag{5.4.5}$$

To verify the validity of Equation (5.4.4), let us introduce a sequence of i.i.d.r.v $\xi(n)$:

$$\xi(n) = \begin{cases} +\sqrt{\tau} & w.p.\ \frac{1}{2} \\ -\sqrt{\tau} & w.p.\ \frac{1}{2} \end{cases} \tag{5.4.6}$$

The logarithmic return can be written as $m\tau + \sigma\xi(n)$. It can be shown that the sum of these variables $w(n) = \xi(1) + \xi(2) + ... + \xi(n)$ with $w(0) = 0$ behaves as a Wiener process $W(t)$. Then the model gives us, with $t = \tau n$,

$$S(t) = S(0)e^{mt+\sigma W(t)} \tag{5.4.7}$$

For the pricing of options, we use the risk-neutrality assumption $m = r$ at this point. This guarantees that the discounted price process will be a martingale under the risk-neutral probability measure. To summarize, given a GBM model of an asset price for which we want to price an American put, we can use the relations in (5.4.5) to find the values for u and d to approximate it with a binomial model, then use the "backwardation" method from part (5.4.1) on this binomial model to price an American put on this asset.

Chapter 6

Aase Model

6.1 The Aase Continuous-Time Trading Model [1]

In the Aase model, we have two securities: a bond $S^0(t)$ and an asset $S(t)$. Furthermore we have the assumptions of no transaction costs and that continuous trading can take place up to some fixed horizon T. The bond exhibits the following behavior:

$$S^0(t) = \exp\left(\int_0^t r(s)ds\right),\ 0 \leq t \leq T \tag{6.1.1}$$

The discounted stock price is given by (denoting $\beta_t = 1/S^0(t)$)

$$S^*(t) = \beta(t)S(t) \tag{6.1.2}$$

(Ω, F, P) is defined as a probability space for $S^0(t)$ and $S(t)$ (here $\mathbb{F} = \{F_t : 0 \leq t \leq T\}$ is a standard filtration).

Denoting by $\mathbb{P}$ the set of probability measures P^* on (Ω, F) equivalent to P such that S^* is a martingale under P^*, we have the result that if X is a contingent claim, and $V_t^* = E^*(\beta_T X|F_t)$ where E^* denotes the risk-neutral expectation under P^*, then X is P-attainable if and only if V^* is in $L^2(P^*)$. In this case $V^*(\phi) = V^*$ for any admissible self-financing strategy ϕ which generates X.

We now outline the specifics of our model for S_t. We pick a positive semi-martingale equipped

with specific structures. The two components of out model are a sample continuous S^c, and a pure jump part S^d. The latter behaves well, and contains a finite number of jumps on a finite time interval. It is described as a random measure, while S^c is usually an Ito process. We'll examine Markov models, in which case S^c becomes a diffusion process, and S^d, a Markov process of the marked point process type, for applications.

Assume the process S_t is right-continuous, has a limit on the left, and is described as follows, where the coefficients $\mu(t,\omega)$, $\sigma(t,\omega)$ and $\gamma(t,\omega)$ are assumed to be predictable processes:

$$\frac{dS(t)}{S(t-)} = \mu(t,\omega)dt + \sigma(t,\omega)dB_t + \int_R \gamma(t;y)\nu(dy;dt) \tag{6.1.3}$$

In (Equation 6.1.3) B_t is a (P,F_t) standard Brownian motion (which is independent of the random measure term). The last term is the source of jumps of random relative size at the time points τ_n. $\nu(dy;dt)$ is a random point measure. It can be interpreted as follows: $\nu(A;t)$ is the number of jumps from the embedded marked point process N_t (with values in the Borel set $A \subseteq R$ before time t).

6.2 Pure Jump Model in the Aase Formulation [1]

6.2.1 Description of the Pure Jump Model

We present below an Aase formulation-compatible, which is a special case of processes satisfying (6.1.3). We also present European option pricing formulas which result. Suppose we can represent the contingent claim X as $X = \psi(S_T)$, i.e. $\psi(S_T) = (c - S_T)^+ := \max(c - S_T, 0)$; or $\psi(S_T) = (S_T - c)^+ := \max(S_T - c, 0)$ (European put option or call option respectively). Consider the following geometric compound Poisson process model (where $N(t)$ are the Poisson-distributed jump times, and the Y_n are i.i.d.r.v independent of $N(t)$).

$$S_t = S_0 \prod_{n=1}^{N(t)} (1 + Y_n) \tag{6.2.1}$$

It should be noted at this point that the geometric compound Poisson process $\{S_t^*\}_{t \in R_+}$ presented in (6.2.1) is used as a trading model in many financial applications as a pure jump model. Uses

of pure jump models include the realistic modeling of trends in markets where trend following is viewed as a go-to strategy, or in the eyes of some, the only reliable strategy. Examples of such markets include the futures market, where mean reversion drives traders to consider trend following strategies.

Writing $L_t = \prod_{n=1}^{N(t)} h(Y_n)$, and $H^* = hH$ is the distribution of Y_n under the risk-neutral measure P^*. If r is constant, $S_t^* = e^{-rt}S_t$ is aP^*-martingale whenever $m^*\lambda = r$, writing $m^* = \int yH^*(dy)$. The purpose of $h(y)$ is to shift the mean of Y from m to $m^* = r/\lambda$. The time-zero price π_τ of X equals

$$\pi_\tau = e^{-rT}E^*\{\psi(S_T)|F_0\} = e^{-rT}E^*\{E^*\{\psi(S_T)|N(T),F_0\}\}$$

$$= e^{-rT}\sum_{k=0}^{\infty}\frac{e^{-rT}(\lambda T)^k}{k!}\int_{-1}^{\infty}...\int_{-1}^{\infty}\psi(S_0\prod_{n=1}^{k}(1+y_n))H^*(dy_1)\times...\times H^*(dy_k) \tag{6.2.2}$$

where we have

$$\int_{-1}^{\infty} yH^*(dy) = \int_{-1}^{\infty} y\,h(y)H(dy) = \frac{r}{\lambda} \tag{6.2.3}$$

6.2.2 Simulation of a Case of the Aase Pure Jump Model

6.2.2.1 Guide to the MATLAB Code AaseSimulation.m

In this section we present a MATLAB code which simulates the Aase pure jump model introduced in (6.2.1).

We will assume that Y_n are all i.i.d. random variables from the uniform distribution on $[-C,C]$ where $0 < C < 1$.

First we need the following theorem to help us in designing the code as efficiently as possible:

Theorem 6.1. *(Waiting Times Between Poisson-Distributed Jumps)*

Suppose that the number of events (jumps, etc.) between two times s and t, where $0 \le s < t < \infty$ is modeled by a Poisson process with parameter λ_P. Then, the waiting times are exponentially distributed with p.d.f. $\lambda e^{-\lambda x}$, with the parameter $\lambda = \lambda_P$.

Thus we can model the inter-jump times for (6.2.1) with exponentially distributed random variables.

The MATLAB/Octave code AaseSimulation.m takes as user input the intensity of jump parameter λ, the max time to model the jump process, T, the and a control mechanism for the maximum number of jumps in the interval (to prevent the code from running too long if so desired).

6.2.2.2 Sample Run of the MATLAB Code AaseSimulation.m

We run the code AaseSimulation.m in Octave (a open-source MATLAB clone) using the input value of $\lambda = 20$, starting stock price = 2, max time = 100, C = 0.05, and we input max number of jumps =1e5.

Below is what we get on the Octave Terminal:

```
This code simulates an Aase-compatible pure jump process with
Yk i.i.d. distributed as uniform on Scale*[-1,1]
Input lambda:   20

Enter the starting stock price, S_0 :   2

What is the maximum time T you want to extend the process to?
Please enter T:   100

Enter the scale factor C for the jump sizes; i.e. from uniform distribution [-C,C];
Generally 0<C<1  :  .05

Enter the maximum allowed number of jumps in the time
interval; recommended value is 2*1e4:   1e5
```

Running the code produces a picture, which can be seen in Figure 6.2.1.

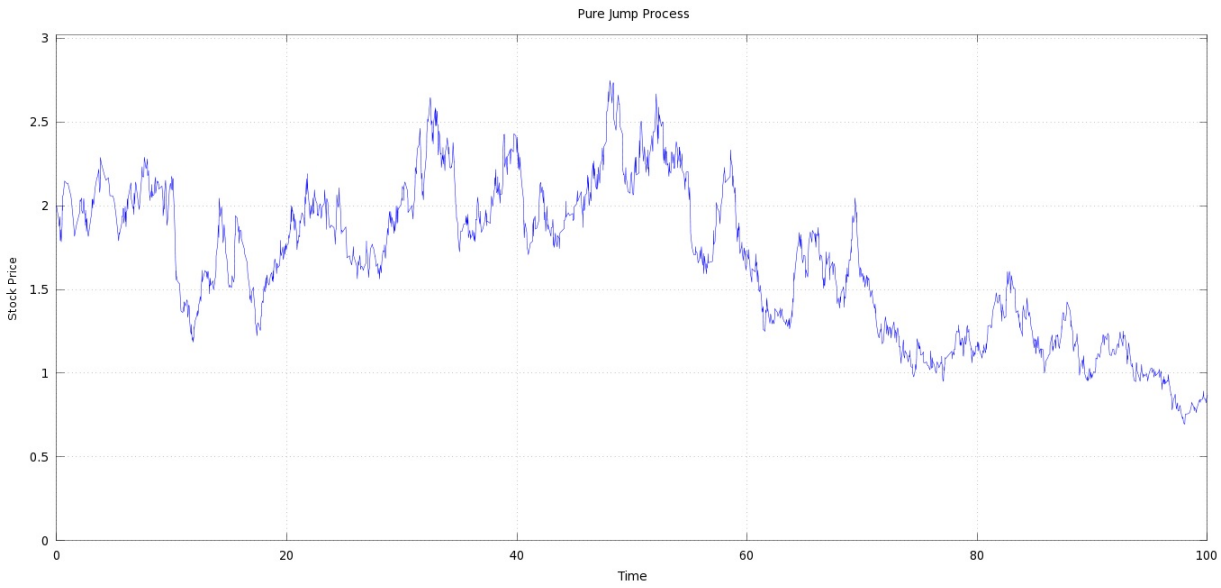

Figure 6.2.1: Resulting Pure Jump path of Aase Model with high λ

To see the influence of the Poisson process on the model, we set a larger value for C and run the code on a shorter time frame:

We run the code AaseSimulation.m in Octave (a open-source MATLAB clone) using the input value of $\lambda = 20$, starting stock price = 2, max time = 2, C = 0.1, and we input max number of jumps =1e5.

Below is what we get on the Octave Terminal:

```
 This code simulates an Aase-compatible pure jump process with
Yk i.i.d. distributed as uniform on Scale*[-1,1]
Input lambda:   20

Enter the starting stock price, S_0 :   2

What is the maximum time T you want to extend the process to?
Please enter T:   2

Enter the scale factor C for the jump sizes; i.e. from uniform distribution [-C,C];
```

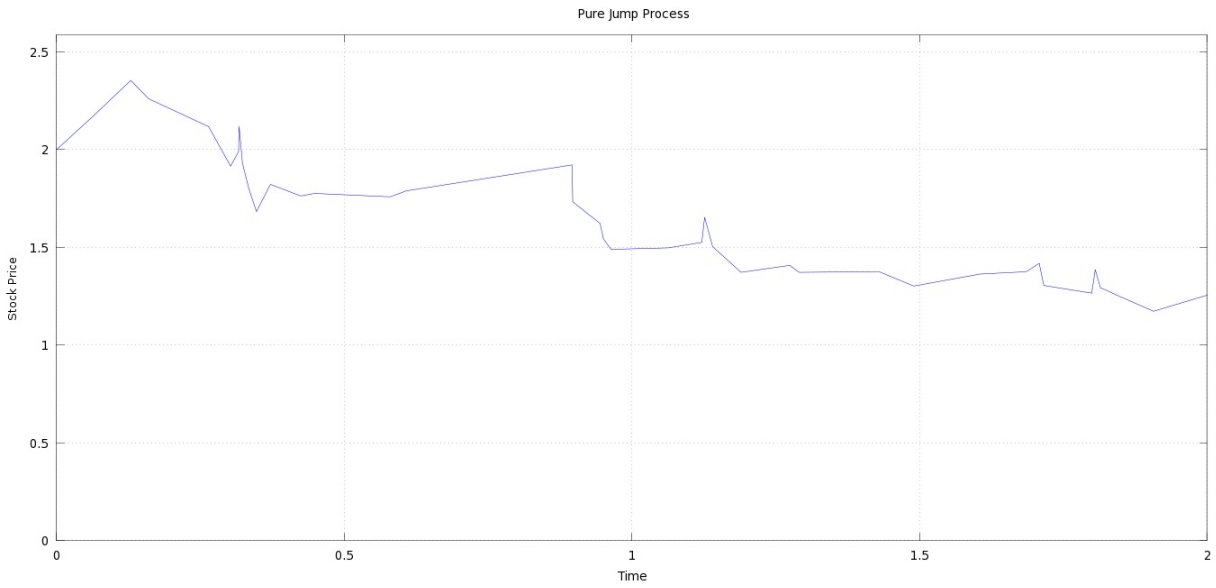

Figure 6.2.2: Resulting Pure Jump path of Aase Model with high λ

```
Generally 0<C<1  :  .1

Enter the maximum allowed number of jumps in the time
interval; recommended value is 2*1e4:   1e5
```

Running the code produces a picture, which exhibits the choppiness and effect of random jump times. This picture can be seen in Figure 6.2.2.

6.2.2.3 MATLAB Code AaseSimulation.m

The code can be found in Appendix B.1.

Chapter 7

Geometric Markov Renewal Process (GMRP)

7.1 Standard GMRP

7.1.1 Theoretical Background [24]

If $(x_k)_{k\in Z_+}$ is a Markov chain in phase space (X,χ) with transition probabilities $P(x,A)$ where $x\in X$, $A\in\chi$, $(\tau_k)_{k\in Z_+}$ is a sequence of i.i.d. random variables such that $P(\tau_{n+1}-\tau_k<t|x_n=x)=G_x(t)$ where $x\in X$, $t\in R_+$, and if we let $\theta_{n+1}:=\tau_{n+1}-\tau_n$, and denote

$$\tau_n=\sum_{k=1}^{n}\theta_k$$

and $\nu(t):=\max\{n:\tau_n\leq t\}$ be a counting process:

Then $(x_n;\theta_n)_{n\in Z_+}$on phase states of $X\times R_+$ is called a Markov renewal process (MRP). For $\rho(x)$ such that $\rho(x)>-1$, a bounded continuous function on X , the following functional on Markov renewal process $(x_n,\theta_n)_{n\in Z_+}$, where S_0 is a fixed positive starting price, is called the Geometric Markov Renewal Process (GMRP):

$$S_t=S_0\prod_{k=1}^{\nu(t)}(1+\rho(x_k)) \tag{7.1.1}$$

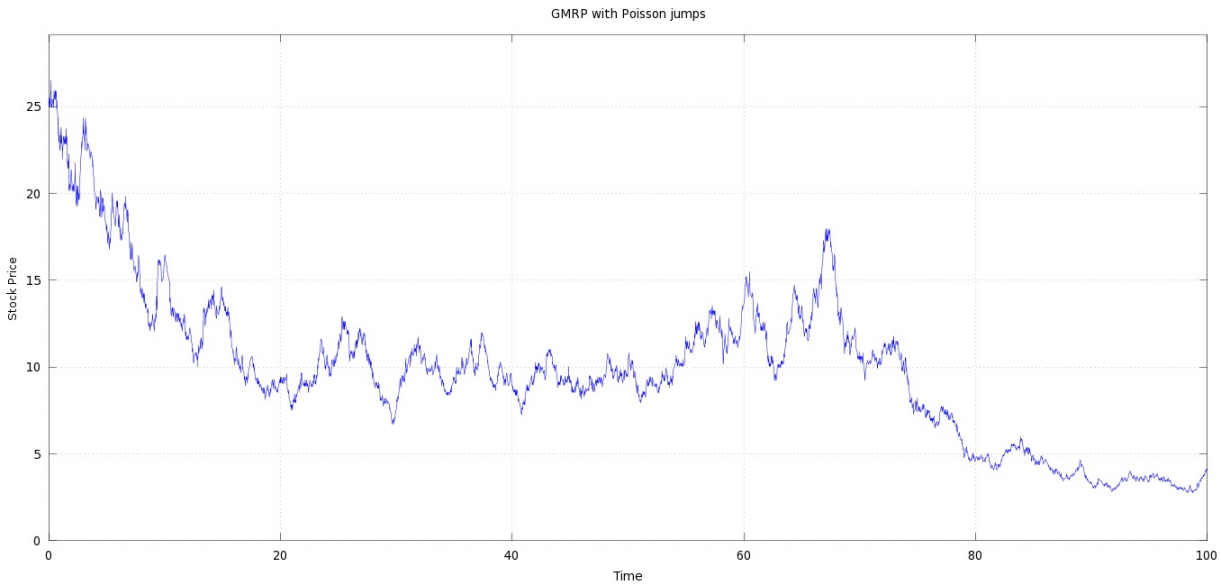

Figure 7.1.1: Sample GMRP path with Poisson jumps

In many cases, the counting process $\nu(t)$ is Poisson distributed. We present a MATLAB code which can simulate the GMRP with a Poisson jump model.

7.1.2 Guide to the MATLAB Code GMRP.m

The MATLAB code GMRP.m takes as input the number of states in the Markov chain, the transition matrix entries, the rho-values, the values for λ for each state, the starting stock price, the time till expiry of the simulation, and the starting state. It outputs a graph of a simulated GMRP asset path with Poisson jumps. For an example, we run the code GMRP.m with the following input parameters:

Transition matrix:

$$\begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

$\rho = 0.02$ for state 1, $\rho = -0.02$ for state 2;$\lambda = 40$ for state 1; $\lambda = 40$ for state 2; starting stock price = 25, and time to simulate price path for = 100. Starting state = 1. After running the code, we get the picture which can be seen in Figure (7.1.1). The MATLAB code can be found in Appendix B.2.

7.2 Fixed Time Increments GMRP[24]

7.2.1 Definition of Fixed Time Increments GMRP [24]

Now we introduce Fixed Time Increments GMRP, sometimes called (rather vaguely) discrete-time GMRP or discretized GMRP. In this case, we allow the counting process $N(t)$ to be deterministic: in particular, we let $N(t)$ be the number of subdivisions of the time interval $[0,t]$. This is a practical simplification and is commonly applied in various regime-switching models. Larger jumps modeled by a pure-jump process with exponentially weighted times between jumps such as the Aase model in (6.2.1) can be added as a second layer if needed, depending on the situation. Our assumptions lead us to the following specific GMRP formulation of S_t, where $C(t)$ is the number of subdivisions of the interval $[0,t]$, and the total number of subdivisions of the interval $[0,T]$ is N:

$$S_t = S_0 \prod_{k=1}^{C(t)} (1+\rho(x_k)) \tag{7.2.1}$$

We can also rephrase this as follows:

For a discrete Fixed Time Increments GMRP -modulated (B,S)- security market, we have the following bond and asset dynamics, where parameters are as in Section (7.1.1) and $B_0 > 0$, $S_0 > 0$:

$$\begin{cases} B_n = B_0(1+r)^n \\ S_n = S_0 \prod_{k=1}^{n} (1+\rho(x_k)) \end{cases} \tag{7.2.2}$$

7.2.2 Option Pricing, Optimal Stopping Times, and Solution [24]

7.2.2.1 Optimal Stopping Time

Assume we have the setup and notation as in Section 7.1.1, but with the Fixed Time Increments GMRP form as in Equation (7.2.2). We can formulate the pricing of options on assets with a GMRP model in the following way, where τ_N^* is an optimal stopping time, E is an expectation, and $g(s,x)$ is a $\mathbb{R} \times \chi$ measurable function, and we have the following expression for the risk-neutral price $C_N(s,x)$:

$$E_{s,x}g(S_{\tau_n^*}, x_{\tau_n^*}) = C_N(s,x) \tag{7.2.3}$$

It can be shown that $\tau_n^* := \min\{0 \leq m \leq n : C_{n-m}(S_m, x_m) = g(S_m, x_m)\}$ and $C_n(s,x) = E_{s,x}g(S_{\tau_n^*}, x_{\tau_n^*})$ under this model, where we have the operator T defined as follows:

$$Tg(s,x) := E_{s,x}g(S_1, x_1) = E_{s,x}g(s(1+\rho(x_1)), x_1) \tag{7.2.4}$$

This translates to τ^* being *exactly the first time* that the current value of the option (if exercised) is greater than the time-discounted expectation of the payoff at the next step (this involves summing over all possible next states). This can be observed given the payoff function of the option. Certainly, the current payoff if exercised will be equal to zero unless the option is "in the money", so this should occur within the "in the money" regions of the asset path. It follows, as expected, that the corresponding expectation in Equation (7.2.3) is indeed a risk-neutral valuation (as the Fixed Time Increments GMRP formulation results in a complete market), and under this probability, the discounted asset price is a F_n-martingale, where F_n is an appropriate filtration for the GMRP.

7.2.3 Optimal Stopping Times for American-Style Options [24]

7.2.3.1 Optimal Exercise Region, Stopping Time, and Valuation

Assuming that we have an American-style derivative on the stock modeled by Equation (7.2.2) with a payoff function of the following form, with $0 < \beta \leq 1$ and :

$$f_n = \beta^n g(S_n, x_n) \tag{7.2.5}$$

We have the result that

$$V_n := \max\{g(s,x); (1+r)^{-1}\beta T V_{n-1}(s,x)\} \text{ and } V_0(s,x) := g(s,x) \tag{7.2.6}$$

where $V_n$is the option price at time n, $g(s,x)$ is the payoff function divided by β^n, r is the risk-free interest rate, and the operator T is defined as follows:

$$Tg(s,x) = E^*_{s,x} g(s(1+\rho(x_1),x_1) = \int_X P(x,dy) g(s(1+\rho(y)),y) \tag{7.2.7}$$

The expectation in Equation (7.2.7) can be proven to be a Risk-Neutral measure, which is easy to calculate given a starting state, the functions ρ and g, and the Markov transition matrix.

Further, we have the following result on the optimal stopping time, τ_n; i.e. that τ_n is the first time such that the current value of the option (if exercised) is greater than the time-discounted expectation of the payoff at the next step:

$$\tau_n = \min\{0 \le m \le n : V_{n-m}(S_m, x_m) = g(s,x)\} \tag{7.2.8}$$

and we have the option valuation

$$E^*_{s,x}((1+r)^{-1}\beta)^{\tau_n} g(S_{\tau_n}, x_{\tau_n}) = V_n(s,x) \tag{7.2.9}$$

7.2.3.2 Guide to the MATLAB Code AOP.m

For purposes of simulation and observing the optimal exercise region, we have the MATLAB code AOP.m. The MATLAB code AOP.m takes as user input the number of states in the Markov chain, the transition probabilities, the number of iterations N (that is, the number of subdivisions of the entire interval), the function ρ(defined on each state of the Markov chain), the starting stock price, the starting state, the values of β and r, and the function $g(s,x)$. A Markov chain is simulated, and the values of $V_i : i = 1..N$ are calculated as presented in Equation (7.2.7). A price path is output for the user's convenience, as is V_n, as well as the optimal stopping time τ_n (in terms of the n^{th} iteration, as calculated in Equation (7.2.8)). Also output is a graph, which shows the price path of the stock, and shades the area before τ, the optimal stopping time.

Here we run the MATLAB code AOP.m using the following Markov transition matrix and data: $M = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$; $\rho(x_1) = 0.02$; $\rho(x_2) = -0.02$; N = 500, $x_0 = 1$; β = 0.9; r = 0.05; $S_0 = 20$; $g(s,x) = \max(s-22,0)$.

We are given that τ = 47; V_N= 0.86996, as well as the graph seen in Figure 7.2.1. Please note that

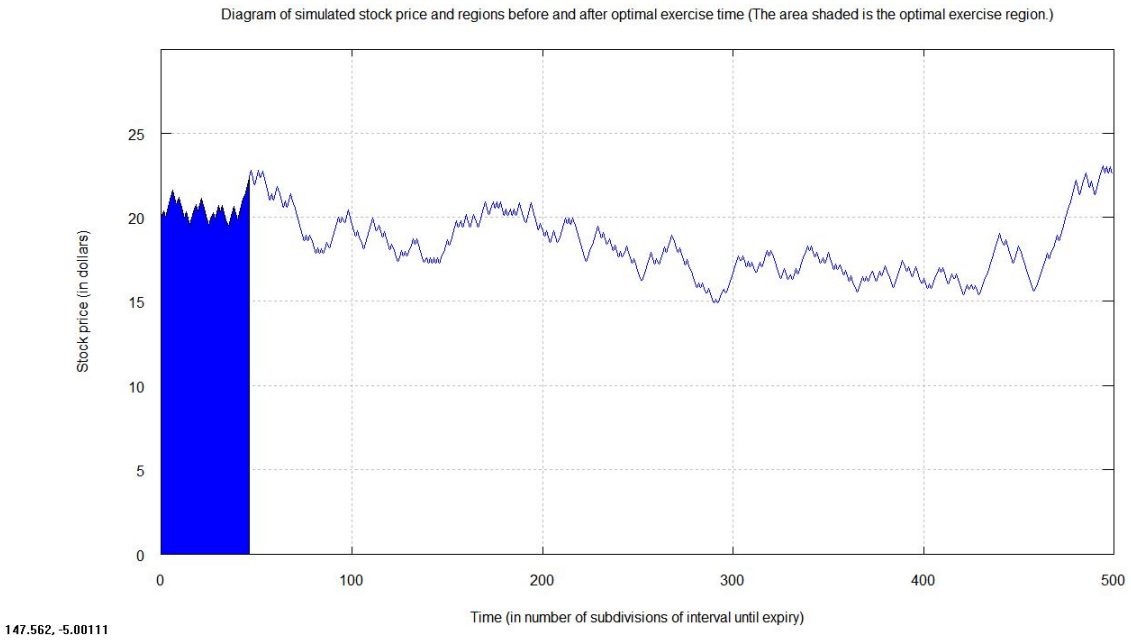

Figure 7.2.1: Graph produced by AOP.m showing pre-optimal exercise (blue) and post-optimal exercise regions of a simulated Fixed Time Increments GMRP path

this code is run in Octave rather than MATLAB. Octave is an open-source software with syntax nearly identical to MATLAB. Syntax is identical for purposes of this code. The code AOP.m can be found in Appendix B.3

7.2.4 Limit Case When $N \to +\infty$

7.2.4.1 Statement of the Problem

A result of interest is that the value of an American-style derivative of a stock under Fixed Time Increments GMRP assumptions, as we let N tend to infinity, then the price must satisfy the following equation:

$$V(s,x) = (1+r)^{-N}\beta \int_X P(x,dy)V(s(1+\rho(y)),y) \tag{7.2.10}$$

Financially, this represents a perpetual American-style option. We will present an algorithm for solving this problem, as well as a MATLAB code which finds V_N. This statement of the problem is presented by Swishchuk and Limnios [24] in their 2011 results as a problem to be solved. We

solve it here in the next sections in the case of Black-Scholes assumptions in the different Markov states.

7.2.4.2 Theoretical Research and Formulation as a Optimization Problem

We will solve the integral equation given in Equation (7.2.10) given some assumptions on the behavior of the stock. However, it will require some manipulation of the variables, exploiting their relationships to one another.

To solve for $V(s,x)$ in Equation (7.2.10), we note that the integral is actually a sum over all next states y of the Markov chain, with the stock price appropriately modified by a factor of $(1+\rho(y))$. Note that since this holds for all possible starting states of the stock, we must have that the following equations hold true, where S_0 is the starting stock price, $x_i : i = 1...D$ are the D states of the Markov chain with transition matrix M, $M_{i,j}$is the $(i,j)^{th}$entry of M (i.e. the probability of going from state i to state j), and $\chi = \frac{\beta}{(1+r)}$:

$$\begin{aligned}
V(S_0,x_1) &= \chi[V(S_0(1+\rho(x_1)),x_1)M_{1,1} + V(S_0(1+\rho(x_2)),x_2)M_{1,2} + ... + V(S_0(1+\rho(x_D)),x_D)M_{1,D}] \\
V(S_0,x_2) &= \chi[V(S_0(1+\rho(x_1)),x_1)M_{2,1} + V(S_0(1+\rho(x_2)),x_2)M_{2,2} + ... + V(S_0(1+\rho(x_D)),x_D)M_{2,D}] \\
&\vdots \\
V(S_0,x_D) &= \chi[V(S_0(1+\rho(x_1)),x_1)M_{D,1} + V(S_0(1+\rho(x_2)),x_2)M_{D,2} + ... + V(S_0(1+\rho(x_D)),x_D)M_{D,D}]
\end{aligned} \tag{7.2.11}$$

Now, assume that the stock price is in state x_i. We assume $(1+\rho(x_i))$ is small for all states x_i. It is a reasonable assumption that we can find bounds s_i and t_i such that there exists a number $\delta_i \in [s_i,t_i]$ such that $V(S_0,x_i)+\delta_i = V(S_1,x_i)$, where $S_1 = S_0(1+\rho(x_i))$. Indeed, an appropriate measure would be to make Black-Scholes assumptions on the market when it remains in one state, and the influence of slight differences in the starting stock price S_0on $V(S_0,x_i)$ in the Black-Scholes model can be used to approximate the value of δ_i. Hence, after dividing by χ, the Equations (7.2.11) become:

$$\begin{aligned}
\tfrac{1}{\chi}V(S_0,x_1) &= (V(S_0,x_1)+\delta_1)M_{1,1}+(V(S_0,x_2)+\delta_2)M_{1,2}+\ldots+(V(S_0,x_D)+\delta_D)M_{1,D}\\
\tfrac{1}{\chi}V(S_0,x_2) &= (V(S_0,x_1)+\delta_1)M_{2,1}+(V(S_0,x_2)+\delta_2)M_{2,2}+\ldots+(V(S_0,x_D)+\delta_D)M_{2,D}\\
&\vdots\\
\tfrac{1}{\chi}V(S_0,x_D) &= (V(S_0,x_1)+\delta_1)M_{D,1}+(V(S_0,x_2)+\delta_2)M_{D,2}+\ldots+(V(S_0,x_D)+\delta_D)M_{D,D}
\end{aligned} \tag{7.2.12}$$

By expanding the sums and re-arranging, the Equations (7.2.12) become the following:

$$\begin{aligned}
\sum_{i=1}^{D}\delta_i M_{1,i} &= (M_{1,1}+\tfrac{1}{\chi})V(S_0,x_1)+(M_{1,2}+\tfrac{1}{\chi})V(S_0,x_2)+\ldots+(M_{1,D}+\tfrac{1}{\chi})V(S_0,x_D)\\
\sum_{i=1}^{D}\delta_i M_{2,i} &= (M_{2,1}+\tfrac{1}{\chi})V(S_0,x_1)+(M_{2,2}+\tfrac{1}{\chi})V(S_0,x_2)+\ldots+(M_{2,D}+\tfrac{1}{\chi})V(S_0,x_D)\\
&\vdots\\
\sum_{i=1}^{D}\delta_i M_{D,i} &= (M_{D,1}+\tfrac{1}{\chi})V(S_0,x_1)+(M_{D,2}+\tfrac{1}{\chi})V(S_0,x_2)+\ldots+(M_{D,D}+\tfrac{1}{\chi})V(S_0,x_D)
\end{aligned} \tag{7.2.13}$$

Now, writing in matrix notation, we have that Equations (7.2.13) become:

$$M\begin{bmatrix}\delta_1\\ \delta_2\\ \vdots\\ \delta_D\end{bmatrix} = (M*)\begin{bmatrix}V(S_0,x_1)\\ V(S_0,x_2)\\ \vdots\\ V(S_0,x_D)\end{bmatrix} \tag{7.2.14}$$

where $M* = M-\frac{1}{\chi}I$ (I is the identity matrix of size D). Assuming that $\frac{1}{\chi}=\frac{\beta}{1+r}$ is not an eigenvalue of M, we have that $M*$ is invertible. Hence, writing $Z=(M*)^{-1}M$, Equation (7.2.14) becomes:

$$\begin{bmatrix}V(S_0,x_1)\\ V(S_0,x_2)\\ \vdots\\ V(S_0,x_D)\end{bmatrix} = Z\begin{bmatrix}\delta_1\\ \delta_2\\ \vdots\\ \delta_D\end{bmatrix} \tag{7.2.15}$$

Since the stock prices must always remain positive, and the delta-vector is bounded above and below by the values of s_i and t_i, solving for $V(S_0,x_1)$ through $V(S_0,x_D)$ in Equation (7.2.15) becomes

a linear optimization (LP) problem. If we want to find bounds on $V(S_0,x_i)$, where $i \in \{1,2,...D\}$, the question can be posed as follows:

$$\text{Given } i \in [1,2,...,D],\ \text{maximize and minimize } V(S_0,x_i),\ \text{subject to } \begin{bmatrix} V(S_0,x_1) \\ V(S_0,x_2) \\ \vdots \\ V(S_0,x_D) \end{bmatrix} = Z \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_D \end{bmatrix};$$
$$\text{where } \begin{bmatrix} V(S_0,x_1) \\ V(S_0,x_2) \\ \vdots \\ V(S_0,x_D) \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_D \end{bmatrix} \leq \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_D \end{bmatrix} \leq \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_D \end{bmatrix}. \tag{7.2.16}$$

In the MATLAB language of linear optimization, Equations (7.2.16) are equivalent to:

$$\text{Given } i \in [1,2,...,D],\ \text{ find } \max\{V(S_0,x_i)\} \text{ and } \min\{V(S_0,x_i)\} \text{ such that:}$$

$$\begin{bmatrix} I & -Z \\ 0_{DxD} & 0_{DxD} \end{bmatrix} \begin{bmatrix} V(S_0,x_1) \\ V(S_0,x_2) \\ \vdots \\ V(S_0,x_D) \\ \delta_1 \\ \delta_2 \\ \vdots \\ \delta_D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ s_1 \\ s_2 \\ \vdots \\ s_D \end{bmatrix} \leq \begin{bmatrix} V(S_0,x_1) \\ V(S_0,x_2) \\ \vdots \\ V(S_0,x_D) \\ \delta_1 \\ \delta_2 \\ \vdots \\ \delta_D \end{bmatrix} \leq \begin{bmatrix} \infty \\ \infty \\ \vdots \\ \infty \\ t_1 \\ t_2 \\ \vdots \\ t_D \end{bmatrix} \tag{7.2.17}$$

(0_{DxD} is the zero matrix of size D by D.)

Now we have reduced this problem to one that can be solved by linear optimization algorithms which are already functions in MATLAB. We have re-formulated the problem of solving a iterative summation of a function with itself into a problem of optimization. We write a MATLAB code, exhibited in the next sections, which can solve this problem.

7.2.4.3 Guide to MATLAB Code ISol.m

The MATLAB code ISol.m takes as user input the number of states in the Markov chain, the transition probabilities, the function $\rho(x)$ (defined on each state of the Markov chain), the starting stock price, the starting state, the values of β and r, and all values of s_i and t_i for $i = 1, 2, ..., D$. Then, the *linprog* command is used to find $\max\{V(S_0, x_i)\}$ and $\min\{V(S_0, x_i)\}$, for starting state i. The code outputs upper and lower bounds on the price of the option, $V(S_0, x_i)$. Here we run the MATLAB code ISol.m using the following Markov transition matrix and data:

$$M = \begin{bmatrix} 0.3 & 0.3 & 0.4 \\ 0.3 & 0.4 & 0.3 \\ 0.4 & 0.3 & 0.3 \end{bmatrix}; \rho(x_1) = 0.02;\ \rho(x_2) = 0.01;\ \rho(x_3) = -0.02;\ \text{starting state} = x_2;\ S_0 = 20;\ \beta = 0.9;\ \text{r} = 0.05;$$

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.2 \\ -0.5 \end{bmatrix};\ \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} 1.1 \\ 0.3 \\ -0.4 \end{bmatrix}.$$

We are given that $V(S_0, x_2) \in [1.3969, 1.9969]$ as seen in the output in Figure 7.2.2. Please note that this code is run in Octave rather than MATLAB. Octave is an open-source software with syntax nearly identical to MATLAB. Syntax is identical for purposes of this code. The MATLAB code ISol.m can be found in Appendix B.4.

7.3 European Option Pricing Formulas for Fixed Time Increments GMRP -Modulated (B, S)-Security Market [24]

We have the results that for a European option with payoff function $f(S_N)$. Assume that the bond B_N and asset S_N follow the following, where where N is a maturity date:

$$\begin{cases} B_N = B_0(1+r)^N \\ S_N = S_0 \prod_{k=1}^{N} (1 + \rho(x_k)) \end{cases} \tag{7.3.1}$$

Then the price may be found by the formula where $P(x, dy)$ is a risk-neutral expectation, i.e. $\int_X P(x, dy)\rho(y) = r$.

```
M =

   0.30000   0.30000   0.40000
   0.30000   0.40000   0.30000
   0.40000   0.30000   0.30000

Enter function rho defind on each state of the Markov chain:
Rho 1 =  .02
Rho 2 =  .01
Rho 3 =  -.02
Enter the starting stock price:  20
Enter the starting state:  2
XO =  2
Enter the value of beta:  0.9
Enter the interest rate r:  .05
We assume that for small changes in the stock price, we have that
given a state x_i,
V(S_1,x_i) - V(S_0,x_i) = delta_i
for some real (small) number delta_i, where S_1 = S_0 (1+rho(x_i)).
Please enter the lower bound, s_i, and
 the upper bound, delta_i, for each state of the chain,
 so that s_i <= delta_i <=t_i for each of i from 1 to D.
 Now, enter s_i and t_i one by one: s_1 =  1
t_1 =  1.1
s_2 =  .2
t_2 =  .3
s_3 =  -.5
t_3 =  -.4
Z =

   1.9762   1.9687   2.0551
   1.9687   2.0625   1.9687
   2.0551   1.9687   1.9762

The lower bound is:
 1.3969
The upper bound is:
 1.9969
>>>
```

Figure 7.2.2: Output of ISol.m

$$C_T(x) = E^*(1+r)^{-N} f(S_N) = \sum_{k=0}^{N} (1+r)^{-N} \int_X \int_X \dots \int_X f(S_0 \prod_{i=1}^{k} (1+\rho(x_i)), K) P(x_{i-1}, dx_i) \quad (7.3.2)$$

We have the following formula for the price $C_T(x)$ of a European-style option, with the stock price modeled as in Equation 7.1.1, i.e. in the more general case of an arbitrary counting process $\nu(t)$:

$$C_T(x) = \sum_{k=0}^{\infty} P\{\upsilon(t) = k\} \int_X \int_X \dots \int_X f(S_0 \prod_{i=1}^{k} (1+\rho(x_i)), K) Q(x_{i-1}, x_i)$$
$$f(S,K) = \begin{cases} (S-K)^+ = \max(S-K, 0) & \text{for call options} \\ (K-S)^+ = \min(K-S, 0) & \text{for put options} \end{cases} \quad (7.3.3)$$

Here $P\{\upsilon(t) = k\}$ is the probability distribution of the jump process (often taken to be Poisson), and $Q(x,y)$ is the probability of going from state x to state y, i.e., the $(x,y)^{th}$ entry of the transition probability matrix M. $S_0$is the starting stock price,K is the exercise price, and ρ is a function defined on the phase space so that $\rho(x_k) > -1$.

7.3.1 Guide to the MATLAB Code EuroGO.m

The MATLAB code EuroGO.m takes as user input the number of states in the Markov chain, the transition probabilities, the function ρ(defined on each state of the Markov chain), the starting stock price, the starting state, and the values of β and r. Furthermore, it requests whether to use the Poisson distribution to model the number of jumps, in which case the intensity parameter $\lambda$and the time to expiry T are taken as user input, or to use some other user-defined jump distribution. The user also inputs whether the option is a call or a put, as well as its exercise price. The user is also asked to input the error tolerance as well as the maximum number of iterations allowed (if this number is achieved without the error tolerance condition being satisfied, the code will output an error message). It is useful to note that the convergence of the formula in Equation 7.3.3 is not dealt with here.

The code EuroGO.m generates all possible paths for each number of jumps as row vectors, and calculates the expectations according to these paths. These values are used to calculate the quantity in Equation 7.3.3 to within the user-input error tolerance, provided the maximum number of

iterations are not exceeded. Note that the code ensures that before termination, even if the error tolerance condition is satisfied, the previous entry is also positive, to avoid the scenario in which enumeration of possible Markov paths results in a stock price path slightly above the call option's exercise price (or in the case of a put, below) on the k^{th}iteration, such that that the difference between this and the answer in the previous iteration (which is zero) results in a number which is less than the input error tolerance level. The code outputs the value of C_T along with the number of iterations performed.

Here we run the MATLAB code EuroGO.m using the following Markov transition matrix and data: $M = \begin{bmatrix} 0.4 & 0.6 \\ 0.6 & 0.4 \end{bmatrix}$; $\rho(x_1) = 0.02$; $\rho(x_2) = -0.02$; starting state =x_1; $S_0 = 20$; β = 0.9; r = 0.05; $K = 22$; option type = call; error tolerance = 0.01, maximum number of iterations = 30;

Poisson distribution of number of jumps with λ=2, T = 1

We are given that $C_T(x)$ = 4.7227 in 13 iterations. The results can be seen in Figure 7.3.1. Please note that this code is run in Octave rather than MATLAB. Octave is an open-source software with syntax nearly identical to MATLAB. Syntax is identical for purposes of this code. The MATLAB code EuroGO.m can be found in Appendix B.5.

```
>>>Enter the number of states in the Markov chain: \n 2
Prob to go from state 1 to 1 .4
Prob to go from state 1 to 2 .6
Prob to go from state 2 to 1 .6
Prob to go from state 2 to 2 .4
Matrix for Markov chain looks like this:
M =

  0.40000  0.60000
  0.60000  0.40000

Enter function rho defind on each state of the Markov chain:
Rho 1 = .02
Rho 2 = -.02
Enter 1 if Poisson distribution is used to model jumps; enter 2 otherwise: \n 1
Enter lambda : \n 2
Enter the time to expiry, T: \n 1
Enter 1 for a call option; enter 2 for a put option: \n 1
Enter the exercise price: \n 22
Enter the starting stock price: 20
Enter the starting state: 1
XO = 1
Enter the value of beta: \n .9
Enter the interest rate r: .05
Enter the maximum amount of iterations: \n (NOTE: The recommended value is 20.)
 \n 30
Enter epsilon : \n .01
continu = 0
Value of the option is:
 4.7227
Number of iterations, i.e. k:
 13
>>>
```

Figure 7.3.1: Output of EuroGO.m

Chapter 8

GMRP Approximations

8.1 Formal GMRP presentation [21]

Let (Ω, B, F_t, P) be a probability space and let F_t be a filtration, and $(x_k)_{k\in\mathbb{Z}_+}$ be a Markov chain in phase space (X, χ). We have the transition probability matrix $P(x,A)$ for $x \in X$, $A \in \chi$. Let $(\theta_k)_{k\in\mathbb{Z}_+}$ be a renewal process; a sequence of i.i.d. r.v. with distribution $F(x) := P\{\omega : \theta_k(\omega) \le x\}$. Put

$$\tau_k := \sum_{n=0}^{k} \theta_n \tag{8.1.1}$$

Then $\nu(t) := \sup\{k : \tau_k \le t\}$ is the counting process.

Definition 8.1. A homogeneous 2-dimensional Markov chain $(x_n, \theta_n)_{n\in\mathbb{Z}_+}$ on the phase space $X \times R_+$ is called a Markov renewal process (MRP) if the semi-Markov kernel gives the transitional probabilities:

$$Q(x,A,t) = P\{x_{n+1} \in A, \theta_{n+1} \le t | x_n = x\},\ \forall x \in X, A \in \chi, t \in R_+ \tag{8.1.2}$$

Definition 8.2. This process is a semi-Markov process:

$$x(t) := x_{\nu(t)} \tag{8.1.3}$$

Let $(x_n, \theta_n)_{n\in\mathbb{Z}_+}$ be a Markov renewal process; let $X \times R_+$ be the phase space; let the semi-Markov kernel be $Q(x,A,t)$ and let $x(t) := x_{\nu(t)}$ be a semi-Markov process; here $\nu(t)$ is the counting process for the GMRP. Provided $\rho(x) > -1$ for each state x, we can define the GMRP $\{S_t\}_{t\in R_+}$ as a stochastic functional S_t, where S_0 is the initial value of S_t:

$$S_t := \prod_{k=1}^{\nu(t)} (1+\rho(x_k)),\ t \in R_+ \tag{8.1.4}$$

This is directly analogous with the so-called Geometric Compound Poisson Process (see Equation 6.2.1) defined as below, where $S_0^* > 0$, $N(t)$ is a standard Poisson process. $(Y_k)_{k\in\mathbb{Z}_+}$ are i.i.d.r.v.:

$$S_t^* := S_0^* \prod_{k=1}^{N(t)} (1+Y_k) \tag{8.1.5}$$

8.2 Ergodic GMRP Approximation [21]

Let the GMRP S_t with Markov chain $(x_n)_{n\in\mathbb{Z}_+}$ have stationary distribution $p(A)$, $A \in \chi$. We shall approximate the behavior of S_t by using the ergodicity (or near-ergodicity) of the underlying Markov chain $(x_n)_{n\in\mathbb{Z}_+}$. Approximations of GMRP are extremely useful when looking at long-term behavior, implying that we should look at large time scale behavior. The idea is to speed up time in the counting process $\nu(t)$, thus effectively observing large swaths of GMRP-modulated data. We introduce the time scaling factor $T > 0$ and consider $\nu_T(t) := \nu(tT)$ in "fast time". we shall observe the effect of letting $T \to \infty$ with the assumption that S_t is T-dependent. This induces us that $\rho(x)$ should depend on T (as $S_{\tau_k} - S_{\tau_k-} = S_{\tau_k-}\rho(x_k)$), i.e. $\rho \equiv \rho_T(x)$ so that $\rho_T(x) \to 0$ uniformly by x. This can be summarized as follows $\forall x \in X$:

$$\rho_T(x) = \frac{\rho(x)}{T} \tag{8.2.1}$$

In this way, S_t is described as:

$$S_t^T = S_0 \prod_{k=0}^{\nu(tT)} (1+\rho_T(x_k)) := S_0 \prod_{k=0}^{\nu(tT)} (1+T^{-1}\rho_T(x_k)) \tag{8.2.2}$$

Denoting $m := \int_X p(dx)\bar{m}(x)$ and $\bar{m}(x) := \int_X (1 - G_x(t))dt$ where $G_x(t) = P(\tau_{n+1} - \tau_n < t | x_n = x)$ and $\hat{\rho} := \int_X p(dx)\rho(x)/m$, if $\int_X p(dx)(\rho(x))^2 < +\infty$ we claim that the GMRP has the following form $\forall t \in R_+$, $S_0 > 0$, giving a characterization in approximation of an ergodic GMRP as a bond with interest rate $\hat{\rho}$:

$$S_t \approx \hat{S}_t = S_0 e^{\hat{\rho} t} \tag{8.2.3}$$

To derive this result, we must expand Equation (8.2.2):

$$S_t^T = S_0 \exp\left\{ \sum_{k=0}^{\nu(tT)} \ln\left(1 + \frac{\rho(x)}{T}\right)\right\} \tag{8.2.4}$$

which can be re-written as

$$\ln \frac{S_t^T}{S_0} = \sum_{k=0}^{\nu(tT)} \ln\left(1 + \frac{\rho(x_k)}{T}\right) \tag{8.2.5}$$

For large T we have that $\frac{\rho(x)}{T}$ is small, and using a Taylor expansion yields the following, where the function r tends to 0 as $T \to +\infty$.

$$\ln\left(1 + \frac{\rho(x)}{T}\right) \approx \frac{\rho(x)}{T} - \left(\frac{1}{2}\right)\left(\frac{\rho(x)}{T}\right)^2 + r\left(\frac{\rho(x)}{T}\right)\left(\frac{\rho(x)}{T}\right)^2 \tag{8.2.6}$$

By the above and (8.2.2) we have

$$\ln \frac{S_t^T}{S_0} \approx \frac{1}{T}\sum_{k=0}^{\nu(tT)} \rho(x_k) - \frac{1}{2T^2}\sum_{k=0}^{\nu(tT)} (\rho(x_k))^2 + \frac{1}{T^2}\sum_{k=0}^{\nu(tT)} r\left(\frac{\rho(x)}{T}\right)(\rho(x_k))^2 \tag{8.2.7}$$

Noting that $\nu(tT)$ has order of growth of $\frac{tT}{m}$, the last 2 terms in (8.2.7) tend to 0 as $T \to +\infty$. Using existing algorithms of phase averaging for Markov chains, the first term of the right-hand of (8.2.7) has the following limit:

$$\frac{1}{T}\sum_{k=0}^{\nu(tT)} \rho(x_k) \to \hat{\rho} t \text{ as } T \to +\infty \text{ where } \hat{\rho} := \int_X p(dx)\rho(x)/m \tag{8.2.8}$$

Thus, summing up the approximated terms in (8.2.7) we have

$$\ln \frac{S_t^T}{S_0} \to \hat{\rho} t \text{ as } T \to +\infty \tag{8.2.9}$$

and as claimed, defining

$$\hat{S}_t := \lim_{T \to \infty} S_t \tag{8.2.10}$$

and we have, for all $t \in R_+$:

$$S_t \approx \hat{S}_t = S_0 e^{\hat{\rho} t} \text{ as } T \to +\infty \tag{8.2.11}$$

8.3 Merged GMRP Approximation [21]

Suppose that a GMRP has a Markov state space X consisting of r ergodic classes X_k ($k = 1, 2, ..., r$), and with stationary distribution $p_i(dx)$ in each class. Then we merge the Markov chain x_k to the Markov chain $\hat{x}_k$ in the merged phase space defined $\hat{X} = \{1, 2, ..., r\}$. Using phase merging algorithms and defining $\tilde{\rho}(t) := \int_X \hat{\rho}(\hat{x}(s))ds$ and the important parameter $\hat{\rho}(k)$ as follows,

$$\hat{\rho}(k) := \int_{X_k} p_k(dx)\rho(x)/m(k) \tag{8.3.1}$$

and finally denoting $m(k) := \int_{X_k} p_k(dx)m(x)$, then $\hat{x}(s)$ is a merged Markov process in phase states of space $\hat{X}$. We obtain:

$$\ln \frac{S_t^T}{S_0} \to \int_0^t \hat{\rho}(\hat{x}(s))ds \text{ as } T \to +\infty \tag{8.3.2}$$

Thus $S_t^T \to \tilde{S}_t$ as $T \to +\infty$ and we have:

$$\tilde{S}_t = S_0 e^{\int_0^t \hat{\rho}(\hat{x}(s))ds} \tag{8.3.3}$$

or

$$\tilde{S}_t = S_0 e^{\rho\tilde{(}t)} \tag{8.3.4}$$

The above has the following interpretation: The merged GMRP exhibits the dynamics of a bond with different interest rates (represented by the various values of $\hat{\rho}(k)$ for each state k).

8.4 Double Averaged GMRP [21]

Suppose that the merged phase space $\hat{X}$ of the merged Markov process $\hat{x}(t)$ has one ergodic class and that the stationary probabilities are given by $(\hat{p}_k)_{k=1,2,\ldots,r}$. Algorithms of double averaging yield that

$$\frac{1}{T}\sum_{k=0}^{\nu(tT)} \rho(x_k) \to \breve{\rho} t \text{ as } T \to +\infty \tag{8.4.1}$$

where, with $\hat{\rho}(k)$ is defined as in Equation (8.3.1)

$$\breve{\rho} := \sum_{k=1}^{r} \int_X \hat{p}_k \hat{\rho}(k) \tag{8.4.2}$$

Hence

$$\ln \frac{S_t^T}{S_0} \to \ln \frac{\breve{S}_t}{S_0} = t\breve{\rho} \text{ as } T \to +\infty \tag{8.4.3}$$

Note that as

$$\lim_{T\to\infty} S_t^T = \breve{S}_t$$

we have the following result about double averaged GMRP, for $t \in R_+$ and $S_0 > 0$ (with the financial interpretation that double averaged GMRP exhibits the dynamics of a bond price with interest rate $\breve{\rho}$):

$$\breve{S}_t = S_0 e^{t\breve{\rho}} \tag{8.4.4}$$

8.5 Diffusion Approximation of GMRP

8.5.1 Ergodic Diffusion [20]

Here we present results on the diffusion approximation for a GMRP where the underlying Markov chain has one ergodic state. Suppose that the balance condition

$$\hat{\rho} = \frac{\int_X p(dx) \int_X P(x,dy)\rho(y)}{m} = 0 \tag{8.5.1}$$

is fulfilled for S_t as below, where $p(x)$ is the ergodic distribution of the Markov chain $(x_k)_{k\in\mathbb{Z}_+}$:

$$S_t^T = S_0 \prod_{k=1}^{\nu(tT)} (1+\rho_T(x_k)) \tag{8.5.2}$$

Now we construct S_t^T in the new time scale tT^2, to make the process $S_T(t)$ fluctuate near the point S_0 as $T \to \infty$:

$$S_T(t) := S_{tT^2}^T = S_0 \prod_{k=1}^{\nu(tT^2)} (1+T^{-1}\rho(x_k)) \tag{8.5.3}$$

The log presentation of the asset price yields the following, where $r \to 0$ as $T \to \infty$:

$$\ln \frac{S_T(t)}{S_0} = T^{-1} \sum_{k=1}^{\nu(tT^2)} \rho(x_k) - \frac{1}{2}T^{-2} \sum_{k=1}^{\nu(tT^2)} \rho^2(x_k) + T^{-2} \sum_{k=1}^{\nu(tT^2)} (T^{-1}\rho(x_k))\rho^2(x_k) \tag{8.5.4}$$

resulting in the following limit:

$$\lim_{T\to\infty} \frac{1}{2}T^{-2} \sum_{k=1}^{\nu(tT^2)} \rho^2(x_k) = \frac{1}{2}t\hat{\rho}_2 \tag{8.5.5}$$

The second term in Equation (8.5.4) has the following limit:

$$\hat{\rho}_2 := \int_X p(dx) \int_X P(x,dy)\rho^2(y)/m \tag{8.5.6}$$

and using calculation algorithms for diffusion approximation in the first term in (8.5.4) yields the

following limit, where $\omega(t)$ is a standard Wiener process:

$$\lim_{T\to+\infty}\frac{1}{T}\sum_{k=1}^{\nu(tT)}\rho(x_k)=\sigma_\rho\omega(t) \tag{8.5.7}$$

where we have the following parameter, where R_0 is the potential of $(x_n)_{n\in\mathbb{Z}_+}$:

$$\sigma_\rho^2:=\int_X p(dx)\left[\frac{1}{2}\int_X P(x,dy)\rho^2(y)+\int_X P(x,dy)\rho(y)R_0P(x,dy)\rho(y)\right]\frac{1}{m}$$

The last term in Equation (8.5.4) has limit 0 as $T\to\infty$. Hence, if we denote

$$\hat{S}(t):=\lim_{T\to\infty}S_T(t) \tag{8.5.8}$$

we have that

$$\lim_{T\to\infty}\ln\frac{S_T(t)}{S_0}=\ln\frac{\hat{S}(t)}{S_0}=\sigma_\rho\omega(t)-\frac{1}{2}t\hat{\rho}_2 \tag{8.5.9}$$

which is a GBM approximation. This is summed up as follows: In the case of ergodic diffusion approximation of a GMRP, we have the following GBM approximation:

$$\lim_{T\to\infty}S_T(t)=\hat{S}(t)=S_0e^{-\frac{1}{2}t\hat{\rho}_2+\sigma_\rho\omega(t)} \tag{8.5.10}$$

and the following stochastic differential equation:

$$\frac{d\hat{S}(t)}{\hat{S}(t)}=\frac{1}{2}(\sigma_\rho^2-\hat{\rho}_2)dt+\sigma_\rho d\omega(t) \tag{8.5.11}$$

8.5.2 Merged Diffusion [20]

Here we present the setup in detail for the merged diffusion approximation of a GMRP. Suppose we have several ergodic states in the merged Markov chain, and the following so-called balance condition is satisfied for all states $k=1,2,...,r$ and $(x_n)_{n\in\mathbb{Z}_+}$ of this "embedded" Markov chain, where p_k is the stationary density for the ergodic component X_k:

$$\hat{\rho}(k) = \frac{\int_{X_k} p_k(dx) \int_{X_k} P(x,dy)\rho(y)}{m(k)} \tag{8.5.12}$$

Furthermore, we assume the conditions of reducibility of X are satisfied. Existing algorithms of merged averaging yield the following limit for the second term in the right hand side of (8.5.4):

$$\lim_{T\to\infty} \frac{1}{2} T^{-2} \sum_{k=1}^{\nu(tT)} \rho^2(x_k) = \frac{1}{2} \int_0^t \hat{\rho}_2(\hat{x}(s))ds \tag{8.5.13}$$

where

$$\hat{\rho}_2(k) := \frac{\int_X p_k(dx) \int_{X_k} P(x,dy)\rho^2(y)}{m(k)} \tag{8.5.14}$$

Now, existing algorithms of merged averaging yield the following limit for the first term in the right hand side of (8.5.4):

$$\lim_{T\to\infty} = \frac{1}{2} T^{-1} \sum_{k=1}^{\nu(tT^2)} \rho(x_k) = \int_0^t \hat{\sigma}_\rho(\hat{x}(s))d\omega(s) \tag{8.5.15}$$

using the notation

$$\hat{\sigma}_\rho^2(k) := \int_{X_k} p_k(dx) \int_{X_k} P(x,dy)\rho^2(y) + \int_{X_k} P(x,dy)\rho(y)R_0 \int_{X_k} \frac{P(x,dy)}{m(k)}$$

The third term in (8.5.4) has a limit of 0 as $T \to \infty$. So summing up, for (8.5.4), we have the following limit, in the logarithmic asset return notation:

$$\lim_{T\to\infty} \ln \frac{S_T(t)}{S_0} = \ln \frac{\tilde{S}(t)}{S_0} = \int_0^t \hat{\sigma}_\rho(\hat{x}(s))d\omega(s) - \frac{1}{2} \int_0^t \hat{\rho}_2(\hat{x}(s))ds \tag{8.5.16}$$

where we have denoted

$$\lim_{T\to\infty} S_T(t) = \tilde{S}(t)$$

Summarizing, for the case of the merged diffusion approximation, we have the following approximation of the GMRP with a GBM:

$$\lim_{T\to\infty} S_T(t) = \tilde{S}(t) = S_0 e^{-\frac{1}{2}\int_0^t \hat{\rho}_2(\hat{x}(t))ds + \int_0^t \hat{\sigma}_\rho(\hat{x}(s))d\omega(s)} \tag{8.5.17}$$

and

$$\frac{d\tilde{S}(t)}{\tilde{S}(t)} = \frac{1}{2}(\hat{\sigma}_\rho(\hat{x}(t)) - \hat{\rho}_2(\hat{x}(t)))dt + \hat{\sigma}_\rho(\hat{x}(t))d\omega(t) \tag{8.5.18}$$

where $\hat{x}(t)$ is the merged Markov process. The financial interpretation of the above is that $S_T(t)$ behaves as a GBM with values of the parameters switching along with the state in which the Markov chain the process is currently in.

8.5.3 Diffusion in the Case of Double Averaging [20]

Suppose that the phase space $\hat{X} = \{1,2,...,r\}$ of the Markov process $\hat{x}(t)$ consists of one ergodic class with stationary distribution $(\hat{p}_k; k = 1,2,...,r)$, and that the following balance condition is also satisfied:

$$\sum_{k=1}^{r} \hat{p}_k \hat{\rho}(k) = 0 \tag{8.5.19}$$

Using the notation

$$\lim_{T\to\infty} S_T(t) = \breve{S}(t) \tag{8.5.20}$$

we have, after using a Taylor expansion, the limit (in convergence in distribution)

$$\lim_{T\to\infty} \ln\frac{S_T(t)}{S_0} = \ln\frac{\breve{S}(t)}{S_0} = \breve{\sigma}_\rho \omega(t) - \frac{1}{2}t\breve{p}_2 \tag{8.5.21}$$

where we define the following new parameters (using the same notation as in section 8.5.2 and 8.4):

$$\breve{\sigma}_\rho^2 := \sum_{k=1}^{r} \hat{p}_k \hat{\sigma}_\rho^2(k) \text{ and } \breve{p}_2 := \sum_{k=1}^{r} \hat{p}_k \hat{\rho}_2(k)$$

Thus we have the following GBM approximation of the behavior of $S_T(t)$:

$$\lim_{T\to\infty} S_T(t) = \breve{S}(t) = S_0 e^{-\frac{1}{2}t\breve{\rho}_2 + \breve{\sigma}\rho\omega(t)} \tag{8.5.22}$$

i.e.

$$\frac{d\breve{S}(t)}{\breve{S}(t)} = \frac{1}{2}(\breve{\sigma}_\rho^2 - \breve{\rho}_2)dt + \breve{\sigma}_\rho d\omega(t) \tag{8.5.23}$$

8.5.4 Option Pricing Under the Ergodic and Double Averaged GMRP Approximations

8.5.4.1 European Option Valuation

For the cases of the Ergodic Diffusion (8.5.11) and Double Averaged Diffusion (8.5.3), since we have (in both cases) approximated the GMRP in the form of a stochastic differential equation for a GBM, we can directly use the Black-Scholes formulas to find the price for European calls and puts, after appropriate considerations for a risk-neutral valuation.

Since we have the result from (8.5.11) that

$$\frac{d\hat{S}(t)}{\hat{S}(t)} = \frac{1}{2}(\sigma_\rho^2 - \hat{\rho}_2)dt + \sigma_\rho d\omega(t)$$

and the analogous expression for double averaged diffusion in (8.5.3), we can calculate the risk-neutral probability measure P^* using Girsanov's theorem and the Radon-Nikodym derivative [19] as follows:

$$\frac{dP^*}{dP} = \exp\left\{ -\theta\omega(t) - \frac{1}{2}\theta^2 t \right\} \tag{8.5.24}$$

where we have as the market price of risk

$$\theta = \frac{\frac{1}{2}(\sigma_\rho^2 - \hat{\rho}_2) - r}{\sigma_\rho} \tag{8.5.25}$$

By established results, the following discounted process is a P^*- martingale:

$$e^{-rt}\hat{S}_t \tag{8.5.26}$$

and the following process is a Brownian motion (with drift).

$$\omega^*(t) = \omega(t) + \theta t \tag{8.5.27}$$

Hence viewing our results in the risk-neutral world, the process $\hat{S}_t$ has the following representation:

$$\frac{d\hat{S}(t)}{\hat{S}(t)} = rdt + \sigma_\rho d\omega^*(t) \tag{8.5.28}$$

and we may directly substitute the value for σ_ρ into our Black-Scholes formulas for d_1 and d_2, and proceed with European call and European put valuation. See parts (4.2.6), (4.2.5), and (4.2.8). This solves the valuation problem of valuation of both European calls and European puts.

8.5.4.2 American Option Valuation

We can use the risk-neutral Black-Scholes valuation for an American call, since an American call has the same value as a European call. For American puts, we can employ the method of the binomial method "backwardation" (5.4). This is a risk-neutral valuation (see Section 5.4.2), since we use the assumption of $m = r$ where r is the risk-free interest rate, in Equation 5.4.7. It should be noted that it is best if m and r are close, which, in practice, is usually true. The major factor influencing the validity of our backwardation-based American put valuation is the value of σ.

8.5.5 Sample Calculation for Ergodic Diffusion Approximation

Suppose that we have a two-state GMRP to which we will apply the Ergodic Diffusion Approximation. Suppose the Markov transition matrix is

$$P = \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix}$$

We have the returns (in vector form) for each state as follows:

$$\rho = \begin{bmatrix} 0.02 & -0.02 \end{bmatrix}$$

We have exponentially distributed waiting times between jumps, i.e. Poisson distributed jumps, with intensity parameters as follows:

$$\lambda(x_0) = 8, \quad \lambda(x_1) = 10$$

Here we have the limiting probabilities p_0 and p_1 satisfying

$$\begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} = \begin{bmatrix} p_0 & p_1 \end{bmatrix}$$

Hence we have $p_0 = 0.5$ and $p_1 = 0.5$. We define

$$\Pi = \begin{bmatrix} p_0 & p_1 \\ p_0 & p_1 \end{bmatrix}$$

The balance condition is indeed fulfilled, as

$$\begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} \begin{bmatrix} \rho_0 \\ \rho_1 \end{bmatrix} = 0$$

We have for each state x

$$\bar{m}(x) = \int_0^\infty e^{-\lambda(x)t} dt = \frac{1}{\lambda(x)}, \quad m(x) = \int_X p(dx) \frac{1}{\lambda(x)}$$

Following the algorithms for the ergodic approximation we have:

$$\hat{\rho}_2 = \int_X p(dx) \int_X P(x,dy)\rho^2(y)/m = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} \begin{bmatrix} \rho_0^2 \\ \rho_1^2 \end{bmatrix} / \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} \bar{m}(x_0) \\ \bar{m}(x_1) \end{bmatrix}$$

$$= 3.556 \times 10^{-3}$$

We need a special computation for the potential operator R_0:

$$\sum_{n=0}^{\infty} (P^n - \Pi)$$

This can be computed by diagonalization of the matrix P, where E is the matrix of eigenvectors and D is a diagonal matrix of eigenvalues.

$$P = EDE^{-1}$$

Thus the exponentiation formula for P is

$$P^n = ED^nE^{-1}$$

Since P is a Markov matrix, it will have an eigenvalue equal to 1. The other eigenvalues will be smaller in magnitude than 1, and these will decay to 0 in the exponentiation D^n. Since the limit will be the matrix Π, we have that

$$\sum_{n=0}^{\infty} (P^n - \Pi)$$

converges as $n \to \infty$. We can accomplish this computationally by taking enough terms of the sum above and stopping when the sum of squares of the entries of the next matrix term in the sum falls below some pre-specified threshold δ. See Appendix A for a detailed example.

Now we use this sum in the following portion of the algorithms for the ergodic approximation:

$$\sigma_\rho^2 = \int_X p(dx) \left(\frac{1}{2} \int_X P(x,dy)\rho^2(y) + \sum_{n=0}^{\infty} (P^n - \Pi) \int_X P(x,dy)\rho(y) \int_X P(x,dy)\rho(y) \right) / m$$

$$= \frac{\begin{bmatrix} p_0 & p_1 \end{bmatrix} \left(\frac{1}{2} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} \begin{bmatrix} \rho_0^2 \\ \rho_1^2 \end{bmatrix} + \sum_{n=0}^{\infty} \left(\begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}^n - \begin{bmatrix} p_0 & p_1 \\ p_0 & p_1 \end{bmatrix} \right) \begin{bmatrix} (p_{00}\rho_0 + p_{01}\rho_1)^2 \\ (p_{10}\rho_0 + p_{11}\rho_1)^2 \end{bmatrix} \right)}{\begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} \bar{m}(x_0) \\ \bar{m}(x_1) \end{bmatrix}}$$

$$= 1.778 \times 10^{-3}$$

From the above data we have

$$\hat{S}(t) = S_0 e^{-\frac{1}{2}\hat{\rho}_2 + \sigma_\rho W(t)}$$

where by completing all calculation yields that

$$\hat{\rho}_2 = 3.556 \times 10^{-3}$$
$$\sigma_\rho^2 = 1.778 \times 10^{-3}$$

Suppose that

$$S_0 = 20 \text{ (starting stock price)}$$
$$E = 18 \text{ (strike price)}$$
$$r = 0.01 \text{ (risk-free rate)}$$
$$T = 4 \text{ (time to expiry)}$$

By algorithms presented in Section (4), we have

$$d_1 = 1.7659$$
$$d_2 = 1.6816$$

So the value of a European call will be (after calculations)

$$C^E = 2.732833$$

The value of a European put will be (after calculations)

$$P^E = 0.961296$$

The value of an American call will be (it is the same price as a European call)

$$C^A = 2.732833$$

The value of an American put will be (using the backwardation method with $M = 100$ steps as in Section 5.4)

$$P^A = 0.030138$$

8.5.6 Guide to MATLAB code ErgodicDiffusionApprox.m

The MATLAB code ErgodicDiffusionApprox.m is designed to graph and calculate option prices (European and American) for a GMRP-modulated asset with exponentially distributed waiting times for jumps (Poisson waiting times). It takes as input the transition matrix, the ρ values, and the λ in each case for exponential distributions in each state. If the option to graph paths is selected, it outputs sample asset paths. If the option to price options is selected, it prices American or European options according to the algorithm described in the section (8.5.1). The code can be found in Appendix B.6.

8.5.7 Sample run of MATLAB code ErgodicDiffusionApprox.m

We run the code on the same data as given in Section (8.5.5). The results are identical to those calculated in Section (8.5.5). Here is the output:

```
 The Balance Condition is fulfilled.
value is equal to 0.000000

 Critical Parameters:
```

```
 Rhohat2 = 0.003556
 Sigma_rho_2 squared = 0.001778



 European option valuation
 Please enter necessary parameters below.
 Enter the current stock price:   20
 Input the exercise price:   18
 Input the interest rate r:   .01
 Enter tau (time to expiry, T-t):   4
d1 =  1.7659
d2 =  1.6816

 European option information is below:
 European Call value: 2.732833
 Delta of European Call: 0.961296
 European Put value: 0.027043
 Delta of European Put: -0.038704



 American option valuation via the Binomial Method () and Black-Scholes
 Please enter necessary parameters below.
 Enter the current stock price:   20
 Input the exercise price:   18
 Input the interest rate r:   .01
 Enter the time to expiry:   4
Enter M, the number of steps in the binomial calculation:   100

 The value of an American put is:   0.030138
```

```
The value of an American call is:   2.7328
```

8.5.8 Sample Calculation for Merged Double Averaging Approximation

Suppose we have the Markov transition matrix

$$P = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} 0.49 & 0.49 & 0.01 & 0.01 \\ 0.70 & 0.28 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.70 & 0.28 \\ 0.01 & 0.01 & 0.49 & 0.49 \end{bmatrix}$$

Stationary probability calculations, as in Section (8.5.5), yields

$$\pi = \begin{bmatrix} 0.546448 \\ 0.384822 \\ 0.030757 \\ 0.037821 \end{bmatrix}^T$$

We have the returns as follows:

$$\rho = \begin{bmatrix} \rho_0 \\ \rho_1 \\ \rho_2 \\ \rho_3 \end{bmatrix} = \begin{bmatrix} -0.04032 \\ 0.05768 \\ -0.03571 \\ 0.06629 \end{bmatrix}$$

We have exponentially distributed waiting times between jumps, i.e. Poisson distributed jumps, with intensity parameters as follows:

$$\lambda(x_0) = 8 \;\; \lambda(x_1) = 10, \;\; \lambda(x_2) = 12, \;\; \lambda(x_3) = 10$$

We will merge this Markov chain to the merged state space (X_0, X_1) where state X_0 is the merged

states 0 and 1 of the original Markov chain, and state X_1 is the merged states 2 and 3 of the original Markov chain. Here we have the Markov matrix for state X_0 as

$$P_0 = \begin{bmatrix} p^*_{00} & p^*_{01} \\ p^*_{10} & p^*_{11} \end{bmatrix} = \begin{bmatrix} \frac{0.49}{0.49+0.49} & \frac{0.49}{0.49+0.49} \\ \frac{0.7}{0.7+0.28} & \frac{0.28}{0.28+0.7} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.7143 & 0.2857 \end{bmatrix}$$

with limiting probabilities

$$\pi_{X_0} = \begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} = \begin{bmatrix} 0.5882 & 0.4118 \end{bmatrix}$$

The balance condition is met, as

$$\begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} \begin{bmatrix} p^*_{00} & p^*_{01} \\ p^*_{10} & p^*_{11} \end{bmatrix} \begin{bmatrix} \rho(X_0(0)) \\ \rho(X_0(1)) \end{bmatrix} = 0.000033$$

which we deem to be close enough to 0.

and we have the Markov matrix for state X_1 as

$$P_1 = \begin{bmatrix} p^*_{22} & p^*_{23} \\ p^*_{32} & p^*_{33} \end{bmatrix} = \begin{bmatrix} \frac{0.7}{0.7+0.28} & \frac{0.28}{0.28+0.7} \\ \frac{0.49}{0.49+0.49} & \frac{0.49}{0.49+0.49} \end{bmatrix} = \begin{bmatrix} 0.7143 & 0.2857 \\ 0.5 & 0.5 \end{bmatrix}$$

with limiting probabilities

$$\pi_{X_1} = \begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix} = \begin{bmatrix} 0.6364 & 0.3636 \end{bmatrix}$$

The balance condition is met, as

$$\begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix} \begin{bmatrix} p^*_{22} & p^*_{23} \\ p^*_{32} & p^*_{33} \end{bmatrix} \begin{bmatrix} \rho(X_2(0)) \\ \rho(X_3(1)) \end{bmatrix} = 0.001381$$

which we deem to be close enough to 0.

Furthermore we have

$$m(X_0) = 0.5882 \cdot \frac{1}{8} + 0.4118 \cdot \frac{1}{10} = 0.1147$$

and

$$m(X_0) = 0.6364 \cdot \frac{1}{12} + 0.3636 \cdot \frac{1}{10} = 0.08939$$

By algorithms of ergodic averaging in each state, and by our results for the diffusion approximation we have that

$$\hat{\rho}_2(X_0) = \frac{\int_{X_0} \pi_{X_0}(dx) \int_{X_0} P(x,dy)\rho^2(y)}{m(X_0)}$$

$$= \begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} \begin{bmatrix} p^*_{00} & p^*_{01} \\ p^*_{10} & p^*_{11} \end{bmatrix} \begin{bmatrix} \rho(X_0(0))^2 \\ \rho(X_0(1))^2 \end{bmatrix} / \left(\begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} \begin{bmatrix} \bar{m}(X_0(0)) \\ \bar{m}(X_0(1)) \end{bmatrix} \right)$$

$$= 0.020280$$

Similarly

$$\hat{\rho}_2(X_1) = \frac{\int_{X_1} \pi_{X_1}(dx) \int_{X_1} P(x,dy)\rho^2(y)}{m(X_1)}$$

$$= \begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix} \begin{bmatrix} p^*_{00} & p^*_{01} \\ p^*_{10} & p^*_{11} \end{bmatrix} \begin{bmatrix} \rho(X_1(0))^2 \\ \rho(X_1(1))^2 \end{bmatrix} / \left(\begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix} \begin{bmatrix} \bar{m}(X_1(0)) \\ \bar{m}(X_1(1)) \end{bmatrix} \right)$$

$$= 0.026953$$

Writing

$$\Pi_0 = \begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \\ \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} \text{ and } \Pi_1 = \begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \\ \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix}$$

we also have

$$\hat{\sigma}_\rho^2(X_0) = \int_{X_0} \pi_{X_0}(dx)\left[\frac{1}{2}\int_{X_0} P(x,dy)\rho^2(y) + \sum_{n=0}^{\infty}(P_0^n - \Pi_0)\left(\int_{X_0} P(x,dy)\rho(y)\right)^2\right]/m_{X_0}$$

$$= \frac{\left[\begin{array}{cc} \pi_{X_0}(0) & \pi_{X_0}(1) \end{array}\right]\left[\frac{1}{2}\left[\begin{array}{cc} p_{00}^* & p_{01}^* \\ p_{10}^* & p_{11}^* \end{array}\right]\left[\begin{array}{c} \rho_0^2 \\ \rho_1^2 \end{array}\right] + \left(\sum_{n=0}^{\infty}(P_0^n - \Pi_0)\right)\left[\begin{array}{c} (p_{00}^*\rho_0 + p_{01}^*\rho_1)^2 \\ (p_{10}^*\rho_0 + p_{11}^*\rho_1)^2 \end{array}\right]\right]}{\left[\begin{array}{cc} \pi_{X_0}(0) & \pi_{X_0}(1) \end{array}\right]\left[\begin{array}{c} m(X_0(1)) \\ m(X_0(0)) \end{array}\right]}$$

$$= 0.010140$$

and

$$\hat{\sigma}_\rho^2(X_1) = \int_{X_1} \pi_{X_1}(dx)\left[\frac{1}{2}\int_{X_1} P(x,dy)\rho^2(y) + \sum_{n=0}^{\infty}(P_1^n - \Pi_1)\left(\int_{X_1} P(x,dy)\rho(y)\right)^2\right]/m_{X_1}$$

$$= \frac{\left[\begin{array}{cc} \pi_{X_1}(0) & \pi_{X_1}(1) \end{array}\right]\left[\frac{1}{2}\left[\begin{array}{cc} p_{22}^* & p_{23}^* \\ p_{32}^* & p_{33}^* \end{array}\right]\left[\begin{array}{c} \rho_2^2 \\ \rho_3^2 \end{array}\right] + \left(\sum_{n=0}^{\infty}(P_1^n - \Pi_1)\right)\left[\begin{array}{c} (p_{22}^*\rho_2 + p_{23}^*\rho_3)^2 \\ (p_{32}^*\rho_2 + p_{33}^*\rho_3)^2 \end{array}\right]\right]}{\left[\begin{array}{cc} \pi_{X_1}(0) & \pi_{X_1}(1) \end{array}\right]\left[\begin{array}{c} m(X_1(1)) \\ m(X_1(0)) \end{array}\right]}$$

$$= 0.013477$$

Using algorithms of double averaging, we have

$$\breve{\rho}_2 = \frac{1}{2}(\hat{\rho}_2(X_0) + \hat{\rho}_2(X_1)) = \frac{1}{2}(0.020280 + 0.026953) = 0.023617$$

and

$$\breve{\sigma}_\rho^2 = \frac{1}{2}(\hat{\sigma}_\rho^2(X_0) + \hat{\sigma}_\rho^2(X_1)) = \frac{1}{2}(0.010140 + 0.013477) = 0.011808$$

From the above data we have

$$\hat{S}(t) = S_0 e^{-\frac{1}{2}\breve{\rho}_2 t + \breve{\sigma}_\rho W(t)}$$

where

$$\breve{\rho}_2 = 0.023617$$
$$\breve{\sigma}_\rho^2 = 0.011808$$

Suppose that

$$S_0 = 20 \text{ (starting stock price)}$$
$$E = 18 \text{ (strike price)}$$
$$r = 0.01 \text{ (risk-free rate)}$$
$$T = 4 \text{ (time to expiry)}$$

By algorithms presented in Section (4), we have

$$d_1 = 0.77751$$
$$d_2 = 0.56018$$

So the value of a European call will be (after calculations)

$$C^E = 0.56018$$

The value of a European put will be (after calculations)

$$P^E = 0.606600$$

The value of an American call will be (it is the same price as a European call)

$$C^A = 0.56018$$

The value of an American put will be (using the backwardation method with $M = 100$ steps as in Section 5.4)

$$P^A = 3.3124$$

8.5.9 Guide to MATLAB code DoubleAveragedDiffusionApprox.m

The MATLAB code DoubleAveragedDiffusionApprox.m is designed to graph and calculate option prices (European and American) for a GMRP-modulated asset with exponentially distributed waiting times for jumps (Poisson waiting times). It takes as input the transition matrix, the ρ values, and the λ in each case for exponential distributions in each state. Also, it takes information on how to divide the transition matrix. If the option to graph paths is selected, it outputs sample asset paths. If the option to price options is selected, it prices American or European options using the Double Averaging procedure (see Section 8.5.3) according to the algorithm described in the section (5.4). The code can be found in Appendix B.7.

8.5.10 Sample run of MATLAB code DoubleAveragedDiffusionApprox.m

We run the code on the same data as given in Section (8.5.8). The results are identical to those calculated in Section (8.5.8). Here is the output:

```
 The Balance Condition is NOT fulfilled for state no 1 of the embedded process.
value is equal to 0.000033

 The Balance Condition is NOT fulfilled for state no 2 of the embedded process.
value is equal to 0.001381

 Below are displayed the computed critical values for each state of the embedded Markov chain:
 Embedded State #; rhohat; sigmahat squared
 1 0.020280 0.010140
 2 0.026953 0.013477

 Below is the averaged values for the Markov chain:
 rhohat averaged; sigmahat squared averaged
   0.023617   0.011808
```

```
 European option valuation
 Please enter necessary parameters below.
 Enter the current stock price:   20
 Input the exercise price:   18
 Input the interest rate r:   .01
 Enter tau (time to expiry, T-t):   4
d1 =  0.77751
d2 =  0.56018

 European option information is below:
 European Call value: 3.312390
 Delta of European Call: 0.781570
 European Put value: 0.606600
 Delta of European Put: -0.218430



 American option valuation via the Binomial Method () and Black-Scholes
 Please enter necessary parameters below.
 Enter the current stock price:   20
 Input the exercise price:   18
 Input the interest rate r:   .01
 Enter the time to expiry:   4
Enter M, the number of steps in the binomial calculation:   100

 The value of an American put is:   0.63620

 The value of an American call is:   3.3124
```

8.6 Normal Deviations Approximation of GMRP

8.6.1 Ergodic Normal Deviations [22]

We define $\hat{\rho}$ for the states $k = 1, 2, ..., n$ of the GMRP's Markov chain as follows:

$$\hat{\rho}(k) = \frac{\int_{X_k} p_k(dx) \int_{X_k} P(x,dy)\rho(y)}{m(k)} \tag{8.6.1}$$

Assuming $\hat{\rho} \neq 0$, let

$$\omega_T = \sqrt{T}(\alpha_T(t) - \hat{\rho}t) \tag{8.6.2}$$

Now we define $S_T(t)$ as follows derived from the standard GMRP model (1.2.3):

$$S_T(t) := S_0 \prod_{k=1}^{\nu(tT)} (1 + T^{-1}\rho(x_k)) \tag{8.6.3}$$

where

$$\alpha_T(t) = \ln S_T(t) \approx T^{-1} \sum_{k=1}^{\nu(tT)} \rho(x_k)$$

The process $\omega_T(t)$ defines deviations of the initial model $\alpha_T(t)$ in the scale of time tT from the averaged model $\hat{\rho}(t)$. Under large values of T, we have that the $\omega_T(t)$ obtains the properties of a Wiener process. We can rewrite $\omega_T(t)$ in the following form:

$$\omega_T(t) = T^{-\frac{1}{2}} \sum_{k=1}^{\nu(tT)} [\rho(x_k) - m\hat{\rho}] - \hat{\rho}T^{-\frac{1}{2}}(tT - m\nu(tT)) \tag{8.6.4}$$

The second term of 8.6.4 has a limit of 0 as $T \to \infty$. For the first term, consider that the function $P\rho - m\hat{\rho}$, where $Pf(x) = \int_X f(y)P(x,dy)$, satisfies the balance condition for the measure $p(dx)$, since $\int_X p(dx)[P\rho - m\hat{\rho}] = 0$. Now we apply diffusion approximation algorithms to this first term of (8.6.4). We consider the function $P\rho - m\hat{\rho}$ instead of $P\rho$ yielding the following result

$$\lim_{T\to\infty} \omega_T(t) = \hat{\sigma}\hat{\omega}(t) \tag{8.6.5}$$

where $\hat{\omega}(t)$ is a standard Wiener process with diffusion coefficient given by the following, resulting from existing calculation algorithms for normal deviations, (and where R_0 is a potential of $(x_n)_{n\in\mathbb{Z}_+}$):

$$\hat{\sigma}^2 := \int_X p(d)[(P\rho - m\hat{\rho})R_0(P\rho - m\hat{\rho}) + \frac{1}{2}(P\rho - m\hat{\rho})^2]/m \tag{8.6.6}$$

Writing $\alpha_T(t) \approx \hat{\rho}t + T^{-\frac{1}{2}}\hat{\sigma}\hat{\omega}(t)$, we have the approximation (in convergence in distribution)

$$\ln\frac{S_T(t)}{S_0} = \ln\frac{\hat{S}(t)}{S_0} = \hat{\rho}t + T^{-\frac{1}{2}}\hat{\sigma}\hat{\omega}(t) \tag{8.6.7}$$

where we have denoted the limit of $S_T(t)$ as follows:

$$\lim_{T\to\infty} S_T(t) = \hat{S}(t) \tag{8.6.8}$$

Thus the normal deviated GMRP has a GBM approximation with the following form:

$$\lim_{T\to\infty} S_T(t) = \hat{S}(t) = S_0 e^{\hat{\rho}t + T^{-\frac{1}{2}}\hat{\sigma}\hat{\omega}(t)} \tag{8.6.9}$$

Writing this as a stochastic differential equation, we have that

$$\frac{d\hat{S}(t)}{\hat{S}(t)} = (\hat{\rho} + \frac{1}{2}T^{-1}\hat{\sigma}^2)dt + T^{-\frac{1}{2}}\hat{\sigma}d\hat{\omega}(t) \tag{8.6.10}$$

8.6.2 Merged Normal Deviations [22]

Suppose that $(x_n)_{n\in\mathbb{Z}_+}$ is the GMRP's Markov chain, and p_k is the transitional probability for the ergodic component $m(k) = \int_{X_k} p_k(dx)m(x)$ (where $m(x) = \int_0^\infty (1 - G_x(t))dt$ and $G_x(t) = P\{\theta_{n+1} \leq t | x_n = x\}$), and furthermore, conditions for the reducibility of X are fulfilled. If the balance condition is not fulfilled, i.e. for some k,

$$\hat{\rho}(k) = \frac{\int_{X_k} p_k(dx)\int_{X_k} P(x,dy)\rho(y)}{m(k)} \neq 0 \tag{8.6.11}$$

Now consider the normal deviated process

$$\tilde{\omega}_T(t) = \sqrt{T}(\alpha_T(t) - \tilde{\rho}(t)) \tag{8.6.12}$$

where we have used

$$\alpha_T(t) := T^{-1} \sum_{k=1}^{\nu(tT)} \rho(x_k)$$

and

$$\tilde{\rho}(t) := \int_0^t \hat{\rho}(\hat{x}(s))ds$$

and $\hat{\rho}(k)$ is defined as in Equation (8.6.1) for the states $k = 1, 2, ..., n$ of the GMRP's embedded Markov chain.

Using the fact that $\tilde{\omega}_T(t)$ is a stochastic Ito integral for large T values we have the following result:

$$\tilde{\omega}_T(t) \approx \int_0^t \hat{\sigma}(\hat{x}(s))d\hat{\omega}(s) \tag{8.6.13}$$

where for $k = 1, 2, ..., r$

$$\tilde{\sigma}(k) := \int_{X_k} p_k(dx)[(P\rho - m(k)\hat{\rho}(k))R_0(P\rho - m(k)\hat{\rho}(k)) + \frac{1}{2}(P\rho - m(k)\hat{\rho}(k))^2]/m(k) \tag{8.6.14}$$

Thus we have the following approximation:

$$\alpha_T(t) \approx \tilde{\rho}(t) + T^{-\frac{1}{2}} \int_0^t \tilde{\sigma}(\hat{x}(s))d\omega(s) \tag{8.6.15}$$

We use the following notation for the limit of $S_T(t)$:

$$\lim_{T\to\infty} S_T(t) = \tilde{S}(t) \tag{8.6.16}$$

Thus we have

$$\lim_{T\to\infty} \ln \frac{S_T(t)}{S_0} = \ln \frac{\tilde{S}(t)}{S_0} = \tilde{\rho}(t) + T^{-\frac{1}{2}} \int_0^t \tilde{\sigma}(\hat{x}(s))d\omega(s) \tag{8.6.17}$$

Hence, in summary, for the case of normal deviated GMRP we have the following GBM approximation:

$$\lim_{T\to\infty} S_T(t) = \tilde{S}(t) = S_0 e^{\tilde{\rho}(t)+T^{-\frac{1}{2}}\int_0^t \tilde{\sigma}(\hat{x}(s))d\omega(s)} \tag{8.6.18}$$

or, written in stochastic differential equation form:

$$\lim_{T\to\infty} \frac{dS_T(t)}{S_T(t)} = \frac{d\tilde{S}(t)}{\tilde{S}(t)}(\tilde{\rho}(\hat{x}(t)) + \frac{1}{2}T^{-1}\tilde{\sigma}^2(\hat{x}(t)))dt + T^{-\frac{1}{2}}\tilde{\sigma}(\hat{x}(t))d\omega(t) \tag{8.6.19}$$

8.6.3 Normal Deviations in the Case of Double Averaging [22]

Suppose that the merged phase space $\hat{X}$ of the merged Markov process $\hat{x}(t)$ consists of one ergodic class with stationary distribution $(\hat{p}(k))_{k=1,2,...,N}$. Then using double averaging algorithms, it can be shown that

$$\alpha_T(t) \approx \breve{\rho}t \tag{8.6.20}$$

where

$$\breve{\rho} := \sum_{k=1}^{\nu(tT)} \hat{p}_k \hat{\rho}(k)$$

and $\hat{\rho}(k)$ is defined as in Equation (8.6.1) for the states $k = 1,2,...,n$ of the GMRP's embedded Markov chain. Thus we have the following result:

$$\lim_{T\to\infty} \ln \frac{S_t^T}{S_0} = \breve{\rho}t \tag{8.6.21}$$

The above is is equivalent to $S_t \approx S_0 e^{\breve{\rho}t}$.

Now let us suppose that $\breve{\rho} \neq 0$, and let us consider the normal deviated process

$$\breve{\omega}_T(t) = \sqrt{T}(\alpha_T(t) - \breve{\rho}t) \tag{8.6.22}$$

Using the fact that $\breve{\omega}_T(t)$ is a Wiener process with diffusion coefficient $\breve{\sigma}^2$ for large T, we have

$$\breve{\sigma}^2 := \sum_{k=1}^{r} \hat{p}_k \tilde{\sigma}^2(k) \tag{8.6.23}$$

where for $k = 1, 2, ..., r$

$$\tilde{\sigma}^2(k) := \int_{X_k} p_k(dx)[(P\rho - m(k)\rho(k))R_0(P\rho - m(k)\rho(k)) + \frac{1}{2}(P\rho - m(k)\rho(k))^2]/m(k)$$

Hence

$$\breve{\omega}_T(t) \approx \breve{\sigma}^2 \omega(t) \tag{8.6.24}$$

where $\omega(t)$ is a standard Wiener process. Hence double approximation of $\alpha_T(t)$ can be expressed as:

$$\alpha_T(t) \approx \breve{\rho} t + T^{-\frac{1}{2}} \breve{\sigma}^2 \omega(t) \tag{8.6.25}$$

So, using the notation

$$\lim_{T\to\infty} S_T(t) = \breve{S}(t) \tag{8.6.26}$$

we have that

$$\lim_{T\to\infty} \ln \frac{S_T(t)}{S_0} = \ln \frac{\breve{S}(t)}{S_0} = \breve{\rho} t + T^{-\frac{1}{2}} \breve{\sigma}^2 \omega(t) \tag{8.6.27}$$

This can be summarized explicitly to show our approximation of our GMRP with a GBM as follows:

$$\lim_{T\to\infty} S_T(t) = \breve{S}(t) = S_0 e^{\breve{\rho} t + T^{-\frac{1}{2}} \breve{\sigma}^2 \omega(t)} \tag{8.6.28}$$

or, equivalently, written in stochastic differential equation form,

$$\lim_{T\to\infty} \frac{dS_T(t)}{S_T(t)} = \frac{d\breve{S}(t)}{\breve{S}(t)} = (\breve{\rho} + \frac{1}{2} T^{-1} \breve{\sigma}^2) dt + T^{-\frac{1}{2}} \breve{\sigma}^2 d\omega(t) \tag{8.6.29}$$

8.6.4 Option Pricing Under Ergodic and Double Averaged Normal Deviations GMRP Approximation

8.6.4.1 European Option Valuation

For the cases of the Ergodic Normal Deviations (8.6.1) and Merged Normal Deviations with Double Averaged Normal Deviations (8.6.3), since in both cases we have approximated the GMRP in the form of a stochastic differential equation for a GBM, we can directly use the Black-Scholes formulas to find the price for European calls and puts, after appropriate considerations for a risk-neutral valuation.

Since we have the result from (8.6.10) that

$$\lim_{T\to\infty} \frac{dS_T(t)}{S_T(t)} = \frac{d\hat{S}(t)}{\hat{S}(t)}(\hat{\rho} + \frac{1}{2}T^{-1}\hat{\sigma}^2)dt + T^{-\frac{1}{2}}\hat{\sigma}d\hat{\omega}(t)$$

and the analogous expression for Double Averaged Diffusion in (8.6.3), we can calculate the risk-neutral probability measure P^* using Girsanov's theorem and the Radon-Nikodym derivative [19] as follows:

$$\frac{dP^*}{dP} = \exp\left\{ -\theta\omega(t) - \frac{1}{2}\theta^2 t \right\} \tag{8.6.30}$$

where we have as the market price of risk

$$\theta = \frac{(\hat{\rho} + \frac{1}{2}T^{-1}\hat{\sigma}^2) - r}{T^{-\frac{1}{2}}\hat{\sigma}} \tag{8.6.31}$$

By established results, the discounted process

$$e^{-rt}S_T(t) \tag{8.6.32}$$

is a P^*- martingale. Furthermore, the process

$$\omega^*(t) = \omega(t) + \theta t \tag{8.6.33}$$

is a Brownian motion (with drift). Thus, in the risk-neutral world, the process $S_T(t)$ has the following representation:

$$\frac{d\hat{S}(t)}{\hat{S}(t)} = rdt + T^{-\frac{1}{2}}\hat{\sigma}d\omega^*(t) \tag{8.6.34}$$

and we may directly substitute the value for $T^{-\frac{1}{2}}\hat{\sigma}$ into our Black-Scholes formulas for d_1 and d_2, and proceed with European call and European put valuation. See parts (4.2.6), (4.2.5), and (4.2.8). This solves the valuation problem of valuation of both European calls and European puts.

8.6.4.2 American Option Valuation

We can use the risk-neutral Black-Scholes valuation for an American call, since an American call has the same value as a European call). For American puts, we can employ the method of the binomial method "backwardation" (5.4). This is a risk-neutral valuation (see Section 5.4.2), since we use the assumption of $m = r$ where r is the risk-free interest rate, in Equation 5.4.7. It should be noted that it is best if m and r are close, which, in practice, is usually true. The major factor influencing the validity of our backwardation-based American put valuation is the value of σ.

8.6.5 Sample Calculation for Ergodic Normal Deviations Approximation

Suppose that we have a two-state GMRP to which we will apply the Ergodic Diffusion Approximation.

Suppose the Markov transition matrix is

$$P = \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix}$$

We have the returns (in vector form) for each state as follows:

$$\rho = \begin{bmatrix} 0.025 & -0.03 \end{bmatrix}$$

We have exponentially distributed waiting times between jumps, i.e. Poisson distributed jumps, with intensity parameters as follows:

$$\lambda(x_0) = 8 \quad \lambda(x_1) = 10$$

And we have

$$T = 10$$

Here we have the limiting probabilities p_0 and p_1 satisfying

$$\begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} = \begin{bmatrix} p_0 & p_1 \end{bmatrix}$$

Hence we have $p_0 = 0.5$ and $p_1 = 0.5$. We define

$$\Pi = \begin{bmatrix} p_0 & p_1 \\ p_0 & p_1 \end{bmatrix}$$

We have for each state x

$$\bar{m}(x) = \int_0^\infty e^{-\lambda(x)t} dt = \frac{1}{\lambda(x)} \quad m(x) = \int_X p(dx) \frac{1}{\lambda(x)}$$

Following the algorithms for the ergodic approximation:

$$\hat{\rho} = \frac{\int_X p(dx) \int_X P(x,dy)\rho(y)}{m} = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} \begin{bmatrix} \rho_0 \\ \rho_1 \end{bmatrix} / \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} \bar{m}(x_0) \\ \bar{m}(x_1) \end{bmatrix}$$

$$= 0.089284$$

We need a special computation for the potential operator R_0:

$$\sum_{n=0}^{\infty} (P^n - \Pi)$$

See Appendix A for a detailed example.

Now we use this sum in the following portion of the algorithms for the ergodic approximation:

$$\hat{\sigma}^2 = \int_X p(dx)\left((P\rho - m\hat{\rho})R_0(P\rho - m\hat{\rho}) + \frac{1}{2}(P\rho - m\hat{\rho})^2\right)/m$$

$$= \int_X p(dx)\left[\sum_{n=0}^{\infty}(P^n - \Pi)\left(\int_X P(x,dy)\rho(y) - \int_X p(dx)\int_X P(x,dy)\rho(y)\right)\right]^2$$

$$+\frac{1}{2}\left(\int_X P(x,dy)\rho(y) - \int_X p(dx)\int_X P(x,dy)\rho(y)\right)^2\Bigg]/m$$

$$= \begin{bmatrix} p_0 & p_1 \end{bmatrix}\left(\sum_{n=0}^{\infty}(P^n - \Pi)\begin{bmatrix} (p_{00}\rho_0 + p_{01}\rho_1 - m\hat{\rho})^2 \\ (p_{10}\rho_0 + p_{11}\rho_1 - m\hat{\rho})^2 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} (p_{00}\rho_0 + p_{01}\rho_1 - m\hat{\rho})^2 \\ (p_{10}\rho_0 + p_{11}\rho_1 - m\hat{\rho})^2 \end{bmatrix}\right)/m$$

$$= 0.006554$$

From the above data we have

$$\hat{S}(t) = S_0 e^{\hat{\rho} + T^{-\frac{1}{2}}\hat{\sigma}W(t)}$$

where

$$\hat{\rho} = 0.089284$$
$$\hat{\sigma}^2 = 0.006554$$

Suppose that

$$S_0 = 20 \text{ (starting stock price)}$$
$$E = 18 \text{ (strike price)}$$
$$r = 0.01 \text{ (risk-free rate)}$$
$$T = 4 \text{ (time to expiry)}$$

By algorithms presented in Section (4), we have

$$d_1 = 2.8647$$
$$d_2 = 2.8135$$

So the value of a European call will be (after calculations)

$$C^E = 2.706425$$

The value of a European put will be (after calculations)

$$P^E = 0.000635$$

The value of an American call will be (it is the same price as a European call)

$$C^A = 2.7064$$

The value of an American put will be (using the backwardation method with $M = 100$ steps as in Section 5.4)

$$P^A = 6.8718e-004$$

8.6.6 Guide to MATLAB code ErgodicNormalDeviation.m

The MATLAB code ErgodicNormalDeviation.m is designed to graph and calculate option prices (European and American) for a GMRP-modulated asset with exponentially distributed waiting times for jumps (Poisson waiting times). It takes as input the transition matrix, the ρ values, and the λ in each case for exponential distributions in each state. It also takes as input the value of T. If the option to graph paths is selected, it outputs sample asset paths. If the option to price options is selected, it prices American or European options according to the algorithm described in the section (8.5.4) for the case of an ergodic GMRP.

The code may be found in Appendix B.8.

8.6.7 Sample run of MATLAB code ErgodicNormalDeviation.m

We run the code on the same data as given in Section (8.6.5). The results are identical to those calculated in Section (8.6.5). Here is the output:

```
%%%%% Critical Parameters: %%%%%
 Rhohat = 0.089284
 Sigma_hat squared = 0.006554



d1 =  2.8647
d2 =  2.8135

 European option information is below:
 European Call value: 2.706425
 Delta of European Call: 0.997913
 European Put value: 0.000635
 Delta of European Put: -0.002087



 American option valuation via the Binomial Method () and Black-Scholes
 Please enter necessary parameters below.
 Enter the current stock price:   20
 Input the exercise price:   18
 Input the interest rate r:   0.01
 Enter the time to expiry:   4
Enter M, the number of steps in the binomial calculation:   100

 The value of an American put is:  6.8718e-004

 The value of an American call is:   2.7064
```

8.6.8 Sample Calculation for Normal Deviations Approximation

Suppose we have the Markov transition matrix

$$P = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} 0.49 & 0.49 & 0.01 & 0.01 \\ 0.70 & 0.28 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.70 & 0.28 \\ 0.01 & 0.01 & 0.49 & 0.49 \end{bmatrix}$$

Stationary probability calculations, as in Section (8.5.5), yields

$$\pi = \begin{bmatrix} 0.546448 \\ 0.384822 \\ 0.030757 \\ 0.037821 \end{bmatrix}^T$$

We have the returns as follows:

$$\rho = \begin{bmatrix} \rho_0 \\ \rho_1 \\ \rho_2 \\ \rho_3 \end{bmatrix} = \begin{bmatrix} 0.03 \\ -0.02 \\ 0.02 \\ -0.02 \end{bmatrix}$$

We have exponentially distributed waiting times between jumps, i.e. Poisson distributed jumps, with intensity parameters as follows:

$$\lambda(x_0) = 8 \ \lambda(x_1) = 10 \ \lambda(x_2) = 12 \ \lambda(x_3) = 10$$

We will merge this Markov chain to the merged state space (X_0, X_1) where state X_0 is the merged states 0 and 1 of the original Markov chain, and state X_1 is the merged states 2 and 3 of the original Markov chain.

We have

$$m_k = \int_{X_k} \pi_k(dx) m(x) \text{ for } k = 0,1,2,3$$

Here we have the Markov matrix for state X_0 as

$$P_0 = \begin{bmatrix} p^*_{00} & p^*_{01} \\ p^*_{10} & p^*_{11} \end{bmatrix} = \begin{bmatrix} \frac{0.49}{0.49+0.49} & \frac{0.49}{0.49+0.49} \\ \frac{0.7}{0.7+0.28} & \frac{0.28}{0.28+0.7} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.7143 & 0.2857 \end{bmatrix}$$

with limiting probabilities

$$\pi_{X_0} = \begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} = \begin{bmatrix} 0.5882 & 0.4118 \end{bmatrix}$$

and we have the Markov matrix for state X_1 as

$$P_1 = \begin{bmatrix} p^*_{22} & p^*_{23} \\ p^*_{32} & p^*_{33} \end{bmatrix} = \begin{bmatrix} \frac{0.7}{0.7+0.28} & \frac{0.28}{0.28+0.7} \\ \frac{0.49}{0.49+0.49} & \frac{0.49}{0.49+0.49} \end{bmatrix} = \begin{bmatrix} 0.7143 & 0.2857 \\ 0.5 & 0.5 \end{bmatrix}$$

with limiting probabilities

$$\pi_{X_1} = \begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix} = \begin{bmatrix} 0.6364 & 0.3636 \end{bmatrix}$$

We have that

$$\hat{\rho}_{X_0} = \begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} \begin{bmatrix} p^*_{00} & p^*_{01} \\ p^*_{10} & p^*_{11} \end{bmatrix} \begin{bmatrix} \rho_0 \\ \rho_1 \end{bmatrix} / \left(\begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} \begin{bmatrix} \bar{m}(x_0) \\ \bar{m}(x_1) \end{bmatrix} \right)$$

$$= 0.082388$$

$$\hat{\rho}_{X_1} = \begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix} \begin{bmatrix} p^*_{22} & p^*_{23} \\ p^*_{32} & p^*_{33} \end{bmatrix} \begin{bmatrix} \rho_2 \\ \rho_3 \end{bmatrix} / \left(\begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix} \begin{bmatrix} \bar{m}(x_2) \\ \bar{m}(x_3) \end{bmatrix} \right)$$

$$= 0.061203$$

Writing

$$\Pi_0 = \begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \\ \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} \text{ and } \Pi_1 = \begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \\ \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix}$$

we also have

$$\tilde{\sigma}^2_{X_0} = \int_{X_0} \pi_{X_0}(dx) \left[R_0(P\rho - m_0\hat{\rho}_{X_0}) + \frac{1}{2}(P\rho - m_0\hat{\rho}_{X_0})^2 \right] / m_0$$

$$= \begin{bmatrix} \pi_{X_0}(0) & \pi_{X_0}(1) \end{bmatrix} \left[\sum_{n=0}^{\infty} \int_{X_0} (P_0^n - \Pi_0) \left(\begin{bmatrix} (p^*_{00}\rho_0 + p^*_{01}\rho_1 - m_0\hat{\rho}_{X_0})^2 \\ (p^*_{10}\rho_0 + p^*_{11}\rho_1 - m_0\hat{\rho}_{X_0})^2 \end{bmatrix} \right.$$

$$\left. + \frac{1}{2} \begin{bmatrix} (p^*_{00}\rho_0 + p^*_{01}\rho_1 - m_0\hat{\rho}_{X_0})^2 \\ (p^*_{10}\rho_0 + p^*_{11}\rho_1 - m_0\hat{\rho}_{X_0})^2 \end{bmatrix} \right) \Bigg] / m_0$$

$$= 0.000121$$

and

$$\tilde{\sigma}^2_{X_1} = \int_{X_1} \pi_{X_1}(dx) \left[R_0(P\rho - m_0\hat{\rho}_{X_1}) + \frac{1}{2}(P\rho - m_0\hat{\rho}_{X_1})^2 \right] / m_1$$

$$= \begin{bmatrix} \pi_{X_1}(0) & \pi_{X_1}(1) \end{bmatrix} \left[\sum_{n=0}^{\infty} \int_{X_1} (P_1^n - \Pi_1) \left(\begin{bmatrix} (p^*_{22}\rho_2 + p^*_{23}\rho_3 - m_1\hat{\rho}_{X_1})^2 \\ (p^*_{23}\rho_2 + p^*_{33}\rho_3 - m_1\hat{\rho}_{X_1})^2 \end{bmatrix} \right.$$

$$\left. + \frac{1}{2} \begin{bmatrix} (p^*_{22}\rho_2 + p^*_{23}\rho_3 - m_1\hat{\rho}_{X_1})^2 \\ (p^*_{32}\rho_2 + p^*_{33}\rho_3 - m_1\hat{\rho}_{X_1})^2 \end{bmatrix} \right) \Bigg] / m_1$$

$$= 0.000095$$

Now, using algorithms of Double Averaging under the Normal Deviations Approximation, we have

$$\breve{\rho} = \frac{1}{2}(\hat{\rho}_{X_0} + \hat{\rho}_{X_1}) = \frac{1}{2}(0.082388 + 0.061203) = 7.1796 \times 10^{-2}$$

and

$$\breve{\sigma}^2 = \frac{1}{2}(\tilde{\sigma}_{X_0}^2 + \tilde{\sigma}_{X_1}^2) = \frac{1}{2}(0.000121 + 0.000095) = 1.0815 \times 10^{-5}$$

From the above data we have

$$\breve{S}(t) = S_0 e^{\breve{\rho} t + T^{-\frac{1}{2}} \breve{\sigma} W(t)}$$

where

$$\breve{\rho} = 7.1796 \times 10^{-2}$$
$$\breve{\sigma}^2 = 1.0815 \times 10^{-5}$$

Suppose that

$$S_0 = 20 \text{ (starting stock price)}$$
$$E = 18 \text{ (strike price)}$$
$$r = 0.01 \text{ (risk-free rate)}$$
$$T = 4 \text{ (time to expiry)}$$

By algorithms presented in Section (4), we have

$$d_1 = 22.104$$
$$d_2 = 22.097$$

So the value of a European call will be (after calculations)

$$C^E = 2.705790$$

The value of a European put will be (after calculations)

$$P^E = 0$$

The value of an American call will be (it is the same price as a European call)

$$C^A = 2.705790$$

The value of an American put will be (using the backwardation method with $M = 100$ steps as in Section 5.4)

$$P^A = 0$$

8.6.9 Guide to MATLAB code DoubleAveragedNormalDeviation.m

The MATLAB code DoubleAveragedNormalDeviation.m is designed to graph and calculate option prices (European and American) for a GMRP-modulated asset with exponentially distributed waiting times for jumps (Poisson waiting times). It takes as input the transition matrix, the ρ values, and the λ in each case for exponential distributions in each state. Also, it takes information on how to divide the transition matrix. If the option to graph paths is selected, it outputs sample asset paths. If the option to price options is selected, it prices American or European options according to the algorithm described in the section (5.4) for the case of double averaging under a merged GMRP. The code may be found in Appendix B.9.

8.6.10 Sample run of MATLAB code DoubleAveragedNormalDeviation.m

We run the code on the same data as given in Section (8.6.8). The results are identical to those calculated in Section (8.6.8). Here is the output:

```
 Below are displayed the computed critical values for each state of the embedde
d Markov chain:
 Embedded State #; rhohat; sigmahat squared
 1 0.082388 0.000121
 2 0.061203 0.000095

 Below is the averaged values for the Markov chai:n
 rhohat averaged; sigmahat squared averaged
  7.1796e-002  1.0815e-005
```

```
d1 =  22.104
d2 =  22.097

 European option information is below:
 European Call value: 2.705790
 Delta of European Call: 1.000000
 European Put value: 0.000000
 Delta of European Put: 0.000000



 The value of an American put is:  0

 The value of an American call is:   2.7058
```

Chapter 9

Poisson Approximation

9.1 The Poisson Approximation [10, 23]

9.1.1 Basic Theoretical Considerations

A generalization of Equation (1.2.3) can be made by allowing $\rho_k^T(x)$ to be a sequence of random variables for all $x \in X$ and for all $T > 0$. Now, let us consider the process S_t^T defined as below:

$$S_t^T = S_0 \prod_{k=1}^{\nu(tT)} (1 + \rho_k^T(x_k, \omega)) := S_0 \prod_{k=1}^{\nu(tT)} (1 + T^{-1}\rho_k(x_k)) \tag{9.1.1}$$

It can be shown that for large T-values we have the following, where $(x_k, \tau_k)_{k\in\mathbb{Z}_+}$is a Markov renewal process, $\{x_k\}_{k\in\mathbb{Z}_+}$is an embedded Markov chain, $(\tau_k)_{k\in\mathbb{Z}_+}$ are the moments of the jumps, $P(\theta_{k+1} \le t | x_k = x) = 1 - e^{-q(x)t}$, $\theta_k := \tau_{k+1} - \tau_k$, and $P(x,A) := P(x_{k+1} \in A | x_k = x)$.

$$\ln \frac{S_t^T}{S_0} = \sum_{k=1}^{\nu(tT)} \ln(1 + \rho_k^T(x_k)) \approx T^{-1} \sum_{k=1}^{\nu(tT)} \rho_k(x_k) \tag{9.1.2}$$

It should be noted that $x(t) := x_{\nu(t)}$ (where $\nu(t)$ is a counting process) is a Markov process with the transitional kernel given by the following:

$$Q(x,A,t) = P(x,A)(1 - e^{-q(x)t}) \tag{9.1.3}$$

Now assume that $x(t)$ is an ergodic Markov process with stationary distribution $\pi(dx)$. \then we have that $(x_n)_{n\in\mathbb{Z}_+}$is also an ergodic Markov chain. Moreover, its stationary distribution $p(dx)$ is given by the following equations:

$$\pi(dx)q(x) = qp(dx),\ q := \int_X \pi(dx)q(x) \tag{9.1.4}$$

We write $F_x^T(z) := P(\rho_k^T(x,\omega) \leq z)$ for all $z \in R$. Under the additional assumption that for any fixed sequence x_k, the sequence $\rho_x^T(x_k)$ are independent random variables for $k \in \mathbb{Z}_+$, we also have a convergence-determining class, defined as follows: Given $g \in C_3(\mathbb{R})$ (where $C_3(\mathbb{R})$ is the set of all functions that are 3 times continuously differentiable on $\mathbb{R}$) , $g(u)$ is a real-valued continuous function, and has the property that $g(u)/u \to 0$ as $|u| \to 0$. Such a g is a function in our convergence-determining class.

We suppose furthermore that the sequence $\rho_k^T(x_k)$ satisfies the Poisson approximation:

$$\int_R g(u)F_x^T(du) = T^{-1}[F_x(g) + \theta^T(x)] \tag{9.1.5}$$

where we have the Taylor remainder

$$\sup_{x\in R}|\theta^T(x)| \to 0,\ T \to +\infty$$

that $\sup_{x\in E}|F_x(g)| \leq F(g) < \infty$, and the the conditions for square-integrability are met:

$$\sup_{x\in E}\int_R u^2 F_x^T(du) < +\infty \tag{9.1.6}$$

where the distribution function $F_x(du)$ is defined by the relation

$$F_x(g) = \int_R g(u)F_x(du)\ g \in C_3(\mathbb{R}) \tag{9.1.7}$$

Swishchuk, Islam, Korolyuk, and Limnios [10, 23] present the result that the right hand side of 9.1.2 converges weakly to the compound Poisson process $P(t)$ with deterministic drift as follows:

$$\lim_{T\to\infty}\sum_{k=1}^{\nu(tT)}\rho_k^T(x_k)=P(t):=\sum_{k=1}^{N_0(t)}\alpha_k^0+a_0qt \tag{9.1.8}$$

We obtain the distribution function $F^0(x)$ for the i.i.d.r.v. $\alpha_k^0, k\geq 0$ from the convergence-determining class $C_3(\mathbb{R})$ by the following:

$$Eg(\alpha_k^0)=\int_{\mathbb{R}}g(z)F^0(dz)=\frac{\hat{F}(g)}{\hat{F}(1)},\ g\in C_3(\mathbb{R}) \tag{9.1.9}$$

In the above we have denoted

$$\hat{F}(g):=\int_X p(dx)F_x(g),\ \hat{F}(1):=\int_X p(dx)F_x(1) \tag{9.1.10}$$

where we require that $F_x(g)$ be such that

$$\int_X g(z)F_x^T(dz)=T^{-1}[F_x(g)+o_T(x,g)] \tag{9.1.11}$$

where we have $\sup_{x\in X}o_T(x,g)\to 0,\ T\to+\infty$, and $\sup_x|F_x(g)|\leq+\infty$.

Also, we require that $g(z)$ be such that

$$g(z)/z^2\to 0,\ |z|\to 0 \tag{9.1.12}$$

and $F_x(g)=\int_{\mathbb{R}}g(z)F_x(dz)$.

The counting Poisson process $\nu(t)=N_0(t)$ has intensity

$$q_0:=q\hat{F}(1) \tag{9.1.13}$$

The drift a_0 is

$$a_0=\hat{a}-\hat{F}(1)E\alpha_1^0,\ \hat{a}:=\int_X p(dx)a(x) \tag{9.1.14}$$

where $a(x)$ is defined asymptotically as follows, where $\sup_x|a(x)|\leq a<+\infty$ and $\sup_{x\in X}o_T(x)\to 0$ as $T\to+\infty$:

$$\int_{\mathbb{R}} zF_x^T(dz) = T^{-1}[a(x) + o^T(x)] \tag{9.1.15}$$

Writing $S_p(t) := \lim_{T\to+\infty} S_t^T$, we have

$$\lim_{T\to\infty}\left(\ln\frac{S_t^T}{S_0}\right) = \ln\frac{S_p(t)}{S_0} = P(t) = \sum_{k=1}^{N_0(t)} \alpha_k^0 + a_0 qt \tag{9.1.16}$$

or, in the familiar form

$$S_p(t) = S_0 \exp\left(\sum_{k=1}^{N_0(t)} \alpha_k^0 + a_0 qt\right) \tag{9.1.17}$$

9.1.2 Canonical Presentation

Note that we have the following, where $\mu$is a measure of jumps of the process $N_0(t)$:

$$\sum_{k=1}^{N_0(t)} \alpha_k^0 = \int_0^t \int_0^{+\infty} y\mu(dy;ds)$$

Then the Poisson GMRP has the following form:

$$\ln\frac{S_p(t)}{S_0} = \int_0^t \int_0^{+\infty} y\mu(dy;ds) + a_0 qt \tag{9.1.18}$$

This implies that the dynamics of a stock price under the Poisson GMRP can be represented as follows:

$$\frac{dS_p(t)}{S_p(t)} = \int_0^{+\infty} y\mu(dy;dt) + a_0 qdt \tag{9.1.19}$$

9.2 Poisson Approximation with Finite Number of Jumps [9, 10, 23]

Consider the increment process $\sum_{k=1}^{\nu(tT)} \rho_k^T(x_k)$ with a finite number of jump values as follows (where clearly we have $\sum_{m=1}^{M} q_m = 1$):

$$P\left(\rho(x)=\frac{1}{T}a_m\right)=q_m-\frac{1}{T}p_m(x),\ 1\le m\le M,\ P(\rho_k^x=b_m)=\frac{1}{T}p_m(x),\ 1\le m\le M \quad (9.2.1)$$

Then the increment process $\sum_{k=1}^{\nu(tT)}\rho_k^T(x_k)$ and its stochastic exponential will converge (weakly) to the Poisson process

$$P(t):=\sum_{k=1}^{N_0(t)}\alpha_k^0+a_0qt$$

where we have jump distribution given by

$$P(\alpha_k^0=b_m)=p_m^0,\ 1\le m\le M,\ p_m^0=\frac{\hat{p}_m}{\hat{p}} \quad (9.2.2)$$

$$\hat{p}_m=\int_E\rho(dx)p_m(x),\ 1\le m\le M,\ \hat{p}=\sum_{m=1}^{M}\hat{p}_m \quad (9.2.3)$$

In this case, we have the intensity of the counting Poisson process $\nu_0(t)$, $t\ge 0$ to be defined by $q_0:=q\hat{p}$. The drift parameter is given by

$$a_0=\sum_{m=1}^{M}a_mq_m \quad (9.2.4)$$

Furthermore, q_0, the intensity of jumps, is determined by the averaged initial probability $\hat{p}$ and the averaged intensity q of the switched Markov process.

9.3 Financial Applications [9, 10, 23]

9.3.1 Presentation as Levy Process

Consider the case where $A=(A_t)_{0\le t\le T}$ is a special case of Levy jump-diffusion; it is a Brownian motion plus a compensated compound Poisson process. In this case we can describe the paths of this process as follows, where $b\in R,\sigma\in\mathbb{R}_{\ge 0},W=(W_t)_{0\le t\le T}$ is a standard Brownian motion, $N=(N_t)_{0\le t\le T}$ is a Poisson process with parameter λ, and $J=(J_k)_{k\ge 1}$ is an i.i.d. sequence of

random variables with probability distribution function F and $E(J) = \kappa < \infty$:

$$L_t = bt + \sigma W_t + \left(\sum_{k=1}^{N_t} J_k - t\lambda\kappa \right) \tag{9.3.1}$$

The Levy triplet for this formulation is (b, c, ν), and we have the presentation of the Levy process as follows (here the random variable $L_t$of the Levy jump-diffusion is infinitely divisible with Levy triplet $b = b \cdot t, c = \sigma^2 \cdot t$ and $\nu = (\lambda F) \cdot t$):

$$E[e^{iux}] = \exp\left(ibu - \frac{u^2 c}{2} + \int_{\mathbb{R}} (e^{iux} - 1 - iux) 1_{\{|x|<1\}} \nu(dx) \right) \tag{9.3.2}$$

9.3.2 Risk-Neutral Measure

Under the risk-neutral measure $\bar{P}$, the asset price obtains the form

$$S_t = S_0 \exp L_t \tag{9.3.3}$$

with Levy triplet $(\bar{b}, \bar{c}, \bar{\nu})$ and canonical decomposition as follows (with $\bar{W}$ a $\bar{P}$-Brownian motion, and where $\bar{\nu}^L$ is the $\bar{P}$-compensator of the jump measure μ^L):

$$L_t = \bar{b}t + \sqrt{\bar{c}}\bar{W}_t + \int_0^t \int_{\mathbb{R}} x(\nu^L - \bar{\nu}^L)(ds, dx) \tag{9.3.4}$$

Under the assumption that $\bar{P}$ is a risk neutral measure, the asset price has rate of return $\mu = r - \delta$ and the discounted and re-invested process $(e^{(r-\delta)} S_t)_{0 \le t \le T}$ is a $\bar{P}$ - martingale (here $r \ge 0$ is the interest rate and $\delta \ge 0$ is the continuous dividend yield of the asset). The drift term $\bar{b}$ takes the form

$$\bar{b} = r - \delta - \frac{\bar{c}}{2} - \int_{\mathbb{R}} (e^x - 1 - x) \bar{\nu}(dx) \tag{9.3.5}$$

Putting

$$L_t = \sum_{k=1}^{N_0(t)} \alpha_k^0 + a_0 qt \tag{9.3.6}$$

we have the following presentation of L_t:

$$\begin{aligned} L_t &= \bar{b}t + \int_0^t \int_{\mathbb{R}} x * u^L(ds,dx) - E\left(\sum_{k=1}^{N_0(t)} \alpha_k^0\right) \\ &= \bar{b}t + \int_0^t \int_{\mathbb{R}} x * u^L(ds,dx) - \int_0^t \int_0^\infty x * \lambda \bar{F}(dx)(ds) \end{aligned} \tag{9.3.7}$$

where we have the distribution of jumps under the risk neutral measure$\bar{F}(dx)$, so $\bar{\nu}^L(dt,dx) = \lambda * \bar{F}(dx)(dt)$, In the above, the drift term $\bar{b}$ is given by the following (where we have written $\bar{\nu}(dx) = \lambda \bar{F}(dx)$):

$$\bar{b} = r - \int_{\mathbb{R}} (e^x - 1 - x)\bar{\nu}(dx) \tag{9.3.8}$$

9.3.3 Market Incompleteness

Under the assumption that the price process of an asset is modeled by an exponential Levy process under both the real and the risk neutral measures (denoted P and $\bar{P}$ respectively), where the measures and assumed to be equivalent (with Levy triplets (b,c,ν) and $(\bar{b},\bar{c},\bar{\nu})$ respectively), by Girsanov's theorem, we have the following result, writing $\bar{c} = c$, $\bar{\nu} = Y \cdot \nu$, and where (β, Y) is the tuple of the functions related to the density process:

$$\bar{b} = b + c\beta + x(Y-1)\nu \tag{9.3.9}$$

The martingale condition implies:

$$\bar{b} = r - \frac{\bar{c}}{2} - (e^x - 1 - x) * \bar{\nu} \tag{9.3.10}$$

Equating 9.3.9 and 9.3.10 and writing $\bar{c} = c, \bar{\nu} = Y\nu$ we have that

$$\begin{aligned} 0 &= b + c\beta + x(Y-1) * \nu - r + \tfrac{\bar{c}}{2} + (e^x - 1 - x) * \bar{\nu} \\ \iff\ 0 &= b - r + c(\beta + \tfrac{1}{2}) + ((e^x - 1)Y - x) * \nu \end{aligned} \tag{9.3.11}$$

Indeed, every solution (β, Y) for the above corresponds to a different equivalent martingale measure, which explains why the market is not complete in this formulation. So, (β, Y) can be thought

of as the tuple of "market price of risk".

Chapter 10

Conclusions and Future Research

The MATLAB codes developed to produce quantitative results for the mentioned results in the Swishchuk-Limnios paper [24] as well as the Aase model and options valuation under diffusion and normal deviated GMRP approximations have proven rigorous and capable of handling a wide range of data. Future improvements may include efficiency improvement by eliminating as many bit operations as possible. We elaborate in particular on two of the codes presented:

In the case of the code ISol.m, it may be desirable to build in a Black-Scholes approximation scheme to determine the bounds on δ_i. This was left out of ISol.m as presented in this paper for purposes of allowing complete generality in approximations of these δ_i-bounds. It would also be desirable to attempt to determine δ_i-bounds directly from the function $g(s,x)$, whilst taking this function as user input.

In the case of the code EGO.m, investigation into more efficient ways of enumerating all k-length Markov paths is warranted in order to improve efficiency of the code. This may also allow for a higher integer for the "recommended" upper limit on iterations. Also, an analysis of the convergence of $C_T(x)$ in Equation 7.3.3 is warranted to quantify as well as guarantee the speed of convergence when running the code EGO.m.

Further results on GMRP-modeled stocks shall also yield results to be quantified in MATLAB codes, and may produce results to improve the three presented here as well.

On the issue of general GMRP and approximations, it is strongly recommended that a formula be found for calibration in the general case for N-state GMRP. The practicality and usefulness

of any financial model depends heavily on whether or not it can be calibrated to given market data. Furthermore, once these calibrations are done, back-testing should be executed using existing options-based strategies (priced using the Diffusion and Normal Deviated approximations of the calibrated GMRP) to evaluate whether the approximations given here are valid and satisfactory in actual trading. Furthermore, it would be desirable to attempt to derive a new "backwardation" American put valuation scheme that would eliminate the need for the so-called "risk-neutral assumption" of $m = r$ from having to be made in Equation (5.4.7).

In addition, a comparison of option valuation in the Diffusion and Normal Deviated approximations for the same market data would be very useful to analyze the consistency of the pricing schemes. Additionally, schemes should be found to simplify the expressions for the Diffusion and Normal Deviated approximations in the case of simple (i.e. not Double Averaged) Merged GMRP. The difficulty arises since the equations (8.5.18) and (8.6.19) for Diffusion and Normal Deviated (not Double Averaged) Merged approximations contain the term of the current state of the Markov chain in the drift and standard deviation parameters for the GBM approximation. Hence, it is necessary to simulate an actual Markov chain in order to see the realization of this approximation in this scheme, in the absence of any other simplification assumptions. Currently there are no schemes to accomplish this, other than the calculation of limiting probabilities in Double Averaging as has already been presented. If found, it would be useful to include this in the aforementioned tests for consistency between option valuation in different approximation schemes for certain given historical data. Furthermore, a formal presentation of a calibration algorithm for the GMRP given real-world market data would be very useful in assessing appropriateness of GMRP in modelling a particular asset or asset class.

Bibliography

[1] K. Aase. Contigent claims valuation when the securities price is a combination of an Ito process and a random point process. *Stochastic Processes and their Applications*, 40:99–141, 1995.

[2] V. Anisimov. Switching processes: averaging principle, diffusion approximation and applications. *Acta Applicandae Mathematica*, 40.2:95–141, 1995.

[3] F. Black and M. Scholes. The Pricing of Options and Corporate Liabilities. *The Journal of Political Economy*, pages 637–654, 1973.

[4] M. Capinski and T. Zastawniak. *Mathematics for Finance: An Introduction to Financial Engineering*. Springer, 2003.

[5] J. Cox, S. Ross, and M. Rubinstein. Option pricing: A simplified approach. *Journal of financial Economics*, 7.3:229–263, 1979.

[6] R. Elliott, G. Sick, and M. Stein. Modelling electricity price risk. *Preprint, University of Calgary*, 2003.

[7] W. S. Jewell. Markov-renewal programming. II: Infinite return models, example. *Operations Research*, 11:949–971, 1963.

[8] S. Karlin and H. Taylor. *A First Course in Stochastic Processes*. Academic press, 1975.

[9] V. Korolyuk and N. Limnios. Increment processes and its stochastic exponential with Markov switching in Poisson approximation scheme. *Computers & Mathematics with Applications*, 46.7:1073–1079, 2003.

[10] V. Korolyuk and N. Limnios. Poisson approximation of increment processes with Markov switching. *Theory of Probability & Its Applications*, 49.4:629–644, 2005.

[11] V. Korolyuk and A. Swishchuk. *Evolution of Systems in Random Media*. CRC press, 1995.

[12] N. Limnios, A. Swishchuk, et al. Discrete-time semi-Markov random evolutions and their applications. *Advances in Applied Probability*, 45:214–240, 2013.

[13] Robert C Merton. Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3.1:125–144, 1976.

[14] E. H. Moore and R. Pyke. Estimation of the transition distributions of a Markov renewal process. *Annals of the Institute of Statistical Mathematics*, 20:411–424, 1968.

[15] R. Pyke. Markov renewal processes: definitions and preliminary properties. *The Annals of Mathematical Statistics*, pages 1231–1242, 1961.

[16] R. Pyke. Markov renewal processes with finitely many states. *The Annals of Mathematical Statistics*, pages 1243–1259, 1961.

[17] R. Pyke and R. Schaufele. Limit theorems for Markov renewal processes. *The Annals of Mathematical Statistics*, pages 1746–1764, 1964.

[18] S. Ross. *Introduction to probability models*. Academic press, 2014.

[19] S. Shreve. *Stochastic calculus for finance II: Continuous-time models*. Springer, 2004.

[20] A. Swishchuk and Md. Islam. Diffusion approximations of the geometric Markov renewal processes and option price formulas. *International Journal of Stochastic Analysis*, 2010.

[21] A. Swishchuk and Md. Islam. The Geometric Markov Renewal Processes with Application to Finance. *Stochastic Analysis and Applications*, 29:684–705, 2011.

[22] A. Swishchuk and Md. Islam. Normal Deviation and Poisson Approximation of a Security Market by the Geometric Markov Renewal Processes. *Communications in Statistics-Theory and Methods*, 42:1488–1501, 2013.

[23] A. Swishchuk and Md. Islam. *Random Dynamical Systems in Finance*. CRC Press, 2013.

[24] A. Swishchuk and N. Limnios. Optimal stopping of geometric Markov renewal processes and pricing of American and European options. In *15th International Congress on Insurance, Mathematics & Economics (IME 2011)*, 2011.

Appendix A

Computing the Potential R_0 for a Markov Chain

We have that the potential of a Markov chain is R_0 where R_0 is defined as follows [11]:

$$R_0 = \sum_{n=0}^{\infty} (P^n - \Pi)$$

where P is the Markov transition matrix and Π is the matrix of limiting probabilities. We have convergence of P to Π by the definition of limiting probabilities, since we can view $(i,j)^{th}$ entry of the k^{th} power of the Markov transition matrix P as the probability to go from state i to state j in k steps. Thus, clearly, Π will be the limit of P^k as $k \to \infty$.

We present an example to show how to calculate the potential R_0. Suppose we have the P-matrix as follows

$$P = \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} = \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix}$$

We will compute

$$R_0 = \sum_{n=0}^{\infty} (P^n - \Pi)$$

for the given P.

If we solve

$$\pi P = \pi, \text{ i.e. } \begin{bmatrix} \pi_0 & \pi_1 \end{bmatrix} \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix} = \begin{bmatrix} \pi_0 & \pi_1 \end{bmatrix}$$

we obtain $\pi_0 = \pi_1 = 0.5$, thus

$$\Pi = \begin{bmatrix} \pi \\ \pi \end{bmatrix} = \begin{bmatrix} \pi_0 & \pi_1 \\ \pi_0 & \pi_1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

Next we must diagonalize P. We find its eigenvalues λ_1, λ_2 by solving for λ in

$$\det\left(\begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) = 0$$

We obtain

$$\lambda_1 = 0.96, \ \lambda_2 = 1$$

with associated eigenvectors

$$X_1 = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \quad X_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

Thus we have the canonical decomposition

$$P = XDX^{-1} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0.96 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^{-1}$$

and since

$$P^k = XD^kX^{-1} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0.96^k & 0 \\ 0 & 1^k \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0.96^k & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}0.96^k + \frac{1}{2} & -\frac{1}{2}0.96^k + \frac{1}{2} \\ -\frac{1}{2}0.96^k + \frac{1}{2} & \frac{1}{2}0.96^k + \frac{1}{2} \end{bmatrix}$$

since, as can be verified, $X^{-1} = X$.

Notice that, as we need,

$$\lim_{k\to\infty} P^k = XD^kX^{-1} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^{-1} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} = \Pi$$

Now we have

$$\sum_{n=0}^{\infty}(P^n - \Pi) = \frac{1}{2}\sum_{n=0}^{\infty}\begin{bmatrix} 0.96^n & -0.96^n \\ -0.96^n & 0.96^n \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\left(\sum_{n=0}^{\infty} 0.96^n\right)$$

$$= \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\frac{1}{1-0.96} = \frac{1}{2}\cdot\frac{1}{0.04}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 12.5 & -12.5 \\ -12.5 & 12.5 \end{bmatrix}$$

In practice, we can approximate $\sum_{n=0}^{\infty}(P^n - \Pi)$ by computing the sums $\sum_{n=0}^{k}(P^n - \Pi)$, and stopping when the next term $P^{k+1} - \Pi$ is less than some specified value.

Appendix B

MATLAB Codes

B.1 MATLAB Code for Aase Model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code, AaseSimulation, simulates the process
% St=S0 prod of 1 to N(t) (Poisson) * (1+Yk)
% where Yk are i.i.d. variables from the uniform distribution on Scale*[-1,1].
% Zachary Moyer
% University of Calgary
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Take necessary information from the user:
fprintf("\n This code simulates an Aase-compatible pure jump process with ");
fprintf("\n Yk i.i.d. distributed as uniform on Scale*[-1,1] \n");
fprintf("\n Input lambda: ");
lambda=input(" ");
fprintf("\n Enter the starting stock price, S_0 : ");
S=input(" ");
fprintf("\n What is the maximum time T you want to extend the process to? ");
fprintf("\n Please enter T: ");
Tmaxx=input(" ");
fprintf("\n Enter the scale factor C for the jump sizes; i.e. from uniform distribution [-C,C];");
fprintf("\n Generally 0<C<1 :");
```

```
C=input(" ");
fprintf("\n Enter the maximum allowed number of jumps in the time ");
fprintf("\n interval; recommended value is 2*1e4: ");
maxjumps=input(" ");
maxjumps=max(maxjumps,1e4); % ensure maxjumps is not too low
maxjumps=min(maxjumps,1e5); % ensure maxjumps is not too high
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulate jumptimes
jumptimes=exprnd(1/lambda);
i=1;
while (sum(jumptimes)<=Tmaxx)&& (i<=maxjumps);
i=i+1;
jumptimes(i)=exprnd(1/lambda);
if (i==maxjumps);
fprintf("\n \n WARNING: MAXIMUM NUMBER OF ALLOWED JUMPS REACHED!!! \n \n
");
end;
end;
% calculate jumps
numjumps=length(jumptimes);
Y=0;
for i=1:1:numjumps;
Y(i)=C*2*(rand-0.5);
end;
% creat St vector (for asset path)
St = S;
for i=2:1:numjumps+1;
St(i)=St(i-1)*(1+Y(i-1));
end;
% create time vector
T = 0;
for i=2:1:length(St);
T(i)=sum(jumptimes(1:i-1));
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the paths
figure(1);
plot(T,St);
```

```
xlim([0 Tmaxx]);
ylim([0 1.1*max(St)]);
xlabel("Time");
ylabel("Stock Price");
title("Pure Jump Process");
grid on
```

B.2 MATLAB code GMRP.m

```
%%%%%%%%%%%%%%%%%%%%
%This program generates a GMRP asset path.
%%%%%%%%%%%%%%%%%%%%
%Get the number of states for the Markov chain
D = input('Enter the number of states in the Markov chain: \n');
%Set up the Markov matrix
M = zeros(D,D);
for i = 1:1:D
for j = 1:1:D
fprintf('Prob to go from state %d to %d',i,j);
M(i,j) = input(' ');
end
end
%Display the Markov matrix
fprintf('Matrix for Markov chain looks like this: \n'); M
%Get the function rho from the user:
fprintf('Enter function rho defind on each state of the Markov chain: \n')
Rho=zeros(1,D);
for i=1:D
fprintf('Rho %d = ',i);
Rho(i)=input(' ');
end
Lam=0;
%Get lambda vector from user:
for i=1:D;
fprintf("\n Enter lambda for state %d : ",i);
Lam(i)=input(" ");
end;
```

```
fprintf("\n Lambdas in each state are as follows: "); disp(Lam);
Lam=1./Lam;
%Get startingstock price from user:
SO = input('\n Enter the starting stock price: ');
%Get ending time:
fprintf("\n Enter length of time to simulate price:");
END = input(" ");
%Simulate the Markov chain.
CDF = cumsum(M,2); % cumulative distribution function
i = input('\n Enter the starting state: ');
MC = 0;
MC(1) = i;
TIMESBTJUMPS=0; NUMJUMPS=0;
t=2;
while sum(TIMESBTJUMPS)<=END; % simulation for periods.
TIMESBTJUMPS(t)=exprnd(Lam(MC(t-1)));
ug = rand(1); % create pseudo-random numbers from the uniform dist.
j = find(CDF(i,:)>=ug,1,'first');
MC(t) = j;
i = j;
t=t+1;
NUMJUMPS=NUMJUMPS+1;
end
%Find times of jumps.
TIMES=0;
for i=2:1:(NUMJUMPS+1);
TIMES(i)=sum(TIMESBTJUMPS(1:i));
end;
%Generate the stock price path:
S=zeros(1,NUMJUMPS+1);
%Set the starting stock price form user input
S(1)=SO;
%Make a variable to count the cumulative product
p = 1;
for i=2:NUMJUMPS+1
p = p*(1+Rho(MC(i-1)));
S(i)= SO*p;
end
```

```
%Display stock price path
plot(TIMES,S);
xlim([0 END]);
ylim([0 1.1*max(S)]);
xlabel("Time");
ylabel("Stock Price");
title("GMRP with Poisson jumps");
grid on
```

B.3 MATLAB code AOP.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code, AaseSimulation, simulates the process
% St=S0 prod of 1 to N(t) (Poisson) * (1+Yk)
% where Yk are i.i.d. variables from the uniform distribution on Scale*[-1,1].
% Zachary Moyer
% University of Calgary
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Take necessary information from the user:
fprintf("\n This code simulates an Aase-compatible pure jump process with ");
fprintf("\n Yk i.i.d. distributed as uniform on Scale*[-1,1] \n");
fprintf("\n Input lambda: ");
lambda=input(" ");
fprintf("\n Enter the starting stock price, S_0 : ");
S=input(" ");
fprintf("\n What is the maximum time T you want to extend the process to? ");
fprintf("\n Please enter T: ");
Tmaxx=input(" ");
fprintf("\n Enter the scale factor C for the jump sizes; i.e. from uniform distribution [-C,C];");
fprintf("\n Generally 0<C<1 :");
C=input(" ");
fprintf("\n Enter the maximum allowed number of jumps in the time ");
fprintf("\n interval; recommended value is 2*1e4: ");
maxjumps=input(" ");
maxjumps=max(maxjumps,1e4); % ensure maxjumps is not too low
maxjumps=min(maxjumps,1e5); % ensure maxjumps is not too high
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulate jumptimes
jumptimes=exprnd(1/lambda);
i=1;
while (sum(jumptimes)<=Tmaxx)&& (i<=maxjumps);
i=i+1;
jumptimes(i)=exprnd(1/lambda);
if (i==maxjumps);
fprintf("\n \n WARNING: MAXIMUM NUMBER OF ALLOWED JUMPS REACHED!!! \n \n
");
end;
end;
% calculate jumps
numjumps=length(jumptimes);
Y=0;
for i=1:1:numjumps;
Y(i)=C*2*(rand-0.5);
end;
% creat St vector (for asset path)
St = S;
for i=2:1:numjumps+1;
St(i)=St(i-1)*(1+Y(i-1));
end;
% create time vector
T = 0;
for i=2:1:length(St);
T(i)=sum(jumptimes(1:i-1));
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the paths
figure(1);
plot(T,St);
xlim([0 Tmaxx]);
ylim([0 1.1*max(St)]);
xlabel("Time");
ylabel("Stock Price");
title("Pure Jump Process");
grid on
```

B.4 MATLAB code ISol.m

```
%%%%%%%%%%%%%%%%%%%%%
%This program solves the integral equation for the limit as N –> infinity for GMRP.
%%%%%%%%%%%%%%%%%%%%%
%Get the number of states for the Markov chain
D = input('Enter the number of states in the Markov chain: \n');
%Set up the Markov matrix
M = zeros(D,D);
for i = 1:1:D
for j = 1:1:D
fprintf('Prob to go from state %d to %d',i,j);
M(i,j) = input(' ');
end
end
%Display the Markov matrix
fprintf('Matrix for Markov chain looks like this: \n'); M
%Get the function rho from the user:
fprintf('Enter function rho defind on each state of the Markov chain: \n')
Rho=zeros(1,D);
for i=1:D
fprintf('Rho %d = ',i);
Rho(i)=input(' ');
end
%Get startingstock price from user:
SO = input('Enter the starting stock price: ');
%Get starting state from the user:
XO = input('Enter the starting state: ')
%Get beta:
Beta = input('Enter the value of beta: ');
%Get the interest rate, r:
r = input('Enter the interest rate r: ');
%Inform the user about assumptions for this pricing model.
fprintf('We assume that for small changes in the stock price, we have that
\ngiven a state x_i, \nV(S_1,x_i) - V(S_0,x_i) = delta_i \nfor
some real (small) number delta_i, where S_1 = S_0 (1+rho(x_i)). \nPlease
enter the lower bound, s_i, and \n the upper bound, delta_i, for each
state of the chain, \n so that s_i <= delta_i <=t_i for each of i from
```

```
1 to D. \n ');
%Get the K-i from the user.
S=zeros(1,D); T=zeros(1,D); fprintf('Now, enter s_i and t_i one by one: ');
for i = 1:1:D;
fprintf('s_%d = ',i); S(i)=input(' ');
fprintf('t_%d = ',i); T(i)=input(' ');
end
%Set up modified matrix.
Mstar = M;
for i=1:1:D
Mstar(i,i)=M(i,i)-(1+r)/Beta;
end
Z=inv(Mstar)*(-1)*M
%Set up A=xB, F, UB, and LB parameters
F=zeros(1,2*D);
for i=(D+1):1:(2*D)
F(i)=Z( XO , i-D );
end
A=[eye(D),-Z];
Aeq=A;
B=zeros(2*D,1);
Beq = B;
LB=[zeros(1,D)';S'];
UB=[Inf(D,1);T'];
Lowerbound=F*linprog(F, A, B, Aeq, Beq, LB, UB);
Upperbound=F*linprog(-F,A,B,Aeq,Beq,LB,UB);
fprintf('The lower bound is: \n'); disp(Lowerbound);
fprintf('The upper bound is: \n'); disp(Upperbound);
if linprog(-F,A,B,Aeq,Beq,LB,UB) < 0
fprintf('WARNING: INPUT DATA RESULTS IN NEGATIVE STOCK PRICES; BOUNDS ON
DELTA
ARE WRONG. ');
end
```

B.5 MATLAB code EuroGO.m

%%%%%%%%%%%%%%%%%%%%

```
%This program calculates the value of a European option under GMRP assumptions to within error
bounds
%or exceeding of maximum iteration count.
%%%%%%%%%%%%%%%%%%%%%
%Get the number of states for the Markov chain
D = input('Enter the number of states in the Markov chain: \n');
%Set up the Markov matrix
M = zeros(D,D);
for i = 1:1:D
for j = 1:1:D
fprintf('Prob to go from state %d to %d',i,j);
M(i,j) = input(' ');
end
end
%Display the Markov matrix
fprintf('Matrix for Markov chain looks like this: \n'); M
%Get the function rho from the user:
fprintf('Enter function rho defind on each state of the Markov chain: \n')
Rho=zeros(1,D);
for i=1:D
fprintf('Rho %d = ',i);
Rho(i)=input(' ');
end
%Prob density
%Check if Poisson:
check=input('Enter 1 if Poisson distribution is used to model jumps; enter 2 otherwise: \n');
if check == 1
lambda=input('Enter lambda : \n');
tee = input('Enter the time to expiry, T: \n');
st='exp(-lambda*tee*k)*((lambda*tee*k)^k)/factorial(k)';
PK=inline(st,'k','lambda','tee');
end
if check == 2
fprintf('Enter P(v(T)=k) function :');
st = input(' : \n','s');
PK=inline(st,'k')
end
%Get whether this is a call or a put:
```

```
cp = input('Enter 1 for a call option; enter 2 for a put option: \n');
if cp == 1
f=inline('max(s-k,0)','k','s');
end
if cp ==2
f = inline('max(k-s,0)','k','s');
end
%Get the exercise price:
K=input('Enter the exercise price: \n');
%Get starting stock price from user:
SO = input('Enter the starting stock price: ');
%Get starting state from the user:
XO = input('Enter the starting state: ')
%get beta:
B = input('Enter the value of beta: \n');
%Get the interest rate, r:
r = input('Enter the interest rate r: ');
%Get iteration control count from user:
control = input('Enter the maximum amount of iterations: \n (NOTE: The recommended value is
20.) \n');
%get epsilon, the error control, from the user:
fprintf('Enter epsilon'); epsilono=input(' : \n');
%set up and start iteration
k=0;
if check==1
answer = f(SO,K)*PK(0,lambda,tee);
end
if check==2
answer = f(SO,K)*PK(0);
end
entry =answer;
k=1;
continu=1; %ensure that epsilon-condition is not yet satisfied:
while continu == 1
%simulate all paths of length k (for k jumps; k>=1):
veclist=zeros(D^k,k);
vect=ones(1,k);
for i=1:1:D^k
```

```
%adjust indices to not exceed number of markov states
for j=1:1:k
if ( vect(j)== D+1 ) && ( k >= 2 )
vect(j+1)=vect(j+1)+1;
vect(j)=1;
end
end
for m=1:1:k
veclist(i,m)=vect(m);
end
vect(1)=vect(1)+1;
end %%%veclist is now a matrix with rows corresponding to all possible k-length markov paths
%now calculate all probabilities based on these paths
summ=0;
%sum over all possible markov k-length paths
for i=1:1:D^k
%form value for each path
product=SO*(1+Rho(veclist(i,1)));
for j=2:1:k
product=product*(1+Rho(veclist(i,j)));
end
product=f(product,K);
for j=2:1:k
product=product*M(veclist(i,j-1),veclist(i,j));
end
summ=summ+product;
end
oldentry=entry; %store the previous entry as oldentry
%Return the entry for this k-value
if check == 1
entry=PK(k,lambda,tee)*summ;
end
if check == 2
entry=PK(k)*summ;
end
answer=answer+entry; %update the answer to greater degree of accuracy
%check to see whether to stop iterating
if (oldentry>0)&&(entry>0)&&(entry<epsilono)
```

```
continu=0
end
if k==control
continu=0
end
k=k+1;
end
if (k > control) && (diff >= epsilono)
fprintf('WARNING: TRUNCATION ERROR DUE TO EXCEEDING OF MAXIMUM ITERATION. \n ANSWER IS MOST LIKELY NOT ACCURATE TO INPUT EPSILON-VALUE. \n TRY INPUT OF HIGHER NUMBER OF ITEREATIONS.');
end
fprintf('Value of the option is: \n');
disp(answer);
fprintf('Number of iterations, i.e. k: \n');
disp(k-1);
```

B.6 MATLAB code ErgodicDiffusionApprox.m

```
%%%%%%%%%%%%%%%%%%%%%
% This MATLAB script, ErgodicDiffusionApprox.m, gives option prices
% for European and American options under the assumption of a GMRP
% asset price model with the Ergodic Diffusion Approximation,
% with an exponential distribution of jumps.
% Zachary Moyer
% University of Calgary
%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% User Info
fprintf('This MATLAB script, ErgodicDiffusionApprox.m, gives option prices \n for European and American options under the assumption of a GMRP \n asset price model with the Ergodic Diffusion Approximation, \n with an exponential distribution of jumps.');
%%%%%%%%%%%%%%%%%%%%%%
% User Input
D = input("\n Enter the number of states in the Markov chain: \n");
%Set up the Markov matrix
P = zeros(D,D);
for i = 1:1:D
```

```
for j = 1:1:D
fprintf("Prob to go from state %d to %d",i,j);
P(i,j) = input(" ");
end
end
%Display the Markov matrix
fprintf("\n Matrix for Markov chain looks like this: \n"); P
%Get the function rho from the user:
fprintf("\n Enter function rho defined on each state of the Markov chain: \n")
Rho=zeros(1,D);
for i=1:D
fprintf("Rho %d = ",i);
Rho(i)=input(" ");
end
Rho=Rho';
%Display the Rho vector
fprintf("\n The Rho vector looks like this: \n"); disp(Rho);
% Find lambda values for m:
mbar=zeros(D,1);
fprintf("\n Enter lambda values in order: \n");
for i=1:D;
fprintf("Lambda(x%d) = ",i);
mbar(i)=input(" ");
end
fprintf("\n Lambdas for each state are as follows: \n"); disp(mbar);
for i=1:D
mbar(i)=1/mbar(i);
end
fprintf("\n mbar vector is as follows: \n"); disp(mbar);
%%%%%%%%%%%%%%%%%%%%%
% Find limiting probabilities
a = [eye(size(P)(1))-P'; ones( 1,size(P)(1) )];
pe = a\[ zeros( size(P)(1) , 1); 1];
fprintf('\n Limiting probabilities for each state are: \n'); pe
pe=pe';
%%%%%%%%%%%%%%%%%%%%%
% Balance Condition check:
check = pe*P*Rho;
```

```
if abs(check)<=1e-10;
fprintf('\n The Balance Condition is fulfilled. \n');
end;
if abs(check)> 1e-10;
fprintf('\n The Balance Condition is NOT fulfilled. \n');
end;
%%%%%%%%%%%%%%%%%%%%%
% Calculate (and display) necessary variables
rhohat2 = (pe*P*Rho.^2)/(pe*mbar);
%Calculate limit matrix
pii=zeros(D);
for i=1:1:D;
pii(i,:)=pe;
end;
%Find optimal value to sum up to:
if max(max(abs(P^100-pii))) <= 1e-10;
maxsum=100;
elseif max(max(abs(P^1000-pii))) <= 1e-10;
maxsum=1000;
elseif max(max(abs(P^10000-pii))) <= 1e-10;
maxsum=10000;
else
maxsum=100000;
end
summ=zeros(D);
for i=1:1:maxsum;
summ=summ.+P^i-pii;
end
vecc=zeros(D,1);
for i=1:1:D
vecc(i)=P(i,:)*Rho;
end;
sigmarho2 = pe*((0.5*P*(Rho.^2) + (summ)*(vecc.^2) )/(pe*mbar));
%%%%%%%%%%%%%%%%%%%%%
% Display critical parameters to the user
fprintf(" \n Critical Parameters: \n");
fprintf(" Rhohat2 = %f \n",rhohat2);
fprintf(" Sigma_rho_2 squared = %f \n", sigmarho2);
```

```
%%%%%%%%%%%%%%%%%%%%%
while true;
% Print choice information for the user
fprintf(" \n \n Select what you wish to calculate. ");
fprintf("\n 1. Calculate values for European options ");
fprintf("\n 2. Calculate values for American options ");
fprintf("\n 3. Produce graphs of simulated approximate paths ");
fprintf("\n 4. Exit ");
selection = input("\n Selection: ");
if (selection==1); % Euro options task loop
%%%%%%%%%%%%%%%%%%%%%
% European options
fprintf("\n European option valuation ");
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the current stock price: ");
S = input(" ");
fprintf(" Input the exercise price: ");
E = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
sigma = sqrt(sigmarho2);
fprintf(" Enter tau (time to expiry, T-t): ");
tau = input(" ");
if (tau > 0);
d1 = (log(S/E) + (r + 0.5*sigma^2)*(tau))/(sigma*sqrt(tau));
d2 = d1 - sigma*sqrt(tau);
N1 = 0.5*(1+erf(d1/sqrt(2))); N2 = 0.5*(1+erf(d2/sqrt(2)));
C = S*N1-E*exp(-r*(tau))*N2; Cdelta = N1;
Pu = C + E*exp(-r*tau) - S; Pdelta = Cdelta - 1;
else
C = max(S-E,0); Cdelta = 0.5*(sign(S-E) + 1);
Pu = max(E-S,0); Pdelta = Cdelta - 1;
end
fprintf("\n European option information is below: ")
fprintf("\n European Call value: %f ", C);
fprintf("\n Delta of European Call: %f ", Cdelta);
fprintf("\n European Put value: %f ", Pu);
fprintf("\n Delta of European Put: %f ", Pdelta);
```

```
end % end task loop for Euro Options
if (selection==2); %Start task loop for American options
fprintf("\n American option valuation via the Binomial Method () and Black-Scholes");
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the current stock price: ");
S = input(" ");
fprintf(" Input the exercise price: ");
E = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
fprintf(" Enter the time to expiry: ");
T = input(" ");
fprintf("Enter M, the number of steps in the binomial calculation: ");
M=input(" ");
% Calculate for the American Put
sigma = sqrt(sigmarho2); dt = T/M;
u = exp(sigma*sqrt(dt) + (r-0.5*sigma^2)*dt);
d = exp(-sigma*sqrt(dt) + (r-0.5*sigma^2)*dt);
dpowers = d.^([M:-1:0]');
upowers = u.^([0:M]');
W = max(E-S*dpowers.*upowers,0);
p = 0.5;
% option value at time zero
for i = M:-1:1
Si = S*dpowers(M-i+2:M+1).*upowers(1:i);
W = max(max(E-Si,0),exp(-r*dt)*(p*W(2:i+1)+(1-p)*W(1:i)));
end
%for the American call
d1 = (log(S/E) + (r + 0.5*sigma^2)*(T))/(sigma*sqrt(T));
d2 = d1 - sigma*sqrt(T);
N1 = 0.5*(1+erf(d1/sqrt(2))); N2 = 0.5*(1+erf(d2/sqrt(2)));
C = S*N1-E*exp(-r*(T))*N2;
%display the info
fprintf("\n The value of an American put is: "); disp(W);
fprintf("\n The value of an American call is: "); disp(C);
end; % end task loop for American options
if (selection==3); %Start task loop for graphing simulated paths
fprintf("\n Draws dirfted and risk-neutral paths for S ");
```

```
fprintf("\n In figure 1 the blue are paths with calculated drift,");
fprintf("\n and the red paths represent the risk-neutral world. ")
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the starting stock price: ");
S = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
fprintf(" Enter the time to expiry: ");
T = input(" ");
fprintf("Enter the number of timesteps; recommend 100: ");
L=input(" ");
fprintf("Enter the number of paths you want to see: ");
M = input(" ");
% Draw paths
mu = rhohat2; sigma = sqrt(sigmarho2); dt = T/L;
%%%%%%%%%%%%%%%%%%%%%%%%
theta=(mu-r)/sigma;
tvals = [0:dt:T];
Svals = S*cumprod(exp((mu-0.5*sigma^2)*dt + sigma*(sqrt(dt)*(randn(M,L)))),2);
Svals = [S*ones(M,1) Svals]; % add initial asset price
Svals_adj = S*cumprod(exp((r-0.5*sigma^2)*dt + sigma*(sqrt(dt)*(randn(M,L)))),2);
Svals_adj = [S*ones(M,1) Svals_adj]; % add initial asset price
figure(1); clf;
plot(tvals,Svals,'b',tvals,Svals_adj,'r');
title('Sample Asset Paths: Unadjusted in Blue, Risk-Neutral in Red');
xlabel('t'); ylabel('S(t)');
end; % end task loop for graphing simulated paths
if (selection==4); % start break task loop
break
end % end break task loop
end % end while true loop
```

B.7 MATLAB code DoubleAveragedDiffusionApprox.m

```
%%%%%%%%%%%%%%%%%%%%%%%%
% This MATLAB script, DoubleAveragedDiffusionApprox.m, gives option prices
% for European and American options under the assumption of a GMRP
```

```
% asset price model with Double Averaged Diffusion,
% with an exponential distribution of jumps.
% Zachary Moyer
% University of Calgary
%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% User Info
fprintf('This MATLAB script, DoubleAveragedDiffusionApprox.m, gives option prices \n for European and American options under the assumption of a GMRP \n asset price model with the Double Averaged Diffusion, \n with an exponential distribution of jumps.');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
% User Input
D = input("\n Enter the number of states in the Markov chain: \n");
%Set up the Markov matrix
fprintf("\n Enter the Markov matrix in proper strongly diagonal form so as to identify merged chain by blocks: \n")
P = zeros(D,D);
for i = 1:1:D
for j = 1:1:D
fprintf("Prob to go from state %d to %d",i,j);
P(i,j) = input(" ");
end
end
%Display the Markov matrix
fprintf("\n Matrix for Markov chain looks like this: \n"); P
%Get the function rho from the user:
fprintf("\n Enter function rho defined on each state of the Markov chain: \n")
Rho=zeros(1,D);
for i=1:D
fprintf("Rho %d = ",i);
Rho(i)=input(" ");
end
Rho=Rho';
%Display the Rho vector
fprintf("\n The Rho vector looks like this: \n"); disp(Rho);
% Find lambda values for m:
mbar=zeros(D,1);
```

```
fprintf("\n Enter lambda values in order: \n");
for i=1:D
fprintf("Lambda(x%d) = ",i);
mbar(i)=input(" ");
end
fprintf("\n Lambdas for each state are as follows: \n"); disp(mbar);
for i=1:D
mbar(i)=1/mbar(i);
end
fprintf("\n mbar vector is as follows: \n"); disp(mbar);
%%%%%%%%%%%%%%%%%%%%
% Find limiting probabilities (large matrix)
a = [eye(size(P)(1))-P'; ones( 1,size(P)(1) )];
pe = a\[ zeros( size(P)(1) , 1); 1];
fprintf('\n Limiting probabilities for each state are as follows: \n'); disp(pe );
pe=pe';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
% classify by states of embedded Markov chain
fprintf("\n Enter the number of states in the embedded Markov chain: \n");
nstates=input(" ");
fprintf("\n Enter the block coordinates for each element of the embedded Markov chain, up to total
size: \n");
blockend=0;
for i=1:1:nstates
fprintf("End of block %d is: ",i);
blockend(i)=input(" ");
end;
rhohats = zeros(nstates,1);
sigmahats = zeros(nstates,1);
largprob=sum(pe(1:blockend(1)));
for i=2:1:nstates
largprob(i)=sum(pe(blockend(i-1)+1:blockend(i)));
end
%%%%%%%%%%%%%%%%%%%%%%%%
% Save original variables
PP=P;
DD=D;
```

```
RhoRho=Rho;
mbarmbar=mbar;
pepe=pe;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Grand loop over imbedded states
for k=1:1:nstates; % Calculate variables for each state in embedded Markov chain
% Obtain proper block information
if (k==1);
P=PP(1:blockend(1),1:blockend(1));
end;
if (k~=1);
P=PP(blockend(k-1)+1:blockend(k),blockend(k-1)+1:blockend(k));
end;
% convert the block to a proper probablity space
for i=1:1:size(P)(1);
P(i,:)=P(i,:)/sum(P(i,:));
end;
a=[eye(size(P)(1))-P'; ones( 1,size(P)(1) )];
r=a\[ zeros( size(P)(1) , 1); 1];
pe=r'; % calculate limiting probabilities in this imbedded state
if (k==1);
Rho=RhoRho(1:blockend(1));
mbar=mbarmbar(1:blockend(1));
end;
if (k~=1);
Rho=RhoRho(blockend(k-1)+1:blockend(k));
mbar=mbarmbar(blockend(k-1)+1:blockend(k));
end;
%%%%%%%%%%%%%%%%%%
rhohat = (pe*P*Rho.^2)/(pe*mbar);
%Calculate limit matrix
pii=zeros(size(P)(1));
for i=1:1:size(P)(1);
pii(i,:)=pe;
end;
%Find optimal value to sum up to:
if max(max(abs(P^100-pii))) <= 1e-10; maxsum=100;
```

```
elseif max(max(abs(P^1000-pii))) <= 1e-10; maxsum=1000;
elseif max(max(abs(P^10000-pii))) <= 1e-10; maxsum=10000;
else; maxsum=100000;
end;
summ=zeros(size(P)(1));
for i=1:1:maxsum;
summ=summ.+P^i-pii;
end;
vecc=zeros(size(P)(1),1);
for i=1:1:size(P)(1)
vecc(i)=P(i,:)*Rho;
end;
sigmahat = pe*((0.5*P*(Rho.^2) + (summ)*(vecc.^2) )/(pe*mbar));
%Update the variable vectors
rhohats(k) = rhohat;
sigmahats(k) = sigmahat;
% Check for balance condition for this state is met:
check = pe*P*Rho;
if abs(check)<=1e-10;
fprintf('\n The Balance Condition is fulfilled for state no %d of the embedded process. \n',k);
end;
if abs(check)> 1e-10;
fprintf('\n The Balance Condition is NOT fulfilled for state no %d of the embedded process. \n',k);
end;
end; % end calculate for each embedded state grand loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Display the necessary information
fprintf("\n Below are displayed the computed critical values for each state of the embedded Markov
chain: ");
fprintf("\n Embedded State #; rhohat; sigmahat squared ");
for i=1:1:nstates;
fprintf("\n %d %f %f",i,rhohats(i),sigmahats(i));
end;
%Calculate av_rhohat and av_sigmahat
av_rhohat = largprob*rhohats;
av_sigmahat = largprob*sigmahats;
fprintf("\n \n Below is the averaged values for the Markov chain: \n");
```

```
fprintf(" rhohat averaged; sigmahat squared averaged \n");
disp([av_rhohat, av_sigmahat]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% Other tasks
while true;
% Print choice information for the user
fprintf(" \n \n Select what you wish to calculate. ");
fprintf("\n 1. Calculate values for European options ");
fprintf("\n 2. Calculate values for American options ");
fprintf("\n 3. Produce graphs of simulated approximate paths ");
fprintf("\n 4. Exit ");
selection = input("\n Selection: ");
if (selection==1); % Euro options task loop
%%%%%%%%%%%%%%%%%%%%%
% European options
fprintf("\n European option valuation ");
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the current stock price: ");
S = input(" ");
fprintf(" Input the exercise price: ");
E = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
sigma = sqrt(av_sigmahat);
fprintf(" Enter tau (time to expiry, T-t): ");
tau = input(" ");
if (tau > 0);
d1 = (log(S/E) + (r + 0.5*sigma^2)*(tau))/(sigma*sqrt(tau));
d2 = d1 - sigma*sqrt(tau);
N1 = 0.5*(1+erf(d1/sqrt(2))); N2 = 0.5*(1+erf(d2/sqrt(2)));
C = S*N1-E*exp(-r*(tau))*N2; Cdelta = N1;
Pu = C + E*exp(-r*tau) - S; Pdelta = Cdelta - 1;
else
C = max(S-E,0); Cdelta = 0.5*(sign(S-E) + 1);
Pu = max(E-S,0); Pdelta = Cdelta - 1;
end
```

```
fprintf("\n European option information is below: ")
fprintf("\n European Call value: %f ", C);
fprintf("\n Delta of European Call: %f ", Cdelta);
fprintf("\n European Put value: %f ", Pu);
fprintf("\n Delta of European Put: %f ", Pdelta);
end % end task loop for Euro Options
if (selection==2); %Start task loop for American options
fprintf("\n American option valuation via the Binomial Method () and Black-Scholes");
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the current stock price: ");
S = input(" ");
fprintf(" Input the exercise price: ");
E = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
fprintf(" Enter the time to expiry: ");
T = input(" ");
fprintf("Enter M, the number of steps in the binomial calculation: ");
M=input(" ");
% Calculate for the American Put
sigma = sqrt(av_sigmahat); dt = T/M;
u = exp(sigma*sqrt(dt) + (r-0.5*sigma^2)*dt);
d = exp(-sigma*sqrt(dt) + (r-0.5*sigma^2)*dt);
dpowers = d.^([M:-1:0]');
upowers = u.^([0:M]');
W = max(E-S*dpowers.*upowers,0);
p = 0.5;
% option value at time zero
for i = M:-1:1
Si = S*dpowers(M-i+2:M+1).*upowers(1:i);
W = max(max(E-Si,0),exp(-r*dt)*(p*W(2:i+1)+(1-p)*W(1:i)));
end
%for the American call
d1 = (log(S/E) + (r + 0.5*sigma^2)*(T))/(sigma*sqrt(T));
d2 = d1 - sigma*sqrt(T);
N1 = 0.5*(1+erf(d1/sqrt(2))); N2 = 0.5*(1+erf(d2/sqrt(2)));
C = S*N1-E*exp(-r*(T))*N2;
%display the info
```

```
fprintf("\n The value of an American put is: "); disp(W);
fprintf("\n The value of an American call is: "); disp(C);
end; % end task loop for American options
if (selection==3); %Start task loop for graphing simulated paths
fprintf("\n Draws dirfted and risk-neutral paths for S ");
fprintf("\n In figure 1 the blue are paths with calculated drift,");
fprintf("\n and the red paths represent the risk-neutral world. ")
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the starting stock price: ");
S = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
fprintf(" Enter the time to expiry: ");
T = input(" ");
fprintf("Enter the number of timesteps; recommend 100: ");
L=input(" ");
fprintf("Enter the number of paths you want to see: ");
M = input(" ");
% Draw paths
mu = av_rhohat; sigma = sqrt(av_sigmahat); dt = T/L;
%%%%%%%%%%%%%%%%%%%%%%%%
theta=(mu-r)/sigma;
tvals = [0:dt:T];
Svals = S*cumprod(exp((mu-0.5*sigma^2)*dt + sigma*(sqrt(dt)*(randn(M,L)))),2);
Svals = [S*ones(M,1) Svals]; % add initial asset price
Svals_adj = S*cumprod(exp((r-0.5*sigma^2)*dt + sigma*(sqrt(dt)*(randn(M,L)))),2);
Svals_adj = [S*ones(M,1) Svals_adj]; % add initial asset price
figure(1); clf;
plot(tvals,Svals,'b',tvals,Svals_adj,'r');
title('Sample Asset Paths: Unadjusted in Blue, Risk-Neutral in Red');
xlabel('t'); ylabel('S(t)');
end; % end task loop for graphing simulated paths
if (selection==4); % start break task loop
break
end % end break task loop
end % end while true loop
```

B.8 MATLAB code ErgodicNormalDeviation.m

```
%%%%%%%%%%%%%%%%%%%%%%
% This MATLAB script, ErgodicNormalDeviation.m, gives option prices
% for European and American options under the assumption of a GMRP
% asset price model with the Ergodic Normal Deviations,
% with an exponential distribution of jumps.
% Zachary Moyer
% University of Calgary
%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% User Info
fprintf('This MATLAB script, ErgodicNormalDeviation.m, gives option prices \n for European and American options under the assumption of a GMRP \n asset price model with Ergodic Normal Deviations, \n with an exponential distribution of jumps.');
%%%%%%%%%%%%%%%%%%%%%%
% User Input
D = input("\n Enter the number of states in the Markov chain: \n");
%Set up the Markov matrix
P = zeros(D,D);
for i = 1:1:D
for j = 1:1:D
fprintf("Prob to go from state %d to %d",i,j);
P(i,j) = input(" ");
end
end
%Display the Markov matrix
fprintf("\n Matrix for Markov chain looks like this: \n"); P
%Get the function rho from the user:
fprintf("\n Enter function rho defined on each state of the Markov chain: \n")
Rho=zeros(1,D);
for i=1:D
fprintf("Rho %d = ",i);
Rho(i)=input(" ");
end
Rho=Rho';
%Display the Rho vector
fprintf("\n The Rho vector looks like this: \n"); disp(Rho);
```

```
% Find lambda values for m:
mbar=zeros(D,1);
fprintf("\n Enter lambda values in order: \n");
for i=1:D
fprintf("Lambda(x%d) = ",i);
mbar(i)=input(" ");
end
fprintf("\n Lambdas for each state are as follows: \n"); disp(mbar);
for i=1:D
mbar(i)=1/mbar(i);
end
fprintf("\n mbar vector is as follows: \n"); disp(mbar);
fprintf("\n Enter T: ");
Tee=input(" ");
%%%%%%%%%%%%%%%%%%%%
% Find limiting probabilities
a = [eye(size(P)(1))-P'; ones( 1,size(P)(1) )];
pe = a\[ zeros( size(P)(1) , 1); 1];
fprintf('\n Limiting probabilities for each state are: \n'); pe
pe=pe';
%%%%%%%%%%%%%%%%%%%%
% Calculate (and display) necessary variables
rhohat = (pe*P*Rho)/(pe*mbar);
rhohat = rhohat + 0.5*(1/Tee)*rhohat^2;
%Calculate limit matrix
pii=zeros(D);
for i=1:1:D;
pii(i,:)=pe;
end;
%Find optimal value to sum up to:
if max(max(abs(P^100-pii))) <= 1e-10;
maxsum=100;
elseif max(max(abs(P^1000-pii))) <= 1e-10;
maxsum=1000;
elseif max(max(abs(P^10000-pii))) <= 1e-10;
maxsum=10000;
else
maxsum=100000;
```

```
end
summ=zeros(D);
for i=1:1:maxsum;
summ=summ.+P^i-pii;
end
vecc=zeros(D,1);
for i=1:1:D
vecc(i)=P(i,:)*Rho-(pe*mbar)*rhohat;
end;
sigmahat = pe*(summ*(vecc.^2)+0.5*(vecc.^2))/(pe*mbar);
sigmahat = sigmahat*(1/Tee);
%%%%%%%%%%%%%%%%%%%%
% Display critical parameters to the user
fprintf(" \n%%%%%%%%%% Critical Parameters: %%%%%%%%%%\n");
fprintf(" Rhohat = %f \n",rhohat);
fprintf(" Sigma_hat squared = %f \n", sigmahat);
%%%%%%%%%%%%%%%%%%%%
while true;
% Print choice information for the user
fprintf(" \n \n Select what you wish to calculate. ");
fprintf("\n 1. Calculate values for European options ");
fprintf("\n 2. Calculate values for American options ");
fprintf("\n 3. Produce graphs of simulated approximate paths ");
fprintf("\n 4. Exit ");
selection = input("\n Selection: ");
if (selection==1); % Euro options task loop
%%%%%%%%%%%%%%%%%%%%
% European options
fprintf("\n European option valuation ");
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the current stock price: ");
S = input(" ");
fprintf(" Input the exercise price: ");
E = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
sigma = sqrt(sigmahat);
fprintf(" Enter tau (time to expiry, T-t): ");
```

```
tau = input(" ");
if (tau > 0);
d1 = (log(S/E) + (r + 0.5*sigma^2)*(tau))/(sigma*sqrt(tau));
d2 = d1 - sigma*sqrt(tau);
N1 = 0.5*(1+erf(d1/sqrt(2))); N2 = 0.5*(1+erf(d2/sqrt(2)));
C = S*N1-E*exp(-r*(tau))*N2; Cdelta = N1;
P = C + E*exp(-r*tau) - S; Pdelta = Cdelta - 1;
else
C = max(S-E,0); Cdelta = 0.5*(sign(S-E) + 1);
P = max(E-S,0); Pdelta = Cdelta - 1;
end
fprintf("\n European option information is below: ")
fprintf("\n European Call value: %f ", C);
fprintf("\n Delta of European Call: %f ", Cdelta);
fprintf("\n European Put value: %f ", P);
fprintf("\n Delta of European Put: %f ", Pdelta);
end % end task loop for Euro Options
if (selection==2); %Start task loop for American options
fprintf("\n American option valuation via the Binomial Method () and Black-Scholes");
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the current stock price: ");
S = input(" ");
fprintf(" Input the exercise price: ");
E = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
fprintf(" Enter the time to expiry: ");
T = input(" ");
fprintf("Enter M, the number of steps in the binomial calculation: ");
M=input(" ");
% Calculate for the American Put
sigma = sqrt(sigmahat); dt = T/M;
u = exp(sigma*sqrt(dt) + (r-0.5*sigma^2)*dt);
d = exp(-sigma*sqrt(dt) + (r-0.5*sigma^2)*dt);
dpowers = d.^([M:-1:0]');
upowers = u.^([0:M]');
W = max(E-S*dpowers.*upowers,0);
p = 0.5;
```

```
% option value at time zero
for i = M:-1:1
Si = S*dpowers(M-i+2:M+1).*upowers(1:i);
W = max(max(E-Si,0),exp(-r*dt)*(p*W(2:i+1)+(1-p)*W(1:i)));
end
%for the American call
d1 = (log(S/E) + (r + 0.5*sigma^2)*(T))/(sigma*sqrt(T));
d2 = d1 - sigma*sqrt(T);
N1 = 0.5*(1+erf(d1/sqrt(2))); N2 = 0.5*(1+erf(d2/sqrt(2)));
C = S*N1-E*exp(-r*(T))*N2;
%display the info
fprintf("\n The value of an American put is: "); disp(W);
fprintf("\n The value of an American call is: "); disp(C);
end; % end task loop for American options
if (selection==3); %Start task loop for graphing simulated paths
fprintf("\n Draws dirfted and risk-neutral paths for S ");
fprintf("\n In figure 1 the blue are paths with calculated drift,");
fprintf("\n and the red paths represent the risk-neutral world. ")
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the starting stock price: ");
S = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
fprintf(" Enter the time to expiry: ");
T = input(" ");
fprintf("Enter the number of timesteps; recommend 100: ");
L=input(" ");
fprintf("Enter the number of paths you want to see: ");
M = input(" ");
% Draw paths
mu = rhohat; sigma = sqrt(sigmahat); dt = T/L;
%%%%%%%%%%%%%%%%%%%%%%%%
theta=(mu-r)/sigma;
tvals = [0:dt:T];
Svals = S*cumprod(exp((mu-0.5*sigma^2)*dt + sigma*(sqrt(dt)*(randn(M,L)))),2);
Svals = [S*ones(M,1) Svals]; % add initial asset price
Svals_adj = S*cumprod(exp((r-0.5*sigma^2)*dt + sigma*(sqrt(dt)*(randn(M,L)))),2);
Svals_adj = [S*ones(M,1) Svals_adj]; % add initial asset price
```

```
figure(1); clf;
plot(tvals,Svals,'b',tvals,Svals_adj,'r');
title('Sample Asset Paths: Unadjusted in Blue, Risk-Neutral in Red');
xlabel('t'); ylabel('S(t)');
end; % end task loop for graphing simulated paths
if (selection==4); % start break task loop
break
end % end break task loop
end % end while true loop
```

B.9 MATLAB code DoubleAveragedNormalDeviation.m

```
%%%%%%%%%%%%%%%%%%%%%%
% This MATLAB script, DoubleAveragedNormalDeviation.m, gives option prices
% for European and American options under the assumption of a GMRP
% asset price model with Double Averaged Normal
% Deviations, with an exponential distribution of jumps.
% Zachary Moyer
% University of Calgary
%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% User Info
fprintf('This MATLAB script, DoubleAveragedNormalDeviation.m, gives option prices \n for European and American options under the assumption of a GMRP \n asset price model with the Double Averaged Normal \n Deviations, with an exponential distribution of jumps.');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% User Input
D = input("\n Enter the number of states in the Markov chain: \n");
%Set up the Markov matrix
fprintf("\n Enter the Markov matrix in proper strongly diagonal form so as to identify merged chain by blocks: \n")
P = zeros(D,D);
for i = 1:1:D
for j = 1:1:D
fprintf("Prob to go from state %d to %d",i,j);
P(i,j) = input(" ");
end
```

```
end
%Display the Markov matrix
fprintf("\n Matrix for Markov chain looks like this: \n"); P
%Get the function rho from the user:
fprintf("\n Enter function rho defined on each state of the Markov chain: \n")
Rho=zeros(1,D);
for i=1:D
fprintf("Rho %d = ",i);
Rho(i)=input(" ");
end
Rho=Rho';
%Display the Rho vector
fprintf("\n The Rho vector looks like this: \n"); disp(Rho);
% Find lambda values for m:
mbar=zeros(D,1);
fprintf("\n Enter lambda values in order: \n");
for i=1:D
fprintf("Lambda(x%d) = ",i);
mbar(i)=input(" ");
end
fprintf("\n Lambdas for each state are as follows: \n"); disp(mbar);
for i=1:D
mbar(i)=1/mbar(i);
end
fprintf("\n mbar vector is as follows: \n"); disp(mbar);
% take t as input
fprintf("\n mbar vector is as follows: \n"); disp(mbar);
fprintf("\n Enter T: ");
Tee=input(" ");
%%%%%%%%%%%%%%%%%%%%
% Find limiting probabilities (large matrix)
a = [eye(size(P)(1))-P'; ones( 1,size(P)(1) )];
pe = a\[ zeros( size(P)(1) , 1); 1];
fprintf('\n Limiting probabilities for each state are as follows: \n'); disp(pe );
pe=pe';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% classify by states of embedded Markov chain
```

```
fprintf("\n Enter the number of states in the embedded Markov chain: \n");
nstates=input(" ");
fprintf("\n Enter the block coordinates for each element of the embedded Markov chain, up to total size: \n");
blockend=0;
for i=1:1:nstates
fprintf("End of block %d is: ",i);
blockend(i)=input(" ");
end;
rhohats = zeros(nstates,1);
sigmahats = zeros(nstates,1);
largprob=sum(pe(1:blockend(1)));
for i=2:1:nstates
largprob(i)=sum(pe(blockend(i-1)+1:blockend(i)));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Save original variables
PP=P;
DD=D;
RhoRho=Rho;
mbarmbar=mbar;
pepe=pe;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Grand loop over imbedded states
for k=1:1:nstates; % Calculate variables for each state in embedded Markov chain
% Obtain proper block information
if (k==1);
P=PP(1:blockend(1),1:blockend(1));
end;
if (k~=1);
P=PP(blockend(k-1)+1:blockend(k),blockend(k-1)+1:blockend(k));
end;
% convert the block to a proper probablity space
for i=1:1:size(P)(1);
P(i,:)=P(i,:)/sum(P(i,:));
end;
a=[eye(size(P)(1))-P'; ones( 1,size(P)(1) )];
```

```
r=a\[ zeros( size(P)(1) , 1); 1];
pe=r'; % calculate limiting probabilities in this imbedded state
if (k==1);
Rho=RhoRho(1:blockend(1));
mbar=mbarmbar(1:blockend(1));
end;
if (k~=1);
Rho=RhoRho(blockend(k-1)+1:blockend(k));
mbar=mbarmbar(blockend(k-1)+1:blockend(k));
end;
%%%%%%%%%%%%%%%%%%
rhohat = (pe*P*Rho)/(pe*mbar);
rhohat = rhohat + 0.5*(1/Tee)*rhohat^2;
%Calculate limit matrix
pii=zeros(size(P)(1));
for i=1:1:size(P)(1);
pii(i,:)=pe;
end;
%Find optimal value to sum up to:
if max(max(abs(P^100-pii))) <= 1e-10; maxsum=100;
elseif max(max(abs(P^1000-pii))) <= 1e-10; maxsum=1000;
elseif max(max(abs(P^10000-pii))) <= 1e-10; maxsum=10000;
else; maxsum=100000;
end;
summ=zeros(size(P)(1));
for i=1:1:maxsum; summ=summ.+P^i-pii; end;
vecc=zeros(size(P)(1),1);
for i=1:1:size(P)(1)
vecc(i)=P(i,:)*Rho-(pe*mbar)*rhohat;
end;
sigmahat = pe*(summ*(vecc.^2)+0.5*(vecc.^2))/(pe*mbar);
sigmahat = sigmahat*(1/Tee);
%Update the variable vectors
rhohats(k) = rhohat;
sigmahats(k) = sigmahat;
end; % end calculate for each embedded state grand loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Display the necessary information
fprintf("\n Below are displayed the computed critical values for each state of the embedded Markov chain: ");
fprintf("\n Embedded State #; rhohat; sigmahat squared ");
for i=1:1:nstates;
fprintf("\n %d %f %f",i,rhohats(i),sigmahats(i));
end;
%Calculate av_rhohat and av_sigmahat
av_rhohat = largprob*rhohats;
av_sigmahat = largprob*sigmahats;
fprintf("\n \n Below is the averaged values for the Markov chai:n \n");
fprintf(" rhohat averaged; sigmahat squared averaged \n");
disp([av_rhohat, av_sigmahat]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
% Other tasks
while true;
% Print choice information for the user
fprintf(" \n \n Select what you wish to calculate. ");
fprintf("\n 1. Calculate values for European options ");
fprintf("\n 2. Calculate values for American options ");
fprintf("\n 3. Produce graphs of simulated approximate paths ");
fprintf("\n 4. Exit ");
selection = input("\n Selection: ");
if (selection==1); % Euro options task loop
%%%%%%%%%%%%%%%%%%%%%%%%%
% European options
fprintf("\n European option valuation ");
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the current stock price: ");
S = input(" ");
fprintf(" Input the exercise price: ");
E = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
sigma = sqrt(av_sigmahat);
fprintf(" Enter tau (time to expiry, T-t): ");
```

```
tau = input(" ");
if (tau > 0);
d1 = (log(S/E) + (r + 0.5*sigma^2)*(tau))/(sigma*sqrt(tau));
d2 = d1 - sigma*sqrt(tau);
N1 = 0.5*(1+erf(d1/sqrt(2))); N2 = 0.5*(1+erf(d2/sqrt(2)));
C = S*N1-E*exp(-r*(tau))*N2; Cdelta = N1;
Pu = C + E*exp(-r*tau) - S; Pdelta = Cdelta - 1;
else
C = max(S-E,0); Cdelta = 0.5*(sign(S-E) + 1);
Pu = max(E-S,0); Pdelta = Cdelta - 1;
end
fprintf("\n European option information is below: ")
fprintf("\n European Call value: %f ", C);
fprintf("\n Delta of European Call: %f ", Cdelta);
fprintf("\n European Put value: %f ", Pu);
fprintf("\n Delta of European Put: %f ", Pdelta);
end % end task loop for Euro Options
if (selection==2); %Start task loop for American options
fprintf("\n American option valuation via the Binomial Method () and Black-Scholes");
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the current stock price: ");
S = input(" ");
fprintf(" Input the exercise price: ");
E = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
fprintf(" Enter the time to expiry: ");
T = input(" ");
fprintf("Enter M, the number of steps in the binomial calculation: ");
M=input(" ");
% Calculate for the American Put
sigma = sqrt(av_sigmahat); dt = T/M;
u = exp(sigma*sqrt(dt) + (r-0.5*sigma^2)*dt);
d = exp(-sigma*sqrt(dt) + (r-0.5*sigma^2)*dt);
dpowers = d.^([M:-1:0]');
upowers = u.^([0:M]');
W = max(E-S*dpowers.*upowers,0);
p = 0.5;
```

```
% option value at time zero
for i = M:-1:1
Si = S*dpowers(M-i+2:M+1).*upowers(1:i);
W = max(max(E-Si,0),exp(-r*dt)*(p*W(2:i+1)+(1-p)*W(1:i)));
end
%for the American call
d1 = (log(S/E) + (r + 0.5*sigma^2)*(T))/(sigma*sqrt(T));
d2 = d1 - sigma*sqrt(T);
N1 = 0.5*(1+erf(d1/sqrt(2))); N2 = 0.5*(1+erf(d2/sqrt(2)));
C = S*N1-E*exp(-r*(T))*N2;
%display the info
fprintf("\n The value of an American put is: "); disp(W);
fprintf("\n The value of an American call is: "); disp(C);
end; % end task loop for American options
if (selection==3); %Start task loop for graphing simulated paths
fprintf("\n Draws dirfted and risk-neutral paths for S ");
fprintf("\n In figure 1 the blue are paths with calculated drift,");
fprintf("\n and the red paths represent the risk-neutral world. ")
fprintf("\n Please enter necessary parameters below.");
fprintf("\n Enter the starting stock price: ");
S = input(" ");
fprintf(" Input the interest rate r: ");
r = input(" ");
fprintf(" Enter the time to expiry: ");
T = input(" ");
fprintf("Enter the number of timesteps; recommend 100: ");
L=input(" ");
fprintf("Enter the number of paths you want to see: ");
M = input(" ");
% Draw paths
mu = av_rhohat; sigma = sqrt(av_sigmahat); dt = T/L;
%%%%%%%%%%%%%%%%%%%%%%%%%
theta=(mu-r)/sigma;
tvals = [0:dt:T];
Svals = S*cumprod(exp((mu-0.5*sigma^2)*dt + sigma*(sqrt(dt)*(randn(M,L)))),2);
Svals = [S*ones(M,1) Svals]; % add initial asset price
Svals_adj = S*cumprod(exp((r-0.5*sigma^2)*dt + sigma*(sqrt(dt)*(randn(M,L)))),2);
Svals_adj = [S*ones(M,1) Svals_adj]; % add initial asset price
```

```
figure(1); clf;
plot(tvals,Svals,'b',tvals,Svals_adj,'r');
title('Sample Asset Paths: Unadjusted in Blue, Risk-Neutral in Red');
xlabel('t'); ylabel('S(t)');
end; % end task loop for graphing simulated paths
if (selection==4); % start break task loop
break
end % end break task loop
end % end while true loop
```