

https://dspace.org/

Reinforcement learning applied to option pricing

Martin, K. S.

2014-09-01
http://hdl.handle.net/10539/15333

Downloaded from a DSpace repository. https://dspace.org/

Reinforcement Learning Applied to Option Pricing

K.S. Martin

MSc in Computer Science,
University of the Witwatersrand,
Johannesburg.



June 12, 2014

Declaration

I declare that this project is my own, unaided work. It is being submitted for the degree of Master of Science (Computer Science) in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

_____

June 12, 2014

Abstract

This dissertation considers the pricing of European and American options. European option prices are determined by the market and can be verified by a closed-form solution to the Black-Scholes model. These options can only be exercised at the maturity date. American option prices are not derived from the market and cannot be priced using the same closed-form solution as in the case of the European options because American options can be exercised at any time on or before the maturity date. An initial method was investigated in pricing a European option but could not price American options. Improvements were made producing two robust option pricing models. The results of which were compared to the closed-form solution in the case of European options and a numerical approximation solution in the case of American options. The improved models showed two significant benefits. The first benefit is the ability to price both European and American options and the second is the ability to calibrate the models to market prices using market data. Changes to the parameters of the models showed the limitations of each improved model. In conclusion, the improved methods are effective procedures for solving the European and American option pricing problem.

Keywords: European options, American options, Markov Decision Processes, Kernel-Based Reinforcement Learning, Calibration.

Acknowledgements

I would like to express my sincere appreciation to my supervisors, Professor Clint van Alten and Pravesh Ranchod, for their tireless assistance and encouragement. I would also like to thank my family who provided support and encouragement throughout this research.

Contents

1 Introduction **1**
- 1.1 History of Options . . . 1
- 1.2 Brief introduction to financial options . . . 2
  - 1.2.1 Desirability of options . . . 2
  - 1.2.2 Importance of the premium for an option . . . 2
- 1.3 Revealing the research problem . . . 3
  - 1.3.1 The problem . . . 3
  - 1.3.2 The solution . . . 3
- 1.4 Structure of the dissertation . . . 4

2 Core Concepts **5**
- 2.1 Finance Preliminaries . . . 5
  - 2.1.1 Options and Option Types . . . 5
  - 2.1.2 American Options . . . 6
  - 2.1.3 Black-Scholes . . . 7
  - 2.1.4 Monte Carlo Simulation . . . 10
- 2.2 Markov Decision Processes (MDPs) . . . 15
  - 2.2.1 Definitions . . . 15
- 2.3 Reinforcement Learning (RL) . . . 16
  - 2.3.1 Value Iteration . . . 17
  - 2.3.2 Kernel-Based Reinforcement Learning . . . 18
- 2.4 Summary . . . 21

3 European options **23**
- 3.1 Initial Trajectory model . . . 23
  - 3.1.1 Constructing an approximate KBRL MDP . . . 25
  - 3.1.2 Transition Probability refinement . . . 32
  - 3.1.3 Experiment (initial Trajectory model): Pricing a European put option . . . 36
- 3.2 Trajectory model using separated features and only hold action 38
  - 3.2.1 Removing sell action . . . 38
  - 3.2.2 Separating Risk/Time Component . . . 39

3.2.3 Experiment (Trajectory model using separated features and only hold action): Pricing a European call option . . . 42
3.3 Mapped model using separated features and only hold action . . 44
3.3.1 Mapping the state space . . . 46
3.3.2 Kernel function amendment . . . 47
3.3.3 Generating Transition Probabilities . . . 49
3.3.4 Experiment (Mapped model using separated features and only hold action): Pricing a European call . . . 50
3.4 Value Iteration procedure to obtain the value function and an optimal policy for the Option Pricing KBRL MDP . . . 53
3.5 Calculating a price for unseen option . . . 56
3.6 Summary . . . 57

4 American options **58**
4.1 Changes to European option MDP . . . 59
4.2 Trajectory model using separated features . . . 60
4.2.1 Experiment (Trajectory model using separated features): Pricing an American call option . . . 60
4.3 Experiment (Mapped model): Pricing an American call . . . 62
4.4 Analysis of execution time . . . 64
4.5 Pricing an American call option on a dividend-paying asset . . . 65
4.5.1 Experiment (Trajectory model using separated features): Pricing an American call option on a dividend-paying asset . . . 65
4.5.2 Experiment (Mapped model): Pricing an American call option on a dividend-paying asset . . . 68
4.6 Summary . . . 70

5 Calibrating models **71**
5.1 Calibrating improved trajectory model using underlying stock trajectories . . . 71
5.1.1 Replacing simulated Market State trajectories . . . 73
5.1.2 Experiment (calibrating the improved Trajectory model): Pricing a European call and an American call . . . 75
5.2 Calibrating Mapped model using Option data . . . 79
5.2.1 Experiment (calibrating Mapped model): Pricing a European call and an American call . . . 79
5.2.2 Comparison to market data prices . . . 84
5.3 Summary . . . 85

6 Effect of changing KBRL parameters **87**
6.1 Standard deviation of kernels (b value) 87
6.2 Summary 89

7 Conclusion **92**
7.1 Significance of research 92
7.2 Research Findings 93
7.3 Future Research 93

Bibliography **95**

A Pseudocode **98**

B Market Data **101**
B.1 Stock options data 101
B.2 Interest Rate Zero Curve (USD) 101

C Graphs **106**

D Tables **112**

List of Figures

2.1 European put & call price matrices (analytic Black-Scholes) . . 9
2.2 Monte-Carlo European put price matrix (Black-Scholes) 12
2.3 Monte-Carlo American call price matrix (Black-Scholes) 14
2.4 Value Iteration Algorithm via V-update 18
2.5 Illustration of transition probability function using a Gaussian kernel on a Euclidean distance measure 21

3.1 Derivative sampling 26
3.2 Market State sampling 29
3.3 Transition Probability calculation 30
3.4 Kernel function 31
3.5 Gaussian kernels 33
3.6 Transition Probabilities from high risk 34
3.7 Transition Probabilities from low risk 35
3.8 European put price matrix (KBRL-derived model) 37
3.9 Market State sampling 41
3.10 European call price matrix (KBRL with no sell action) 43
3.11 Market States mapping (1000 trajectories) 44
3.12 Trajectory model (only hold action) stock space mapping 45
3.13 Trajectory model (only hold action) stock space mapping 45
3.14 3-dimensional state space mapping) 46
3.15 Procedure to generate state space mapping 47
3.16 Stock vs Time-to-maturity - mapped model 48
3.17 Stock vs Volatility - mapped model 48
3.18 Volatility versus Time-to-maturity - mapped model 49
3.19 Lognormal Strike-Relative stock distribution 50
3.20 Mapped state space Transition Probability calculation 51
3.21 European call price matrix (KBRL with Mapped state space) . 52
3.22 Reward function 54
3.23 Value Iteration 55
3.24 Determing price of previously unseen option 56

4.1 American call price matrix (KBRL with no sell action) 61

4.2 American call price matrix (KBRL with Mapped state space) . 63
4.3 Execution times graph (in seconds) 65
4.4 American call price matrix (KBRL with no sell action) 67
4.5 American call price matrix (KBRL with Mapped state space) . 69

5.1 Procedure to calculate Historical Volatility 73
5.2 Market data stock state space mapping 74
5.3 Volatility vs. Time-to-maturity 75
5.4 Calibrated European call price matrix (trajectory approach) . . 77
5.5 Calibrated American call price matrix (trajectory approach) . . 78
5.6 Market Data stock state space mapping 80
5.7 Volatility vs. Time-to-maturity 81
5.8 Calibrated European call 82
5.9 Calibrated American call 83
5.10 Consistency of Market Data prices 85

6.1 Bumpy European call price matrix (KBRL with Mapped state space) 88
6.2 Smooth European call price matrix (KBRL with Mapped state space) 90
6.3 Non-Zero flat European call price matrix (KBRL with Mapped state space) 91

A.1 Algorithm to generate Black-Scholes stock price paths 99
A.2 Monte Carlo Pricing model 100

C.1 Black-Scholes stock price paths 106
C.2 Market States mapping (10 trajectories) 107
C.3 Market States mapping (100 trajectories) 108
C.4 Market States mapping (10000 trajectories) 109
C.5 Stock Gaussian Transition Probabilities 110
C.6 Risk biased Gaussian Transition Probabilities 111

List of Tables

4.1 Execution times table (in seconds) 64

5.1 List of companies for which market data was extracted from Yahoo Finance 72
5.2 Maturity dates of European call options from 8-Jan-2013 72

B.1 Excerpt Market Data of MSFT stock options for 8-Jan-2013 . . 102
B.2 Market Data for USD swap rate raw data 102
B.3 Market Data for stripped USD swap curve 103
B.4 Dividend payouts per stock 104
B.5 Dividend payouts per stock (continued) 105

D.1 Basic kernel functions and the corresponding extended parameterizations 112

Chapter 1

Introduction

1.1 History of Options

The very first option in financial history dates back to Aristotle's book on Politics from Book 1 Chapter 11[6]. Aristotle describes a story about Thales (a Greek astrologer and mathematician) in which he could observe the position of the stars in winter and predict an extraordinary harvest of olives in the following year. Based on this knowledge, Thales negotiated and bought contracts from all the olive press owners in the area giving him the right to lease the olive presses to the farmers at harvest time. He agreed to the leasing prices but made no upfront payment for the lease so if the harvest prediction turned out to be wrong, he would not have to lease the presses and would only have lost the prepayment of the contract to the olive presses. As harvest time arrived, his forecasts were true, so Thales was able to demand almost any price from the olive farmers because he had leased all the presses from the areas. The idea of paying in advance for the optionality in a contract is still the basic idea in modern option trading.

During the 17th century options had acquired a bad reputation because of the "Tulip Bulb Mania" in Holland. Tulips were popular among the Dutch aristocracy as a status symbol when their popularity emerged into a worldwide market, so the prices for tulips went up drastically. In case of a bad harvest, tulip wholesalers began buying call options while tulip farmers bought put options[1]. As the price of tulip bulbs increased, the value of existing option contracts increased causing families to use their entire fortune to speculate on

[1]Call and put options are explained in Chapter 2

the tulip bulb market. In 1638, chaos hit and the prices of tulips plummeted. Despite the Dutch government's efforts to force speculators to fulfil their obligations on their options, most speculators were either unable or unwilling to make good on their contracts.

1.2 Brief introduction to financial options

Options are financial contracts between two parties, the buyer and the seller, and the price of options is dependent on the price of some other investment instrument or commodity which is known as the "underlying". The underlying investment instrument can be a tangible object such as beans or it could be the share of a company. The buyer or holder of an option has the right but not the obligation to buy or sell a prespecified quantity of the underlying at a prespecified price[17]. Two types of options are focused on in this research; exercising allowed during the whole contract period (American style) and exercising is only possible at the final expiration date (European style).

1.2.1 Desirability of options

The buyer of an option has bought the option so that he can either speculate or hedge. In the former case the holder has an anticipation of the future development of the market and takes on a risky position. In the latter case the holder wants to cover his risky position against undesirable events. The hedger is opposite to the speculator since he wants to mitigate the effect of losses and abandon possible gains in his position. The seller of an option agrees to an option contract because of the possibility to earn the premium of the option that the buyer has to pay for his right. One can compare the motivation of the olive press owners in the story of Thales to the basic idea that underlies option contracts. When the option is not exercised by the buyer, the seller earns the premium as final payoff.

1.2.2 Importance of the premium for an option

One of the most important factors in an option contract is the premium at which the contract is settled. It represents the price the buyer has to pay for the right to exercise the option. The problem is to compute a fair premium which

takes into consideration all the relevant factors. These include the current price of the underlying asset, the underlying asset's volatility, the strike price of the option, the time to maturity and the risk free interest rate. An option's premium can be split into two main factors: the time value and the intrinsic value of the option. The latter defines the actual value of the option without considering the time value of money. The former defines the premium a rational investor would pay based on the probability the value of the option will increase before expiry.

1.3 Revealing the research problem

1.3.1 The problem

The main problem is to efficiently price options in a way that is both tractable and plausible. Existing numerical solution procedures are time consuming because simulations need to be generated for each option that requires a price. Pricing one hundred options is no problem, however pricing six thousand options is a huge problem. Risk numbers inform a trader on how risky his position is in an option trade and multiple valuations of an option are needed to obtain each risk number. Decisions need to be made based on these risk numbers on whether to continue trading or to terminate his position in the trade. Making the wrong decision can prove fatal in that the trader may incur a massive loss! Therefore, the trader requires quick and accurate pricing models to mitigate the risk of loss in option trading.

1.3.2 The solution

Existing numerical methods such as Monte Carlo simulation require the construction of a model for each option. This thesis will use Kernel-Based Reinforcement methods, which are able to construct a global, reusable pricing model. Kernel-Based Reinforcement Learning (KBRL) is able to learn a pricing model from simulated or market data and can obtain a price for any new option by using the model.

An initial KBRL method was investigated in pricing European options but could not price American options. Improvements were made producing two robust option pricing models. The results of these two models were compared

to the closed-form solution in the case of European options and a numerical approximation solution in the case of American options. The improved models showed two significant benefits. The first benefit is the ability to price both European and American options and the second is the ability to calibrate the models to market prices using market data. By using the KBRL approach, the time taken to obtain a price for an American option was reduced to a quarter of the time of the commonly used numerical solutions. However, changes to the parameters of these two models showed the limitations of each model.

1.4 Structure of the dissertation

Core concepts of the research are introduced in chapter two dealing with finance preliminaries, Markov Decision Processes and Reinforcement Learning. The purpose of the finance preliminaries is to briefly describe options, the mathematical models used in finance and the ways of pricing European and American options. The rest of the chapter introduces the theory of Reinforcement Learning.

Chapter three is dedicated to the formulation of progressive attempts in solving the European option pricing problem. An essential theorem is proven indicating that the problem can be solved *theoretically* within the research. This chapter makes extensive use of the aspects described in chapter two. Experiments of each of the models are shown illustrating the effectiveness of each model.

Having established a theoretical framework for solving the European option pricing problem, chapter four describes how the American option pricing problem can be solved by manipulating the existing European option pricing framework. The execution times of the KBRL models were compared to the Monte Carlo execution times. These results showed the former models performed significantly better over the latter, time-consuming approach. Chapter five explains how market data is used to calibrate the European and American options to market prices.

The limitations of the models are provided in chapter six showing the lower and upper bounds for the option prices. Chapter seven concludes the dissertation.

Chapter 2

Core Concepts

This chapter introduces the types of options traded in the market and various mathematical models that can be used to obtain prices for exotic options (i.e. options that do not have prices visible in the market). A basic introduction to Markov Decision Processes (MDPs) is provided since the first objective of the dissertation is to transform the option pricing problem into an MDP. Reinforcement Learning is an area of Artificial Intelligence mainly focused on solving real-world problems. In this research, Kernel-Based Reinforcement Learning will be used to solve for the European and American option pricing problem.

2.1 Finance Preliminaries

2.1.1 Options and Option Types

A derivative is a financial instrument whose value is derived from one or more underlying assets, market securities or indices[17]. Essentially, a derivative is a contract between two parties specifying certain conditions, such as the date either party may receive or pay cash on, the resulting cash they may receive or pay on that date, the definitions of the underlying variables, the parties' contractual obligations and the size of the contract (commonly called the notional amount).

Options are one of the most common derivative types traded in the market. There are two types of options: A *call option* is a contract that gives the holder the right to buy an underlying asset at a predetermined price on or before a

specified date in the future[17]. A *put option* is a contract that gives the holder the right to sell an underlying asset at a predetermined price on or before a specified date in the future[17]. The predetermined price in the option contract is the *exercise price* or *strike price*. The intrinsic value (i.e. the payoff) of a call option is calculated as the maximum between the difference in the current stock price and the strike price and zero (i.e. $f(U_t) = \max\{U_t - K, 0\}$ where U_t is the current stock price and K is the strike price). The date in the contract is the maturity or expiration date and European options can only be exercised on the expiration date, while American options can be exercised at any time up to the expiration date.

For example, consider a European put option on XYZ stock: The European put option gives the right to *sell* one share of XYZ stock for R1 in three months' time. Suppose that today's XYZ stock price is R1.10. The R1 is called the strike price (K), the date in three months' time is called maturity (T) and the XYZ stock on which the option is based is called the underlying asset ($U_0 =$ R1.10 is the initial stock price). The European put option would be exercised at maturity if the stock price is below the strike price because the option is worth

$$f(U_T) = \max\{K - U_T, 0\}$$

at the maturity of the option. This function, $f(U_t)$ for $0 \leq t < T$, is called the payoff function and the max represents the optionality. The fair value of an option in the risk-neutral world is the present value of the expected payoff at maturity under a risk-neutral probability measure and is given by

$$P_t = \exp(-rT)\mathbb{E}\left[f(U_T)\right]$$

where r is the riskless interest rate gathered from the market, T is the maturity date of the contract and P_t is the price of the put option at time t. The function $\mathbb{E}\left[\cdot\right]$ defines the expectation (or mean value) of an unknown random event. The payoff function is denoted by $f\left(\cdot\right)$ with U_T denoting the stock price at maturity, which is an unknown random event. The European option price is nothing more than the expected, discounted payoff at maturity.

2.1.2 American Options

American options are contracts that may be exercised at any time prior to maturity. The right to exercise at any time is clearly valuable and the price,

or the fair value, of an American option cannot be less than the equivalent European option[17]. Even though this grants more rights to the holder, the problem is to know the optimal time to exercise the option. This makes American options much more interesting than its European counterparts because the American option value is maximized by an optimal exercise strategy.

The American option pricing problem can be defined as follows

$$P_t = \sup_{\tau \in [t,T]} \mathbb{E}^Q \left[exp\left(-r(\tau - t)\right) f(U_\tau) | U_t \right] \tag{2.1}$$

where Q is an appropriate risk-neutral measure, (see [9, 17] for details on Q), $f(\cdot)$ is the payoff function, and the supremum is achieved by all stopping times $\tau \in [t, T]$ given that the option was not exercised prior to t. The concept of a risk-neutral measure Q gives a theoretical risk of zero on the rate of return on an investment.

However, Equation (2.1) is not easily computable on the infinite time horizon because of the supremum function. Common practice is to discretize the interval $[t, T]$ into N equal intervals in which case Equation (2.1) becomes

$$P_t = \max_{\tau \in [t,T]} \mathbb{E}^Q \left[exp\left(-r(\tau - t)\right) f(U_\tau) | U_t \right] \tag{2.2}$$

thereby changing the complex supremum into a maximum that is more easily computable over the discretized interval $[t, T]$.

2.1.3 Black-Scholes

The Black-Scholes model is a mathematical model of a financial market which contains a certain type of derivative instrument[4]. The Black-Scholes formula is derived from the model to price European-style options.

The model's dynamics are defined by the following stochastic process

$$dU_t = rU_t dt + \sigma U_t dW_t \tag{2.3}$$

$$\Rightarrow U_t = U_0 \exp\left(\left(r - \frac{1}{2}\sigma^2\right) t + \sigma W_t\right) \tag{2.4}$$

where r is the continuously compounded risk-free interest rate, σ is the annualized volatility on the returns of the stock and W_t is a Brownian motion. The stock price process is lognormally distributed while the daily returns are normally distributed. This provides a trajectory with non-negative stock prices. There are a number of assumptions that the Black-Scholes model[4] makes about the market:

- there is no arbitrage (i.e. a trader cannot make a riskless profit);
- a risk-free interest rate is used for borrowing and lending cash;
- it is possible to buy and sell at any real-number amount (including fractions), of a stock (including short-selling[1]);
- there are no fees or costs in the above transactions (i.e. the market is frictionless);
- the stock price follows a geometric Brownian motion process with the drift and volatility fixed as a constant; and
- the underlying security does not pay any dividends.

The analytic Black-Scholes price for a European option at time t with $T-t$ years to maturity is given by

$$P(S,K,r,t,T,\alpha) = \alpha\left(U_t\mathcal{N}(\alpha d_1) - Ke^{-r(T-t)}\mathcal{N}(\alpha d_2)\right) \tag{2.5}$$

where

$$d_1 = \frac{\ln(\frac{U_t}{K}) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}},$$

$$\begin{aligned} d_2 &= d_1 - \sigma\sqrt{T-t} \\ &= \frac{\ln(\frac{U_t}{K}) + \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \end{aligned}$$

and $\alpha = 1$ for a call option and $\alpha = -1$ for a put option. r is the riskless interest rate and $\mathcal{N}(\cdot)$ is the standard normal cumulative distribution function. $T-t$ denotes the time-to-maturity of the option and σ defines the annualized volatility (i.e. standard deviation) on the returns of the stock. The analytic Black-Scholes price matrices for both a European put and call will be compared against the Kernel-Based Reinforcement Learning and Monte-Carlo approaches. Figure 2.1 shows these price matrices for a strike price of 10 and an annualized volatility of 0.4 are used for every example. There is no particular preference for using these two numbers because the shape would be the same regardless of the values of the strike price and annualized volatility. However, higher annualized volatility does cause higher option prices but for a limited time-to-maturity of 10 days, this impact is hardly visible.

[1]Short selling is a concept used to describe selling of stock which a trader does not own.

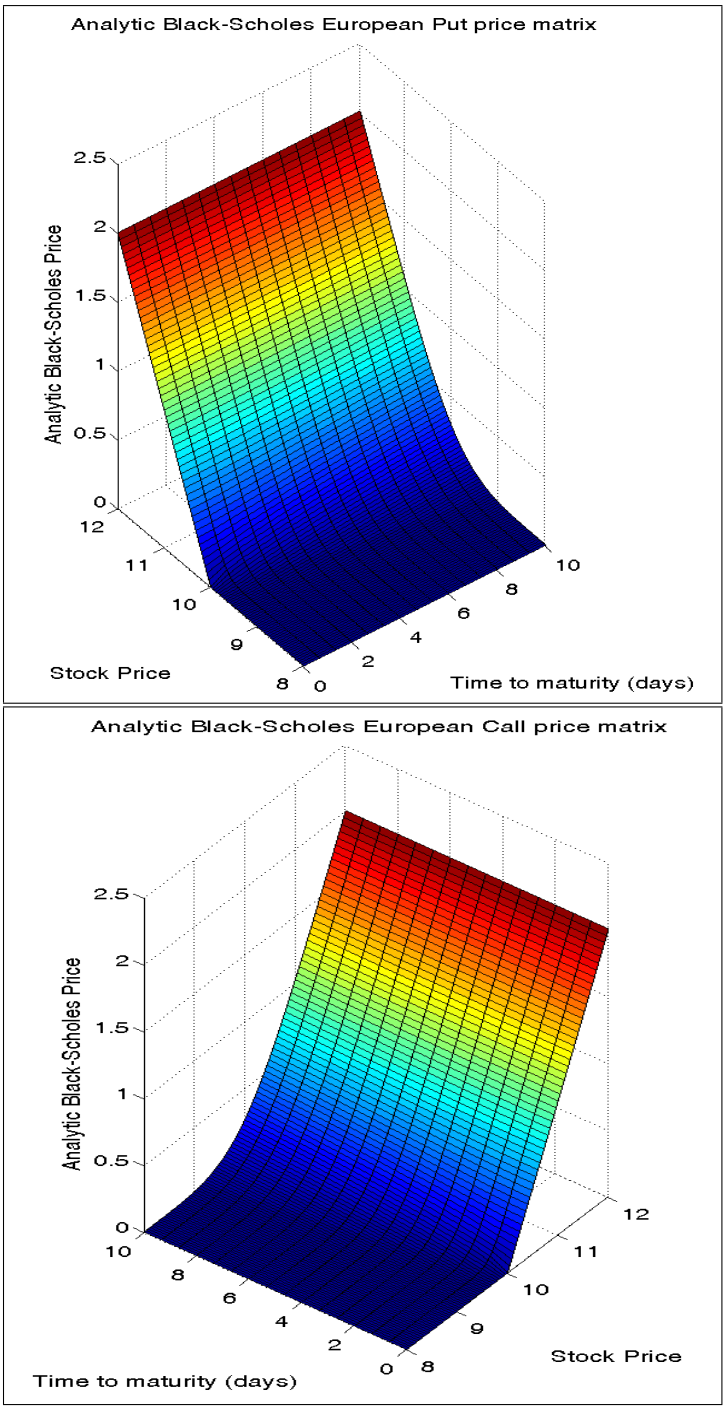


Figure 2.1: European put and call price matrices using analytic Black-Scholes approach

2.1.4 Monte Carlo Simulation

Simulation by pseudorandom numbers is a simplistic pricing model and various techniques exist that can improve the effectiveness of the simple crude Monte Carlo method.

Figure A.2 shows the generic Monte Carlo simulation algorithm that will be used as a comparison to the Kernel-Based Reinforcement Learning model. The algorithm uses the technique of backtracking from the maturity of the option to the inception point taking the maximum between the current payoff and the discounted expectation of the next time period's price. For European options, only the number of simulations needs to be increased leaving the number of time steps as one.

This provides a model that is able to price any class of *Equity* derivative which, by the law of large numbers, converges to the optimal (or fair price) of the option by increasing the number of simulations.

Example: Pricing a European put

Note that a European put's payoff at maturity is given by

$$P_T = \max\{0, K - U_T\} \tag{2.6}$$

and analytical price at any point $t \in [0, T]$ is given by

$$P_t = Ke^{-r(T-t)}\mathcal{N}(-d_2) - U_t\mathcal{N}(-d_1) \tag{2.7}$$

where

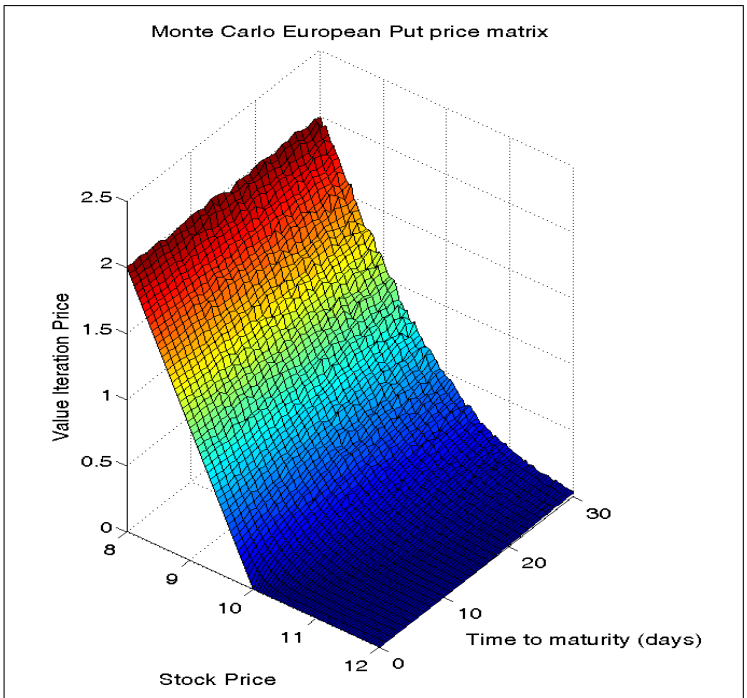
$$d_1 = \frac{\ln(\frac{U_t}{K}) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}$$

and

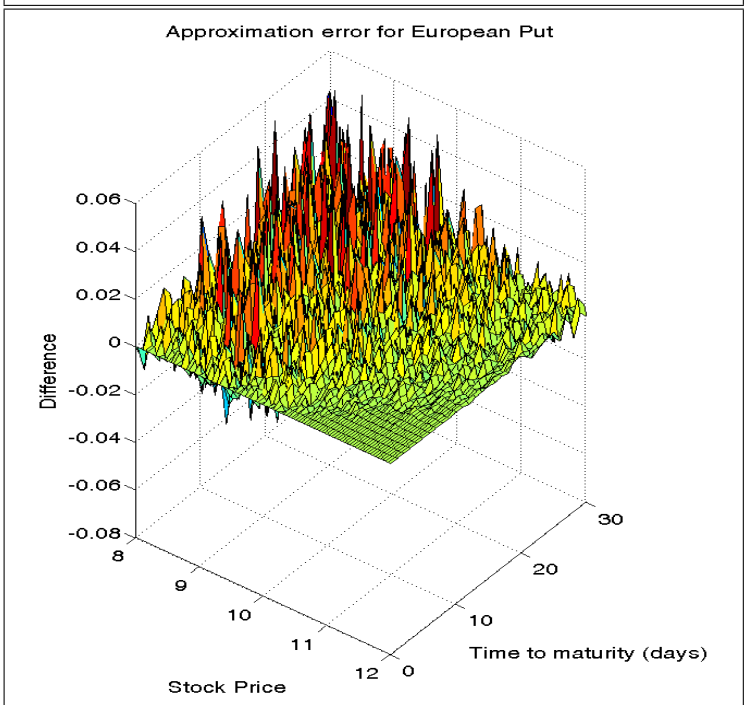
$$\begin{aligned} d_2 &= d_1 - \sigma\sqrt{T-t} \\ &= \frac{\ln(\frac{U_t}{K}) + \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}. \end{aligned}$$

Simulation begins by using the algorithm described in Figure A.1 (Black-Scholes) to calculate a stock price matrix. The resultant stock price matrix, the riskless interest rate (r) and payoff function (Equation (2.6)) are passed as parameters into the Monte Carlo Pricing model depicted in Figure A.2 to obtain a *single* price for a *single* particular option.

The result of this simulation approach (Black-Scholes) are tolerably accurate in comparison to the analytic Black-Scholes European call on $16,000$ states with daily time-steps (Figure 2.2). Both in-the-money and out-the-money regions are consistently bumpy and fluctuate either side of the analytic Black-Scholes European put price. However, the in-the-money regions show slightly higher inaccuracy than the out-the-money regions. By the law of large numbers, increasing the number of simulations of stock prices using the Monte-Carlo approach for pricing options will converge to the analytic Black-Scholes price.



(a) Monte-Carlo European put price matrix (Black-Scholes)



(b) Monte-Carlo European put approximation error (Black-Scholes)

Figure 2.2: European put price matrix using Monte-Carlo Black-Scholes approach with approximation error to the analytic Black-Scholes price matrix

Example: Pricing an American call

From Equation (2.2) in §2.1.2 (American Options) the value for an American call option in the discrete case is

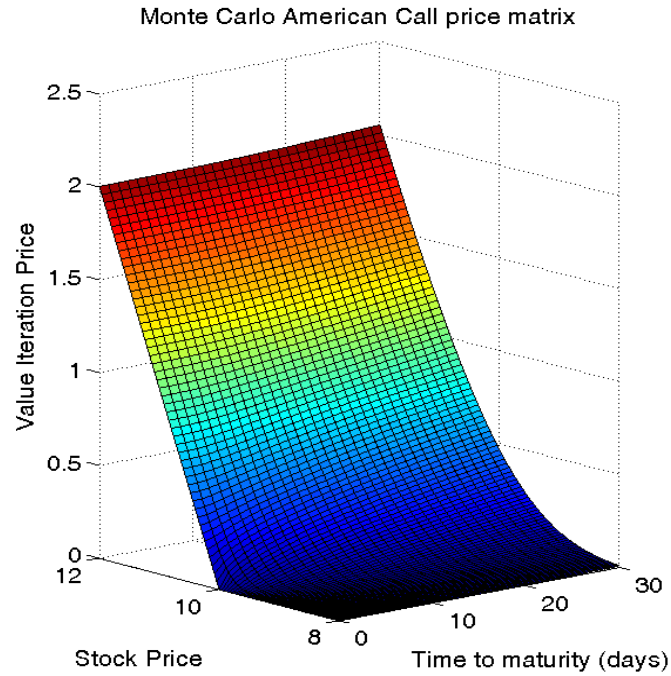
$$C_t = \max_{\tau \in [t,T]} \mathbb{E}^Q \left[exp\left(-r(\tau - t)\right) f(U_\tau) | U_t \right]. \tag{2.8}$$

where

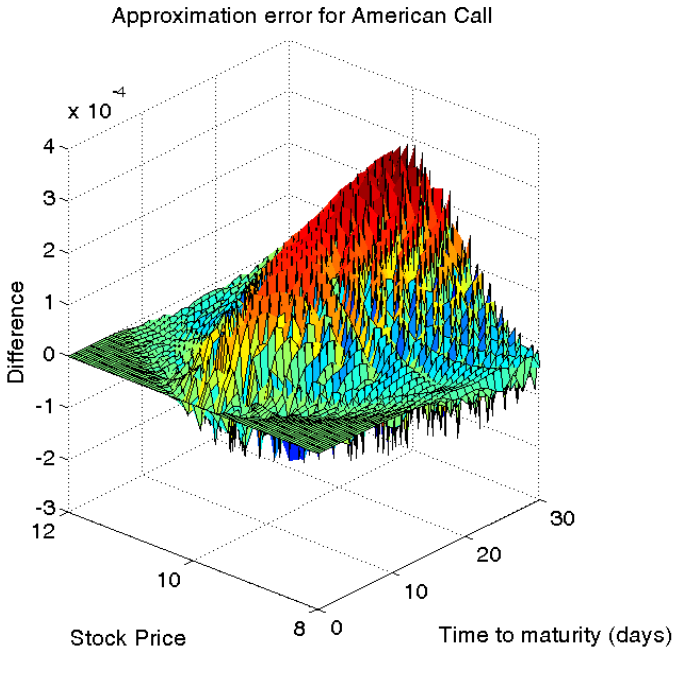
$$f(U_t) = \max\{U_t - K, 0\} \tag{2.9}$$

for the American call option. The Monte Carlo Pricing model in Figure A.2 can be applied because the algorithm works by first applying the payoff at the maturity of each simulated stock path. This then proceeds by progressively working backwards one time-step at each iteration taking the maximum between the discounted value one time-step ahead and the current payoff. The stock paths are generated from the Black-Scholes stock simulation algorithm in Figure A.1.

The result of this simulation approach (Black-Scholes) is satisfactory in comparison to the analytic Black-Scholes European call on $16,000$ states with daily time-steps (Figures 2.3) because the price of an American option on an underlying that pays no dividends should be equal to its European counterpart. For both in-the-money and out-the-money regions, the American call price is fluctuating either side of the analytic European call price.



(a) Monte-Carlo American call price matrix (Black-Scholes)



(b) Monte-Carlo American call approximation error (Black-Scholes)

Figure 2.3: American call price matrix using Monte-Carlo Black-Scholes approach with approximation error to the analytic Black-Scholes price matrix

2.2 Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) were named after Andrey Markov[28]. An MDP is a 5-tuple $(S, A, \mathcal{A}(\cdot), T(\cdot), R(\cdot))$ where

- S is a set, called the state space;
- A is a set, called the action space;
- for each $s \in S$, $\mathcal{A}(s) \subseteq A$ is the set of available actions at s;
- $T : S \times A \times S \to [0, 1]$ is a probability distribution for each $s \in S$ and $a \in A$. This function is called the transition probability function and defines the probabilities of transitioning from state s to state s' if action a is chosen. The following must hold true for T to be a transition probability function:

$$\begin{cases} \sum_{s' \in S} T(s, a, s') = 1 & \text{if the MDP is finite,} \\ \int_{s' \in S} T(s, a, s') = 1 & \text{if the MDP is infinite} \end{cases}$$

  for all $s \in S$ and $a \in \mathcal{A}(s)$; and
- $R : S \times \mathcal{A}(\cdot) \to \mathbb{R}$ is the reward function that assigns a reward to the agent when performing action $a \in \mathcal{A}(s)$ in state $s \in S$.

For every state $s \in S$, a decision (or action a) has to be made from the set of actions $\mathcal{A}(s)$ available in state s. Once an action has been chosen, the system moves into the next state according to a transition probability function $T(s, a, s')$ and observes the one-step reward $R(s, a)$. The transition probability function and the reward function completely specify the most important parts of the dynamics of a finite MDP.

2.2.1 Definitions

Definition 2.1. A *trajectory* is defined as a sequence of state-action-rewards (i.e. $(s_0, a_0, R(s_0, a_0), s_1, a_1, R(s_1, a_1), \ldots, s_j, a_j, R(s_j, a_j), \ldots)$) obtained from the MDP.

Definition 2.2. A *policy* π is a mapping $\pi : S \to \mathcal{A}(\cdot)$ where $\pi(s)$ denotes the action the policy dictates in state $s \in S$.

Definition 2.3. The *return* along a trajectory is the sum of all rewards received by the agent during the trajectory (in the case of an infinite trajectory, a discount factor must be applied so that the return converges to a real number whereas in the case of a finite trajectory, the return will always converge to a real number).

Definition 2.4. The *value function* for a policy π is denoted by $V^\pi : S \to \mathbb{R}$. Given a policy π and state $s \in S$, $V^\pi(s)$ denotes the expected return the agent would obtain across all trajectories from the current state s if the agent chooses actions according to the policy π at every state ($V^\pi(s) = \mathbb{E}\left[\sum_{i=0}^{n} R(s_i, \pi(s_i))\right]$).

Definition 2.5. An *optimal policy* $\pi^* : S \to A$ is a policy such that $V^{\pi^*}(s) \geq V^\pi(s)$ for all policies π and states s. An optimal policy always exists (see Sutton and Barto[32]) and may not be unique.

Definition 2.6. The *optimal value function* $V^* : S \to \mathbb{R}$ is the value function for an optimal policy π^* (i.e. $V^* = V^{\pi^*}$).

2.3 Reinforcement Learning (RL)

Reinforcement Learning is a machine learning paradigm concerned with the problem of learning how to interact with an environment to maximize return. The learner or decision maker is called the agent. The environment is an MDP which comprises of everything outside of the agent's control and the agent interacts with the environment to seek an optimal policy. The agent may be in *only* one of those states of the MDP and may be able to choose from a set of actions it can make in that state. The policy an agent follows provides the agent with exactly *one* action to choose in each state and the value function for that policy provides the return for each state under that policy.

Assume that the environment is a finite MDP (i.e. state and action sets, S and $\mathcal{A}(s)$ for each $s \in S$, are finite) and that its dynamics are given by a transition probability function $T(s, a, s')$ and reward function $R(s, a)$ for all $s \in S$, $a \in \mathcal{A}(s)$ and $s' \in S$.

2.3.1 Value Iteration

A Bellman optimality equation, named after Richard Bellman[2], is a necessary condition for optimality. Optimal policies can easily be obtained once the optimal value functions V^* have been found which satisfy the Bellman optimality equations

$$\begin{aligned} V^*(s) &= \max_{a \in \mathcal{A}(s)} \mathbb{E}[R(s,a) + V^*(s') | s_t = s, a_t = a, s_{t+1} = s'] \\ &= \max_{a \in \mathcal{A}(s)} \sum_{s'} T(s,a,s') \left[R(s,a) + V^*(s') \right] \end{aligned} \tag{2.10}$$

for all $s \in S$. The optimal policy $\pi^* : S \to A$ can be obtained from the optimal value function $V^* : S \to \mathbb{R}$ by

$$\pi^*(s) = \arg \max_{a \in \mathcal{A}(s)} \sum_{s'} T(s,a,s') \left[R(s,a) + V^*(s') \right] \tag{2.11}$$

for all $s \in S$.

Value iteration is an iterative dynamic programming algorithm for solving the Bellman optimality equation of an MDP (2.10) and hence obtaining the optimal value function V^*.

$$\begin{aligned} V_{k+1}(s) &= \max_{a \in \mathcal{A}(s)} \mathbb{E}[R(s,a) + V_k(s')] \\ &= \max_{a \in \mathcal{A}(s)} \sum_{s'} T(s,a,s') \left[R(s,a) + V_k(s') \right] \end{aligned} \tag{2.12}$$

for all $s \in S$ where k denotes the kth value update. For arbitrary V_0, the sequence $\{V_k\}$ can be shown to converge to V^* under the same conditions that guarantee the existence of V^*[32].

Value iteration formally requires an infinite number of iterations to converge exactly to V^*. In practice, the procedure stops once the value function changes by only a small amount in a sweep. Figure 2.4 gives a complete value iteration algorithm with this kind of termination condition.

Although dynamic programming (DP) ideas such as value iteration can be applied to problems with continuous state and action spaces, exact solutions are possible only in special cases. A common way of obtaining approximate solutions for tasks with continuous states and actions is to discretise the state and action spaces and then apply finite-state DP methods.

VALUE-ITERATION$(S, A, \mathcal{A}, T, R)$

S - set of market states
A - set of actions
$\mathcal{A}$ - a function mapping states to a set of available actions in that state
T - transition probablity matrix of size $|S| \times |A| \times |S|$
R - reward function

$V(s) \leftarrow 0$ for all $s \in S$
repeat
  $\Delta \leftarrow 0$
  for each $s \in S$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_{a \in \mathcal{A}(s)} \sum_{s'} T(s, a, s') \left[R(s, a) + V(s')\right]$
    $\Delta \leftarrow \max\left(\Delta, |v - V(s)|\right)$
until $\Delta < \epsilon$ (a small positive number)
return V

Figure 2.4: Value Iteration Algorithm via V-update

2.3.2 Kernel-Based Reinforcement Learning

Kernel-Based Reinforcement Learning (KBRL) is an approach introduced by Ormoneit and Sen[26] and later enhanced by Jong and Stone[21] that accurately handles stochastic environments with continuous state spaces. This approach gives a finite MDP from the infinite state space enabling the application of value iteration calculating the policy and value function for each of those states in the discretised finite set of states.

Extracting samples from continuous state space S

Consider an MDP with a continuous state S. Since KBRL is commonly used when the state space S is continuous, discretization is applied on the continuous space by extracting a set of sample transitions (a transition is a trajectory with only a single step). Extract n transitions $(t_1, \ldots, t_n)$ where each transition is a 4-tuple $t_i = (s_i, a_i, R(s_i, a_i), s_i')$ denoting the current state s_i, an action

a_i that was chosen in state s_i, the reward $R(s_i, a_i)$ given by the environment being in state s_i when a random action a_i was chosen and the agent's successor state s'_i. Let $\bar{S} = \{s'_1, s'_2, \ldots, s'_n\}$ be the set of all sampled successor states.

Defining the transition probability function

For an action a in a given state $s \in S$, KBRL approximates the transition probabilities and reward as a weighted average of previous outcomes of choosing that action. That is, by observing the rewards obtained from the environment on all sampled transitions which chose action $a_i = a$ for all $1 \leq i \leq n$, the weighted average of all those transitions approximates the reward for the current state. The weights are calculated by a kernel function of the distances between the current state s and all initial states s_i from each transition t_i for all $1 \leq i \leq n$. The distance function $d(\cdot, \cdot)$ is used to calculate the distance between potential successor's predecessors and the current state. The closer the successor's predecessor state is to the current state, the higher the probability of getting to that successor state from the current state. The bandwidth parameter b scales the distance function $d(\cdot, \cdot)$ in order to eliminate states deemed to be too far away from the current state. The kernel function $\phi(\cdot)$ is a non-negative function $\left(\text{typically a Gaussian kernel is used, i.e. } \phi(x) = e^{-x^2}\right)$ that determines the relative weight of each transition and

$$Z_{s,a} = \begin{cases} \sum_{i=1}^{n} \phi\left(\frac{d(s_i,s)}{b}\right) & \text{if } a_i = a, \\ 0 & \text{if } a_i \neq a \end{cases} \tag{2.13}$$

normalizes the weights (see Jong and Stone[21]) - note that s_i is the state retrieved from transition t_i for all $1 \leq i \leq n$. The transition probability function $\hat{T} : S \times A \times \bar{S} \to [0, 1]$ over the continuous state space S is

$$\hat{T}(s, a, s'_i) = \begin{cases} \frac{1}{Z_{s,a}} \phi\left(\frac{d(s_i,s)}{b}\right), & \text{if } a_i = a, \\ 0, & \text{if } a_i \neq a \end{cases} \tag{2.14}$$

for all $s \in S$, $a \in A$, and $s'_i \in \bar{S}$ where s_i, a_i and s'_i are retrieved from transition t_i for all $1 \leq i \leq n$. The transition probability function $\tilde{T} : \bar{S} \times A \times \bar{S} \to [0, 1]$ over the finite state space $\bar{S}$ is

$$\tilde{T}(s, a, s'_i) = \begin{cases} \frac{1}{Z_{s,a}} \phi\left(\frac{d(s_i,s)}{b}\right), & \text{if } a_i = a, \\ 0, & \text{if } a_i \neq a \end{cases} \tag{2.15}$$

for all $s \in \bar{S}$, $a \in A$, and $s'_i \in \bar{S}$ where s_i, a_i and s'_i are retrieved from transition t_i for all $1 \leq i \leq n$.

KBRL uses sampled transitions $(t_1, t_2, \ldots, t_n)$ to define an approximate MDP $\hat{M} = (S, A, \mathcal{A}(\cdot), \hat{T}(\cdot), R(\cdot))$. The Bellman optimality equation for $\hat{M}$ is:

$$\hat{V}^*(s) = \max_{a \in \mathcal{A}(s)} \sum_{i=1}^{n} \hat{T}(s, a, s'_i)[R(s_i, a_i) + \hat{V}(s'_i)], \tag{2.16}$$

for all $s \in S$ where s_i, a_i and s'_i are retrieved from transition t_i for all $1 \leq i \leq n$. The more sampled transitions that are extracted from the continuous state S, the closer the optimal value function $\hat{V}(\cdot)$ in (2.16) is to the true optimal value function. Note that $\hat{V}(\cdot)$ is a function over the continuous state space S. Since the sampled successor state space $\bar{S}$ is a finite subset of the continuous state space S ($\bar{S} \subset S$), observe that the MDP

$$\tilde{M} = (\bar{S}, A, \mathcal{A}(\cdot), \tilde{T}(\cdot), R(\cdot))$$

is a well-defined finite MDP. Therefore, the Bellman optimality equation for $\tilde{M}$ is

$$\tilde{V}^*(s) = \max_{a \in \mathcal{A}(s)} \sum_{i | a_i = a} \tilde{T}(s, a, s'_i) \left(R(s_i, a_i) + \tilde{V}(s'_i) \right) \tag{2.17}$$

for all $s \in \bar{S}$ which is equivalent to (2.10). Hence, Value Iteration can be applied to solve for the optimal value function $\tilde{V}^*(s)$ for each state $s \in \bar{S}$. The optimal value for any state $s \in S$ is approximated by

$$\hat{V}^*(s) = \frac{\sum_{i=1}^{n} \phi\left(\frac{d(s_i, s)}{b}\right) \tilde{V}^*(s'_i)}{\sum_{i=1}^{n} \phi\left(\frac{d(s_i, s)}{b}\right)}. \tag{2.18}$$

The effect of using a Gaussian kernel to weight the calculated values is illustrated in Figure 2.5. In this illustration, the states are Euclidean points $s = (x, y)$ and we use a Gaussian kernel on the Euclidean distance function. The current state $s = (1.3, 1.5)$ is at the center of Figure 2.5. The circles indicate initial states that have equal probability of having their successor state as the successor state of the current state. As the number of sampled transitions increases, the bandwidth parameter b must decrease at a suitable rate to balance the variance and bias of $\hat{V}$ [26]. Table D.1 shows different classes of basic kernel functions together with the extended kernels which are weighted variants yielding kernel classes with significantly more hyperparameters.

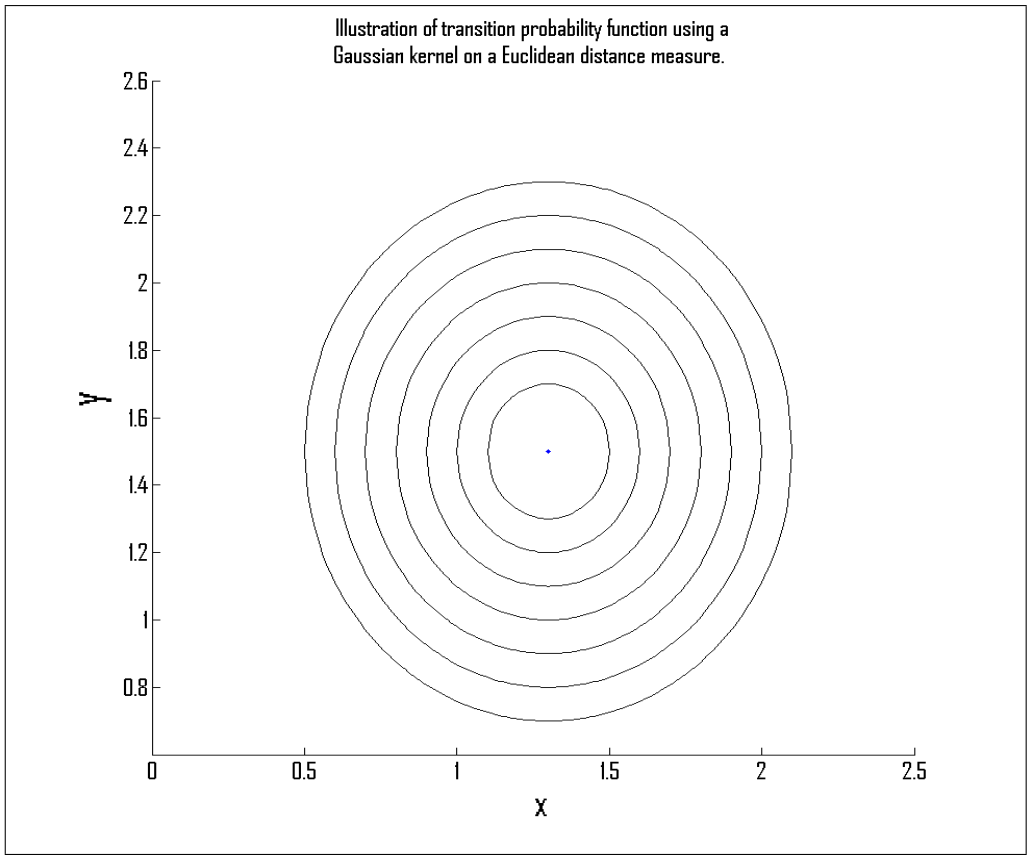


Figure 2.5: Illustration of transition probability function using a Gaussian kernel on a Euclidean distance measure

2.4 Summary

This chapter has introduced the core concepts of financial options. European and American options were described and the differences between them were discussed. The Black-Scholes option pricing model was illustrated and the assumptions of the model were described. Monte Carlo simulation was introduced and applied to the Black-Scholes and Heston models to solve for the European put and American call option prices.

An introduction to Markov Decision Processes (MDPs) was given with a definition of what an MDP is in addition to definitions of important terms relating to MDPs. Reinforcement Learning (RL) introduced the concept of how a finite MDP can be solved via the Value Iteration algorithm. The RL section extended into describing the Kernel-Based Reinforcement Learning (KBRL) method which is able to use a continuous state-space MDP and transform

that MDP into a finite MDP that can be solved via Value Iteration.

The following chapter shows how the European option pricing problem can be formulated as an MDP. This MDP can then be solved via Kernel-Based Reinforcement Learning.

Chapter 3

European options

The previous chapter introduced the core concepts of the type of options, Markov Decision Processes (MDPs) and Reinforcement Learning. This chapter is focused on creating an MDP for the European option pricing problem and showing how this MDP can be solved via Kernel-Based Reinforcement Learning. Experiments will be shown and explained with results being compared to the analytic Black-Scholes prices. For all the following MDPs, let D be a European derivative on a stock U.

3.1 Initial Trajectory model

This MDP is based on unpublished work by Grassl[13]. The unpublished work makes the assumption that the optimal policy on a European option is to hold the option to maturity. In this paper, proof will be given that indeed the optimal policy for a European option will be to hold to maturity. A trader can either sell D at time t at the current spot price D_t or decide to hold on to it. If he still holds D at time T, both exercising and holding will have the same outcome: the derivative's payoff will be received, $D_T = f(U_T)$. A complete market state m for a stock option is the combination of the strike-relative stock price and risk/time component which is equivalent to

$$m = \left(\frac{U_t}{K}, \sigma\sqrt{T-t} \right).$$

The strike price for every market state m is one. A state s is represented as a combination of the trader's position q (whether the trader has the derivative

or not) and the complete market state m. That is,

$$s = (m, q)\,. \tag{3.1}$$

The possible actions for the trader at a state s are

$$a = \begin{cases} A_S & \text{if } D \text{ is being sold,} \\ A_H & \text{if } D \text{ is being held.} \end{cases} \tag{3.2}$$

Since a single trader's actions has a negligible affect on the market, this research assumes that neither action can affect the market's behaviour (i.e. the successor market state of m (m') is independent of a). Only the trader's successor position q' depends on his actions and influences the subsequent rewards. The reward function will be the trader's monetary compensation future dated to time T if sold before T.

$$R(s,a) = \begin{cases} D_T = f(U_T) & \text{if } \sigma\sqrt{T-t} = 0, \\ e^{r(T-t)}D_t & \text{if } a = A_S \text{ and } \sigma\sqrt{T-t} > 0, \\ 0 & \text{if } a = A_H \text{ and } \sigma\sqrt{T-t} > 0. \end{cases} \tag{3.3}$$

which is equivalent to the following reward structure taking into account that the volatility σ is a positive non-zero value

$$R(s,a) = \begin{cases} D_T = f(U_T) & \text{if } t = T, \\ e^{r(T-t)}D_t & \text{if } a = A_S \text{ and } t < T, \\ 0 & \text{if } a = A_H \text{ and } t < T. \end{cases} \tag{3.4}$$

The state s is defined in (3.1), however q is not needed in the reward function $R(s,a)$ since the Value Iteration procedure only considers states where the agent has the derivative (i.e. $q = 1$ for all states). If states where $q = 0$ were considered also, those states would automatically be zero without any need for calculation, hence those states are ignored. Using $R(s,a)$ as defined above, the value function of a given policy π can be written as

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{j=0}^{n} R(s^{(j)}, \pi(s^{(j)}))\right] \tag{3.5}$$

where V^{π} is the value function defined in §2.3.1, π denotes the policy used to retrieve the action a_i in the state s_i and $s = s^{(0)}, s^{(1)}, s^{(2)}, \ldots, s^{(n)} = s_T$ considers all trajectories to terminal states s_T from the current state s. Note that the value of $V^{\pi}(s)$ is future-valued to time T.

3.1.1 Constructing an approximate KBRL MDP

To apply KBRL to the MDP in the previous section, we require a set of transitions of the form $t_i = (s_i, a_i, R(s_i, a_i), s_i')$. Each s_i is a tuple (m_i, q_i) where m_i is the market state and q_i represents the trader's position.

The market state transitions independently of the chosen action a. This means that given market states $m_1 = (\frac{U_1}{K_1}, \sigma_1\sqrt{T-t_1})$ and $m_2 = (\frac{U_2}{K_2}, \sigma_2\sqrt{T-t_2})$, the probability of transitioning from m_1 to m_2 is defined by $\tilde{T}(m_1, m_2)$. This independence allows us to separate the transition function into $\tilde{T}^m(m, m')$ and $\tilde{T}^q(q, a, q')$. Since $\tilde{T}^q$ is known:

$$\tilde{T}^q(q,a,q') = \begin{cases} 1 \text{ if } q=1,\ a=A_H,\ q'=1, \\ 1 \text{ if } q=1,\ a=A_S,\ q'=0, \\ 0 \text{ otherwise} \end{cases} \tag{3.6}$$

we need only generate sample transitions for $\tilde{T}^m$, henceforth referred to as $\tilde{T}$. This yields the following simplified versions of (2.13) and (2.15):

$$Z_m = \sum_{i=1}^{n} \phi\left(\frac{d(m_i, m)}{b}\right) \tag{3.7}$$

$$\tilde{T}(m, m_i') = \frac{1}{Z_m}\phi\left(\frac{d(m_i, m)}{b}\right) \tag{3.8}$$

where m is the market state, m_i' are all the successor market states and m_i are the corresponding predecessor market states of m_i'.

The transitions are obtained by first generating sample market trajectories using the Black-Scholes stock price process[1] and extracting the one step transitions from these trajectories[2].

The starting market states of each of a set of n derivatives are sampled (see Figure 3.1 for the pseudocode for generating n derivatives). Each of these states is used as the initial state of a trajectory of a randomly assigned length. The DERIVATIVES procedure in Figure 3.1 creates random derivatives for use in the sampling of market states in the Kernel-Based Reinforcement Learning MDP. The procedure takes one argument as input: the number of derivatives n

[1] Any other stock price process could be used (such as Heston - which also provides the volatility samples)

[2] Figures C.2, C.3 and C.4 show 10, 100 and 10,000 sampled trajectories, respectively, with approximately 110, 1,100 and 110,000 total market states

to create. A derivative holds five key values: the stock price s_0, strike price K, volatility of returns σ, inception of the derivative t and maturity of the derivative T. Stock prices are uniform random numbers generated on the interval $(0, 50]$ - the upper bound is picked at random and any number greater than 0 could potentially work but keep in mind about floating point arithmetic issues on computers. The strike price is generated using the generated stock price by applying $s_0(1-0.12\times\mathcal{N}(0,1))$ - the constant 0.12 guarantees that we will get a strike price in close proximity of the stock price which is important when price options. The volatilities are uniform random numbers generated over $(0, 1]$ - rarely (if ever) is volatility remotely close to one but allowing for volatility to be sampled around one caters for extreme circumstances. Inceptions are always set to 0 since the maturity is uniformly sampled over $(0, 1]$ - this interval size can be increased or decreased depending on the time-to-maturity of the options that are being priced.

DERIVATIVES(n)
$n \leftarrow$ number of random options to generate

1 $D \leftarrow$ initialize to zero matrix of $n \times 5$
2 **for** $i \leftarrow 1$ **to** n **do**
3 $D(i, s_0) \leftarrow$ uniform random number on $(0, 50]$
4 $D(i, K) \leftarrow D(i, s_0) \times (1 - 0.12 \times \mathcal{N}(0, 1))$
5 $D(i, \sigma) \leftarrow$ uniform random number on $(0, 1]$
6 $D(i, t) \leftarrow 0$
7 $D(i, T) \leftarrow$ uniform random number on $(0, 1]$
8 **return** D

Figure 3.1: Derivative sampling

Successive market states are sampled using the Black-Scholes stock price process using the previous market state on the trajectory (i.e. the second market state is sampled from the starting market state, the third market state is sampled from the second market state and so on until the transition size limit is reached). The MARKET-STATES procedure in Figure 3.2 takes in four arguments: the number of random derivatives to generate n, the risk-free

interest rate r, the minimum number of derivatives to have on a trajectory *lower* and the maximum number of derivatives to have on a trajectory *upper*. First, the n random derivatives are sampled. These derivatives are the starting points for each trajectory, therfore, there will be n trajectories in total. Next, the size of each trajectory is calculated by sampling a random integer over $[lower, upper]$. The first market state of each trajectory is created for each $i = 1, 2, \ldots, n$:

$$M_{i,1} = \left(\frac{s_0^i}{K}, \sigma\sqrt{T-t} \right).$$

For each $k = 2, 3, \ldots, m_i$ (where $m_i + 1$ is the size of the trajectory for each $i = 1, 2, \ldots, n$), the successive states in each trajectory are calculated using the Black-Scholes stock price formula:

$$M_{i,k} = \left(\frac{s_k^i}{K}, \sigma\sqrt{T-t_k^i} \right)$$

where s_k^i is the successive stock price of s_{k-1}^i (using the Black-Scholes stock price formula) and t_k^i is the successive time of t_k^i.

The transition probability matrix is calculated for each sampled market state m in the set of sampled market states by using (3.8) and (3.7). Pseudocode for this calculation is presented in Figure 3.3. A significant computational obstacle is that the size of $\tilde{T}$ is quadratic in the number of observed samples. Jong and Stone[21] suggest that very small entries of $\tilde{T}$ should be set to zero. Population of the rewards matrix is achieved by using Black-Scholes analytic prices for the *sell* action and applying the payoff at maturity for the *hold* action. The TRANSITION-PROBABILITIES procedure in Figure 3.3 calculates the transition probability matrix for each market state. This procedure takes in four input arguments: the number of random derivatives to generate n, the risk-free interest rate r, the minimum number of derivatives to have on a trajectory *lower* and the maximum number of derivatives to have on a trajectory *upper*. First, the market states are retrieved using (in this case) the MARKET-STATES procedure however the MARKET-STATES-SEPARATE-FEATURES can be used instead to get seperated features transition probabilities which is used in the improved models. Final (or terminal) states cannot transition into another state and no non-final state can transition to a start state. Therefore, the algorithm only considers non-final states' transitions to non-start states. The predecessor of the non-final state is retrieved and is passed into the PHI procedure to calculate a potential.

Once all potentials have been calculated for a particular non-final state, the sum of those potentials is calculated and used to transform the potentials into transition probabilities.

MARKET-STATES$(n, r, lower, upper)$

$n \leftarrow$ number of random options to generate
$r \leftarrow$ risk-free interest rate
$[lower, upper] \leftarrow$ range of transitions to generate from

```
1  D ← DERIVATIVES(n)
2
3  transitions ← initialize to zero array of size n
4  m ← n    // m is the total number of market states
5  for i ← 1 to n do
6      transitions(i) ← uniform random number on [lower, upper]
7      m = m+transitions(i)
8
9  M ← initialize market state matrix to zero matrix of size m × 3
10 k ← 0
11 for i ← 1 to n do
12     strikeRelativeStock ← D(i, s_0)/D(i, K)        // s̄
13     riskTime ← D(i, σ)√(D(i, T) − D(i, t))         // σ̄
14
15     M(k, s̄) ← strikeRelativeStock
16     M(k, σ̄) ← riskTime
17     M(k, d̄) ← D(i) // Places derivative in 3rd place holder
18
19     k ← k + 1
20     h ← (D(i, T) − D(i, t)) / transitions(i)
21     for j ← 1 to transitions(i) do
22         t' ← D(i, t) + j ∗ h
23         M(k, s̄) ← BSFORMULA(M(k − 1, s̄), r, D(i, σ), h)
24         M(k, σ̄) ← D(i, σ)√(D(i, T) − t')
25
26         d' = (s_0, K, σ, t', T)
27         M(k, d̄) ← d'        // Places derivative in 3rd place holder
28
29         k ← k + 1
```

Figure 3.2: Market State sampling

```
TRANSITION-PROBABILITIES(n, r, b, ν, lower, upper)
    n ← number of random options to generate
    r ← risk-free interest rate
    b ← standard deviation of kernels
    ν ← egg-shaped parameter
    [lower, upper] ← range of transitions to generate from

 1  M ← MARKET-STATES(n, r, lower, upper)
 2  m ← length of M
 3  T ← initialize to zero matrix of size m × m
 4
 5  for i ← 1 to m do
 6      Z ← 0
 7      if M(i) is not a terminal state then
 8          for j ← 1 to m do
 9              if M(j) is not a start state then
10                  k ← j's predecessor
11                  T(i, j) ← PHI(M(i), M(k), b, ν)
12                  Z ← Z + T(i, j)
13  for j ← 1 to m do
14      T(i, j) ← T(i, j)/Z
15  return (T, M, m)
```

Figure 3.3: Transition Probability calculation

The PHI function in Figure 3.4 provides a potential value for current state to a successor state. The function takes four input arguments: the current market state m, the predecessor market state of a potential successor state $m^{(i)}$, the standard deviation of the kernels parameter b and the egg-shaped parameter ν. The potential value for m and $m^{(i)}$ is defined by

$$e^{-\frac{d\left(m,m^{(i)}\right)}{b^2}}$$

where d is the distance function applicable to m and $m^{(i)}$.

This completes the description of the finite MDP obtained for the KBRL method. Because of the separation of the transition function, the Bellman

PHI$(m, m^{(i)}, b, \nu)$

$m \leftarrow$ current market state
$m^{(i)} \leftarrow$ a predecessor market state
$b \leftarrow$ standard deviation of kernels
$\nu \leftarrow$ egg-shaped conversion parameter

$D_1 \leftarrow m(\bar{s_0}) - m^{(i)}(\bar{s_0})$
$D_2 \leftarrow m(\bar{\sigma}) - m^{(i)}(\bar{\sigma})$
if $D_2 < 0$ **then**
    $D_2 \leftarrow D_2 \times \nu$
$d \leftarrow D_1^2 + D_2^2$

return $\exp(-d/(b^2))$

Figure 3.4: Kernel function

optimality equation for the MDP is

$$\tilde{V}^*(s) = \max_{a \in \mathcal{A}(s)} \left(\sum_{i=1}^{n} \tilde{T}(m, m_i') \left[R(s_i, a) + \tilde{V}(s_i') \right] \right). \tag{3.9}$$

Solving (3.9) gives $\tilde{V}^*$ which can be done by using value iteration shown in section 2.3.1.

Once the MDP has been solved and an optimal value function received, pricing an option that is not in the sampled state space needs to be calculated. The KBRL method of retrieving values from a value function for unseen states is to perform a variation of a weighted average on the sampled states. This weighted average is done by using the transition probability function to determine the weights of each sampled state from the unseen state. For each state, the value of sampled state from the value function is multiplied by the weight (or probability of transitioning to that state from the current unseen state). Summing up those values gives the fair value for being in the unseen state.

The value of state $s = (m, q)$ is

$$V^*(s) = \frac{\sum_{i=1}^{n} \phi\left(\frac{d(m_i, m)}{b}\right) \tilde{V}^*(s_i')}{\sum_{i=1}^{n} \phi\left(\frac{d(m_i, m)}{b}\right)} \tag{3.10}$$

with $q = 1$, $\tilde{V}^*(s'_i)$ are the values of the sampled states and m_i are all the sampled market states. When $q = 0$, $V^*(s) = 0$.

3.1.2 Transition Probability refinement

The transition probabilities are not intuitive at first glance. This section should aid the reader in understanding how these probabilities affect the potential successors of any state. Recall from the previous section (§3.1.1) that the transition probability from one market state m to a potential successor market state m'_i is defined as

$$\tilde{T}(m, m'_i) = \frac{1}{Z_m}\phi\left(\frac{d(m_i, m)}{b}\right)$$

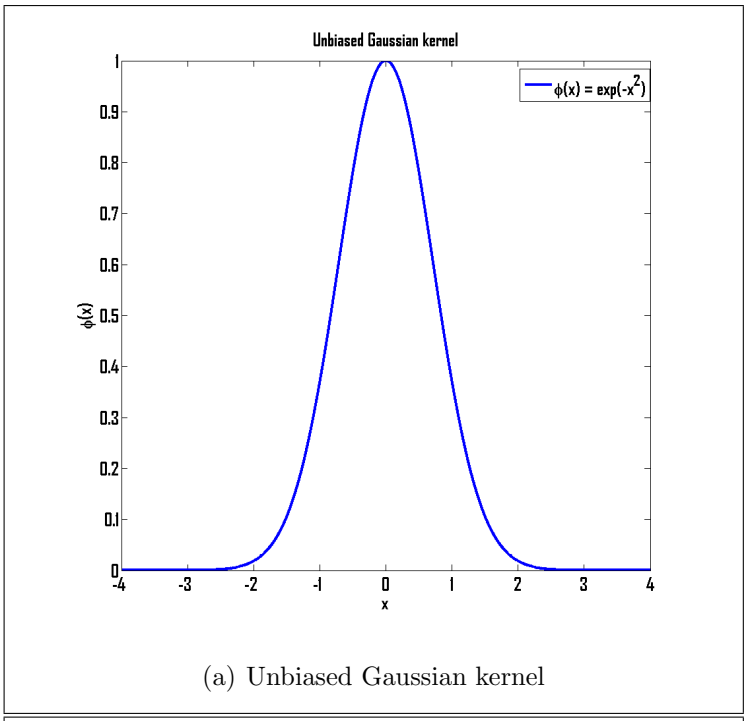
where Z_m is the normalizer defined in (3.7), $d(\cdot, \cdot)$ is the distance function and b is the standard deviation of the kernel.

Using a Euclidean distance function within the Gaussian kernel is problematic in that $\tilde{T}(m_1, m_2)$ may be equal to $\tilde{T}(m_2, m_1)$ if $m_1 = (x_1, x_2)$ and $m_2 = (x_2, x_1)$, for $x_1, x_2 \in \mathbb{R}^+$. That is, the Euclidean distance is unbiased with regards to single components of a state. A Gaussian kernel $\left(\phi(x) = e^{-x^2}\right)$ is a symmetric, bell-shaped curve (Figure 3.5a) with a maximum value of one when $x = 0$. A Euclidean distance metric calculates the straight line distance between two points (the two-dimensional case is shown in (3.11))

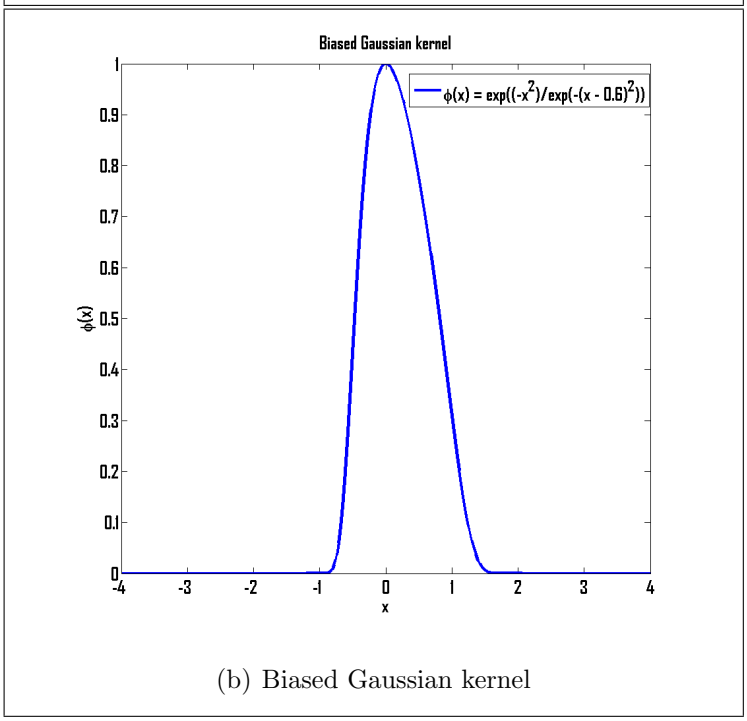
$$d(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.11}$$

where $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ are two points in a two-dimensional space. An *unbiased* Gaussian kernel (Figure 3.5a) is symmetric about the y-axis since $\phi(x) = \phi(-x)$ for all $x \in \mathbb{R}$. A *biased* Gaussian kernel (Figure 3.5b) is a modified Gaussian kernel which is neither symmetric nor bell-shaped.

Experiments suggest that it is a better idea to compare options with similar strike-relative stock prices and different risk than to compare options with different strike-relative stock prices and similar risk[13]. Also, noticing that the risk component ($\sigma\sqrt{T - t}$) is decreasing as t increases, the value function approximation should ideally look at states that *only* lie in the future. However, this would impact states that have risk components very close to zero as these states would suffer from a lack of data. A scheme needs to be devised whereby transition probabilities are more forward looking than backward looking.



(a) Unbiased Gaussian kernel



(b) Biased Gaussian kernel

Figure 3.5: Unbiased and biased Gaussian kernels

The proposed suggestion moves away from a standard Euclidean distance measure and leads toward a *weighted* Euclidean distance measure. This weighted measure is created in such a way that its contours are egg-shaped curves (as

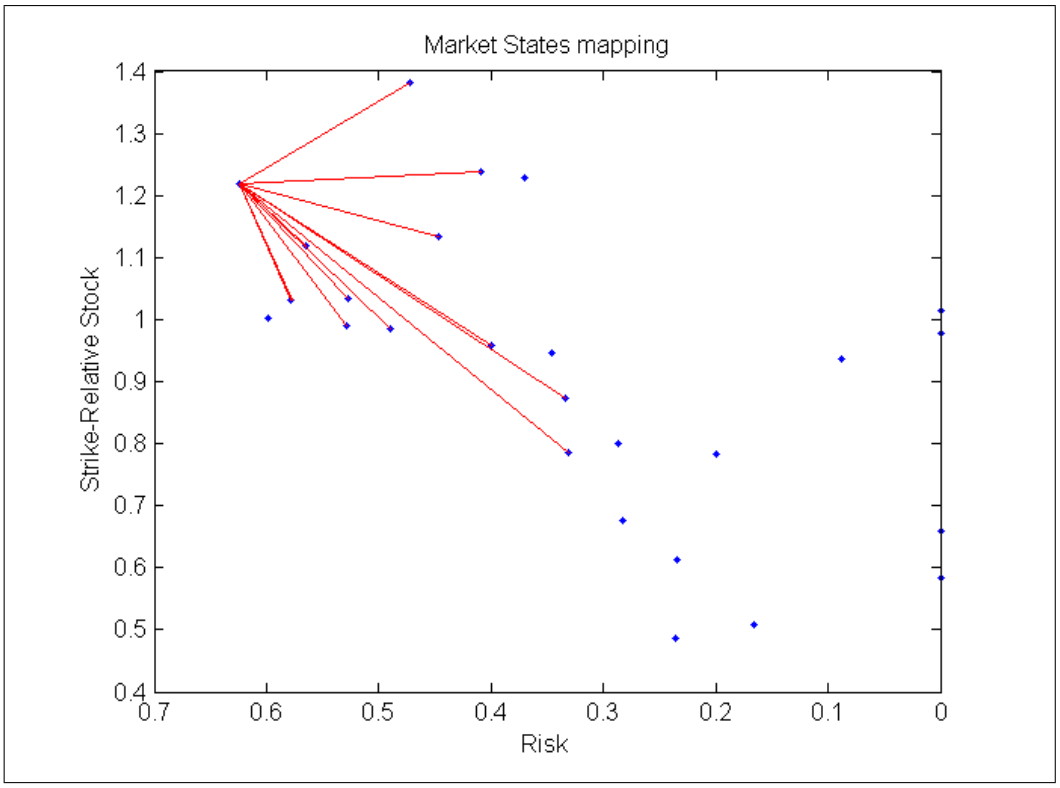


Figure 3.6: Transition Probabilities from high risk

opposed to circular in the standard Euclidean distance measure) and the curves are tilted towards smaller values of $\sigma\sqrt{T-t}$ (i.e. transition probabilities are more forward looking than backward looking - Figures 3.6 and 3.7). This weighted Euclidean distance measure is defined by

$$d_W(m_1, m_2) = \sqrt{\left(\frac{U_1}{K_1} - \frac{U_2}{K_2}\right)^2 + \nu\left(\sigma_1\sqrt{T_1 - t_1} - \sigma_2\sqrt{T_2 - t_2}\right)^2} \tag{3.12}$$

where $m_1 = \left(\frac{U_1}{K_1}, \sigma_1\sqrt{T_1 - t_1}\right)$, $m_2 = \left(\frac{U_2}{K_2}, \sigma_2\sqrt{T_2 - t_2}\right)$ are market states and

$$\nu = \begin{cases} 1 & \text{if } \sigma_1\sqrt{T_1 - t_1} > \sigma_2\sqrt{T_2 - t_2} \\ 3.6 & \text{if } \sigma_1\sqrt{T_1 - t_1} \leq \sigma_2\sqrt{T_2 - t_2}. \end{cases} \tag{3.13}$$

The lines in Figures 3.6 and 3.7 represent the successor states accessible from the current state (the point with which all lines connect is the current state). The circle line in Figure 3.7 shows that states may transition to themselves depending on how far their predecessor is from the current state and how many and how far potential successors' predecessors are from the current

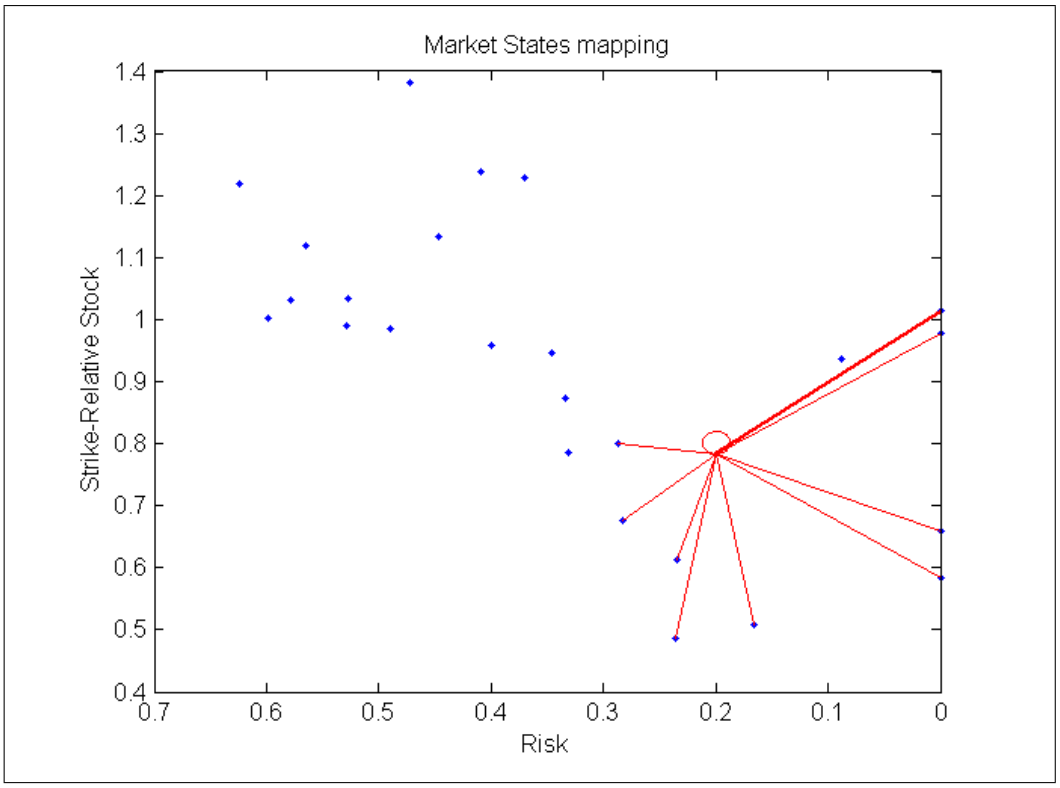


Figure 3.7: Transition Probabilities from low risk

state. Figure C.6a shows the original, unbiased Gaussian with respect to the risk/time component that causes inaccurate prices. Figure C.6b shows the modified, biased Gaussian for the risk/time component. The strike-relative stock price component remains as an unbiased Gaussian.

3.1.3 Experiment (initial Trajectory model): Pricing a European put option

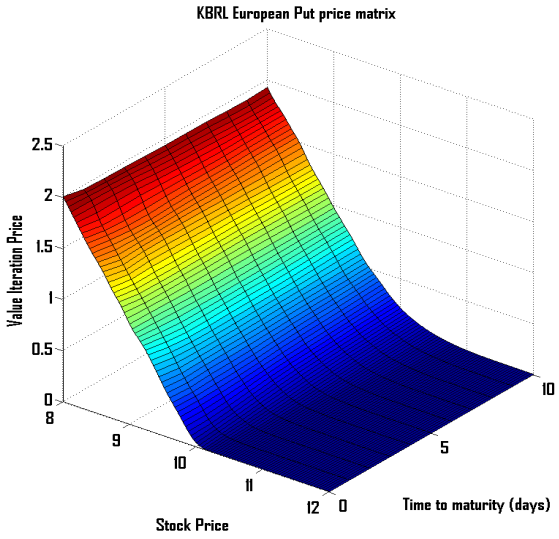
This experiment considers pricing a European put using Grassl's model[13]. The following parameter values are used in solving for the value function using Kernel-Based Reinforcement Learning (KBRL):

- the number n of randomly sampled derivatives: $n = 5,000$;
- the riskless interest r: $r = 0.02$;
- the standard deviation b of the kernel functions: $b = 0.013$;
- the egg-shaped parameter ν applied to the distance function is given by (3.13);
- number of transitions per trajectory is an integer between 1 (inclusive) and 19 (inclusive); and
- the actions to use are the *hold* and *sell* actions previously discussed.

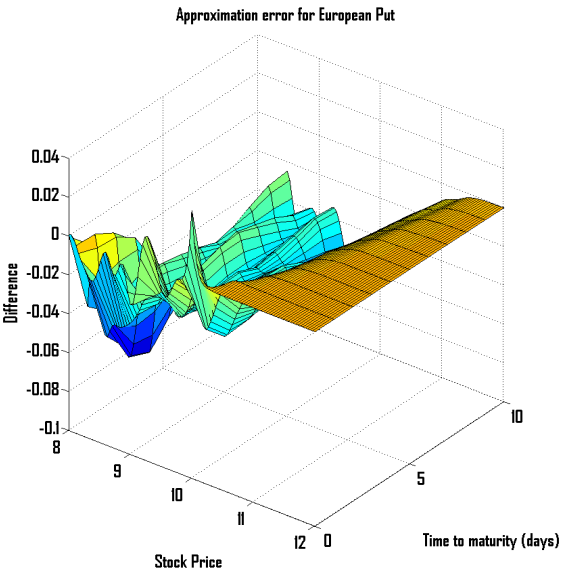
In order to determine the accuracy of the method, the calculated values are compared to the Black-Scholes price for each state. Due to the structure of the reward function, the optimal value ($V^*(s)$) of a state s is the future dated strike relative price of the option. To compare with Black-Scholes, we must apply a discount factor ($e^{-r(T-t)}$) to obtain the present value of the strike relative price. To obtain the absolute price of the option, we must multiply the strike relative price by the strike price K. The option price at state s would then be given by $e^{-r(T-t)}KV^*(s)$.

Figure 3.8 plots the stock price, time-to-maturity and the price of a hypothetical European put option with a strike price of $K = 10$, and time-to-maturity of $T = 10$. These parameters will be used for all pricings throughout this dissertation. The error shown refers to the difference to the Black-Scholes analytic price. The result is promising because the accuracy of the KBRL approach compared to that of Monte-Carlo is significantly better (Figure 2.2). However, the problem still lies in the quadratic storage requirement for the transition probability matrix. Being able to use the same number of market states as the Monte-Carlo European put requires 75 gigabytes of memory to store *only* the transition probability matrix. This is a significant disadvantage

of using the KBRL approach to price options. The main advantage of using Monte-Carlo is the method requires only the final stock prices to be stored which is linear in the amount of storage needed.



(a) European put price matrix (KBRL-derived model with sell action)



(b) European put approximation error (KBRL-derived model with sell action)

Figure 3.8: European put price matrix with approximation error to the analytic Black-Scholes price matrix

3.2 Trajectory model using separated features and only hold action

3.2.1 Removing sell action

The initial model used a *sell* action providing the agent with the analytic Black-Scholes price as the reward for being in a particular state. The problem with using the *sell* action is it requires us to provide the agent with the correct price, when the purpose of the experiment is to calculate the correct price.

Theorem 3.1 proves that the optimal value of a state is equivalent to its fair value or price. This allows us to solve the European option pricing problem using our MDP. It also shows that the optimal policy for a European option is to *hold* to maturity. This result allows us to remove the *sell* action as it is always suboptimal[3].

Theorem 3.1. *Given the fair value of a European option on a stock U*

$$D_t^F = e^{-r(T-t)}\mathbb{E}\left[f(U_T)\middle|U_t\right] \tag{3.14}$$

and a policy π_H, which is the policy of holding the derivative to maturity ($\pi_H(s) = A_H$ for all $s \in S$), then $V^{\pi_H}(s) = e^{r(T-t)}D_t^F$ for all $t \in [0,T]$ and π_H is the optimal policy ($\pi^* = \pi_H$).

Proof. From (3.5)

$$\begin{aligned} V^{\pi_H}(s) =& \mathbb{E}\left[\sum_{j=0}^{n} R(s^{(j)}, A_H)\right] \\ =& \mathbb{E}\left[R(s_T, A_H)\right] \end{aligned}$$

where s_T is a terminal state because every non-terminal state gives a reward of zero under π_H. From the reward function $R(s,a)$ in (3.4)

$$\begin{aligned} R(s_T, A_H) =& f(U_T) \\ \Rightarrow \mathbb{E}\left[R(s_T, A_H)\right] =& \mathbb{E}\left[f(U_T)\middle|U_t\right]. \end{aligned} \tag{3.15}$$

[3]This is consistent with binomial tree approach which proves the fair value of a European option is the discounted expectation of the payoff at maturity (see [8]) meaning hold the option to maturity.

Then (3.14) and (3.15) imply that

$$V^{\pi_H}(s) = e^{r(T-t)} D_t^F. \tag{3.16}$$

Now, suppose π_H is not the optimal policy, i.e. there exists a state s_j for $0 \leq j < n$ where it is better to sell than to hold which implies

$$\begin{aligned} e^{r(T-t_j)} D_{t_j} &> V^{\pi_H}(s_j) \\ D_{t_j} &> e^{-r(T-t_j)} \left(V^{\pi_H}(s_j)\right) \\ &= e^{-r(T-t_j)} \left(e^{r(T-t_j)} D_{t_j}^F\right) \\ &= D_{t_j}^F. \end{aligned}$$

By no-arbitrage, $D_{t_j} = D_{t_j}^F$ for all $0 \leq j \leq n$, hence the state s_j cannot exist, and therefore π_H is an optimal strategy. □

This implies that $V^*(s) = e^{r(T-t)} D_t^F$ for all states s which means that learning the optimal value function, V^*, is equivalent to learning the accrued fair value function $e^{r(T-t)} D_t^F$. Therefore, the fair value for the derivative is calculated by discounting $V^*(s)$ with $e^{-r(T-t)}$.

3.2.2 Separating Risk/Time Component

Separating the risk/time component into a volatility feature and time-to-maturity feature is done to capture states that truly lie in the future. Potential successor states of current state should lie in the future of the current state because time moves forward. Therefore, the probability of moving to a state that lies in the past should be zero. In order to have transition probabilities to successor states that lie strictly in the future of the current state, the risk/time component needs to be separated. Previously, a complete market state of

$$m_t = \left(U_t/K, \sigma\sqrt{T-t}\right)$$

was used. Now, a market state of

$$m_t = (U_t/K, \sigma, T-t)$$

is used. This allows for comparisons between the time components to determine future states.

The risk (or volatility) components are compared using a standard Gaussian distribution while differences in time components uses a "forward-looking" Gaussian distribution[4]. Negative differences in time have a probability of zero. Differences in components are taken from the successor market state m' to the current market state m (i.e. $m'(\cdot) - m(\cdot)$). The weighted Euclidean distance function is defined as

$$d_W(m_1, m_2) = \sqrt{\left(\frac{U_1}{K_1} - \frac{U_2}{K_2}\right)^2 + (\sigma_1 - \sigma_2)^2 + \nu\left(T_1 - t_1 - T_2 + t_2\right)^2} \quad (3.17)$$

where $m_1 = \left(\frac{U_1}{K_1}, \sigma_1, T_1 - t_1\right)$, $m_2 = \left(\frac{U_2}{K_2}, \sigma_2, T_2 - t_2\right)$ are market states and

$$\nu = \begin{cases} 1 & \text{if } T_1 - t_1 > T_2 - t_2, \\ \infty & \text{if } T_1 - t_1 \leq T_2 - t_2. \end{cases} \quad (3.18)$$

This now poses a problem for the maturity states where the time to maturity component is zero (i.e. there are no states that strictly lie in the future from the maturity state). Using the payoff function for the option allows for the retrieval of the price for those maturity states.

The Market-States-Separate-Features procedure in Figure 3.9 performs exactly the same function as the Market-States procedure (Figure 3.2) except that the second feature of the market states M is separated into two features. Now, the initial market states for each $i = 1, 2, \ldots, n$ are

$$M_{i,1} = \left(\frac{s_0^i}{K}, \sigma, T - t\right)$$

and the successive markets for each $k = 1, 2, \ldots, m^i$ are

$$M_{i,k} = \left(\frac{s_k^i}{K}, \sigma, T - t_k^i\right).$$

[4]A forward-looking Gaussian distribution between two market states weights positive, smaller differences in the time component higher than positive, larger differences in the time component

```
MARKET-STATES-SEPARATE-FEATURES(n, r, lower, upper)
    n ← number of random options to generate
    r ← risk-free interest rate
    [lower, upper] ← range of transitions to generate from

 1  D ← DERIVATIVES(n)
 2
 3  transitions ← initialize to zero array of size n
 4  m ← n    // m is the total number of market states
 5  for i ← 1 to n do
 6      transitions(i) ← uniform random number on [lower, upper]
 7      m = m+transitions(i)
 8
 9  M ← initialize market state matrix to zero matrix of size m × 4
10  k ← 0
11  for i ← 1 to n do
12      strikeRelativeStock ← D(i, s_0)/D(i, K)        // s̄
13      volatility ← D(i, σ)                           // σ̄
14      timeToMaturity ← D(i, T) − D(i, t)             // T̄
15
16      M(k, s̄) ← strikeRelativeStock
17      M(k, σ̄) ← volatility
18      M(k, σ̄) ← timeToMaturity
19      M(k, σ̄) ← D(i) // Places derivative in 4th place holder
20
21      k ← k + 1
22      h ← (D(i, T) − D(i, t)) / transitions(i)
23      for j ← 1 to transitions(i) do
24          t' ← D(i, t) + j × h
25          M(k, s̄) ← BSFORMULA(M(k − 1, s̄), r, D(i, σ), h)
26          M(k, σ̄) ← D(i, σ)
27          M(k, T̄)D(i, T) − t'
28
29          d' ← (s_0, K, σ, t', T)
30          M(k, d̄) ← d'        // Places derivative in 4th place holder
31
32          k ← k + 1
```

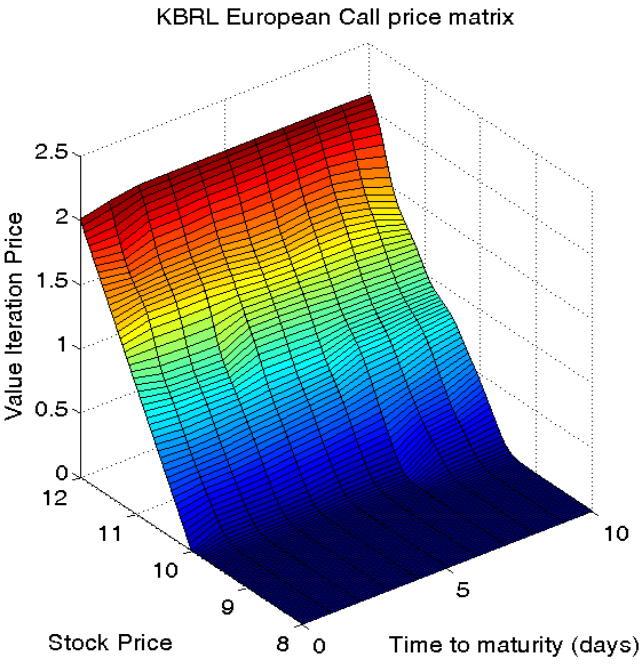
Figure 3.9: Market State sampling

3.2.3 Experiment (Trajectory model using separated features and only hold action): Pricing a European call option

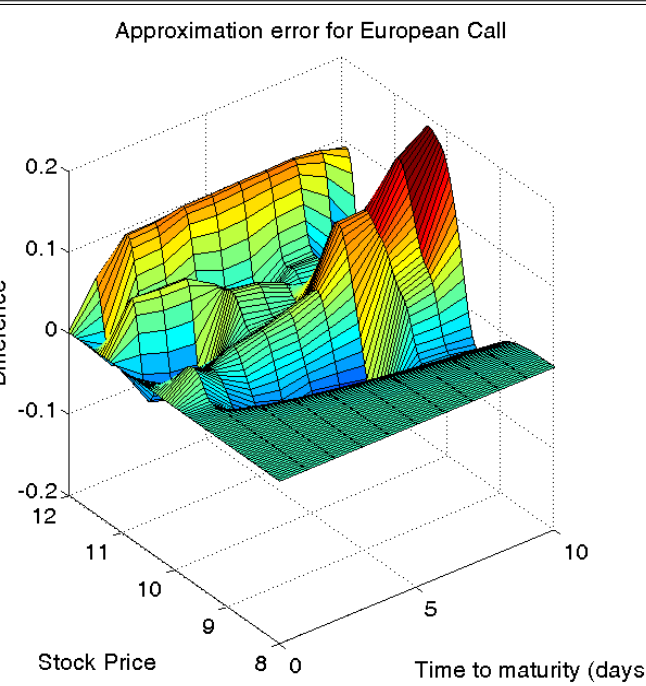
This experiment considers pricing a European call using Grassl's model[13] with the enhancements to this model discussed in the previous sections. The purpose of this experiment is not to compare the approximation error of this experiment to the previous experiment, the purpose is solely to show the results of pricing a European call option to the analytic Black-Scholes price. The following parameter values are used in solving for the value function using Kernel-Based Reinforcement Learning (KBRL):

- the number n of randomly sampled derivatives: $n = 1,500$;
- the riskless interest r: $r = 0.02$;
- the standard deviation b of the kernel functions: $b = 0.026$;
- the egg-shaped parameter ν applied to the distance function is given by (3.18);
- number of transitions per trajectory is an integer between 1 (inclusive) and 19 (inclusive); and
- the action to use for this model in pricing a European call is only the *hold* action (A_H).

The results of this experiment are shown in Figure 3.10 which shows a plot of the stock price, time-to-maturity and the price of a European call option and the approximation error to the Black-Scholes analytic solution. From the result, it can be observed that prices are being consistently overestimated for in the-money regions with the approximation error increasing rapidly as the time-to-maturity increases. The addition of more states will positively influence this result since the standard deviation of the kernels may be decreased.



(a) European call price matrix (KBRL-derived model)



(b) European call approximation error (KBRL-derived model)

Figure 3.10: KBRL Price matrix for European call (no sell action) with approximation error to analytic Black-Scholes price matrix

3.3 Mapped model using separated features and only hold action

The method used for generating trajectories used in the previous experiments generates samples that do not adequately cover the state space.

Figure 3.11 shows the mapping of the strike-relative stock to the risk/time component of samples in the first experiment. This (together with Figures C.2, C.3 and C.4) shows that as the number of trajectories increases, the samples cluster increasingly around a strike relative stock price of 1.

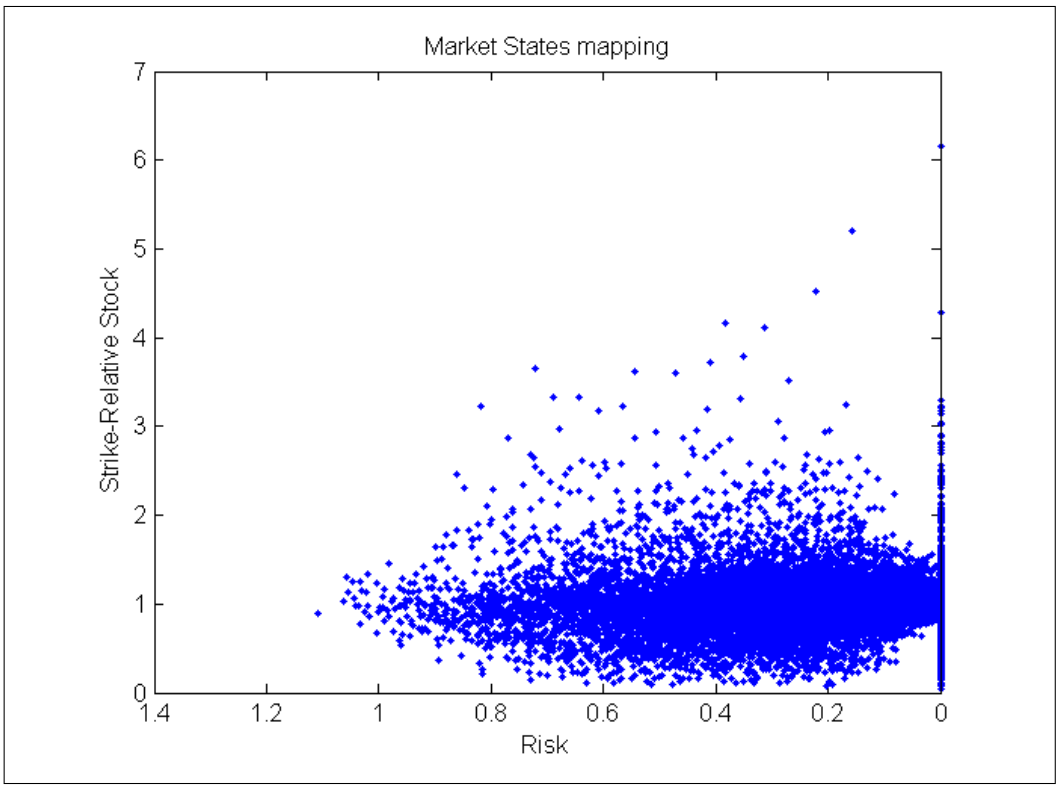


Figure 3.11: Market States mapping on 1000 sampled trajectories of strike-relative stock versus risk/time components

The potential problem with this model is that the space is not effectively covered because 68% of the effective stock prices lie in the interval $[0.8, 1.2]$, another 15% lie in $[0, 0.8)$ while the other 17% are above 1.2. This means the pricing of an option that lies in either of the 15% or 17% regions will suffer from a lack of close proximity states thereby either underestimating or overestimating the fair price.

This problem is not alleviated by the separation of the risk and time components, as these display the same characteristic, shown in Figures 3.12, 3.13, 3.14.

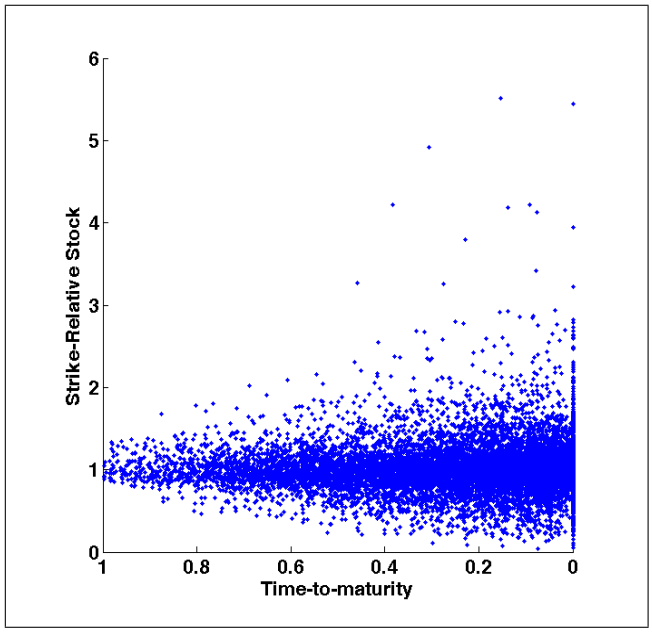


Figure 3.12: Stock vs Time to Maturity - separated risk/time components

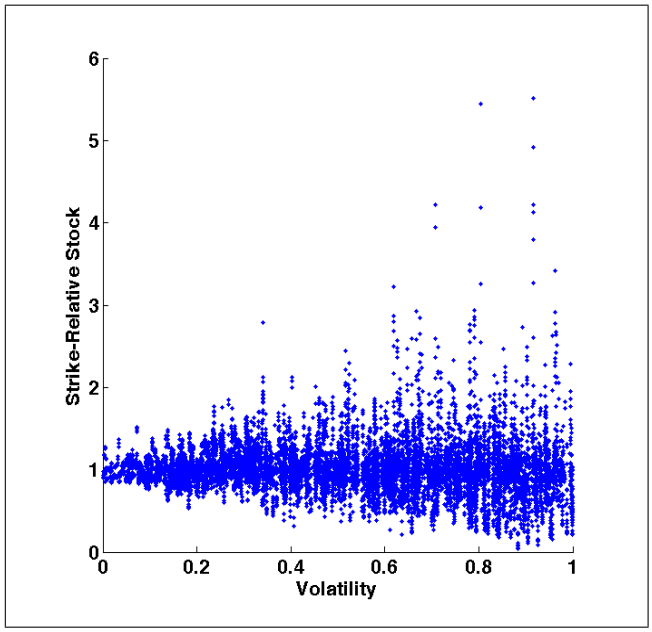


Figure 3.13: Stock vs Volatility - separated risk/time components

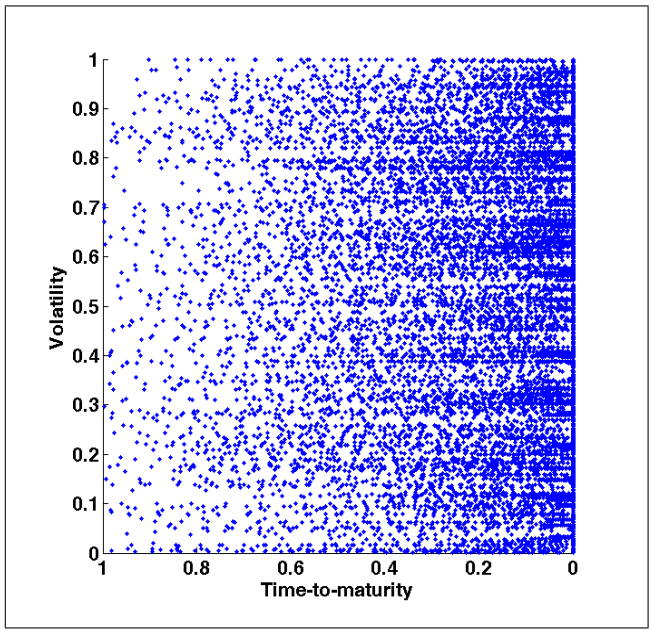


Figure 3.14: Volatility versus time-to-maturity - separated risk/time components

3.3.1 Mapping the state space

To achieve better state space coverage, we will sample individual market states instead of generating complete trajectories.

The generation of the market state space is achieved by generating n uniformly-spaced stock prices on the interval $[0, 2]$ and randomly sampling volatility and time-to-maturity on the interval $(0, 1]$ - see the procedure in Figure 3.15.

The procedure constructs a strike-relative stock price array of size n over the interval $[0, 2]$ and the interval step size is uniform of size $\frac{2}{n-1}$ (line 5). The total number of market states m is calculated, in line 4, by multiplying n by the mean of *lower* and *upper* (originally, *lower* and *upper* was the interval used to sample the size of each trajectory in the state space). Each nth market state has the same strike-relative stock price component (line 9) with randomly sampled volatility (line 10) and time-to-maturity (line 11) on the interval $[0, 1)$. The first n market states are assigned a time-to-maturity of zero (line 12) to guarantee at least n market states will receive an immediate reward.

Figures 3.16, 3.17 and 3.18 show the mapped state space across the three dimensions. This type of mapping is similar to the way market data is presented

```
MAPPED-MARKET-STATES(n, r, lower, upper)
    n ← number of random options to generate
    r ← risk-free interest rate
    [lower, upper] ← range of transitions to generate from

 1  k ← ⌈(upper + lower)/2⌉
 2  m ← n × (k + 1)
 3
 4  M ← initialize market state matrix to zero matrix of size m × 3
 5  s ← {0, 2/(n−1), 4/(n−1), ..., (2n−4)/(n−1), 2}
 6  for i ← 1 to n do
 7      for j ← 1 to k + 1 do
 8              index ← (i − 1) × (k + 1) + j
 9              M(index, 1) ← s(i)
10              M(index, 2) ← uniform random number on [0, 1)
11              M(index, 3) ← uniform random number on [0, 1)
12      M(i, 3) ← 0
```

Figure 3.15: Procedure to generate state space mapping

to traders whereby there is today's stock price with potentially hundreds of different strike prices around the stock price and differing time-to-maturity dates which tends to map very closely to these simulated mappings. In contrast to Figures 3.12, 3.13, 3.14, Figures 3.16, 3.17 and 3.18 illustrate how effectively the mapped model covers the state space. The key here is to avoid redundant states in the sampled state space instead of clustering states around the strike price.

3.3.2 Kernel function amendment

In order to calculate the transition probabilities between the uniformly distributed samples, we need to modify our kernel function. In this model, the distribution of the stock changes to lognormal because this needs to remain consistent with the Black-Scholes stock price process.

To achieve a lognormal distribution, the Brownian motion component needs

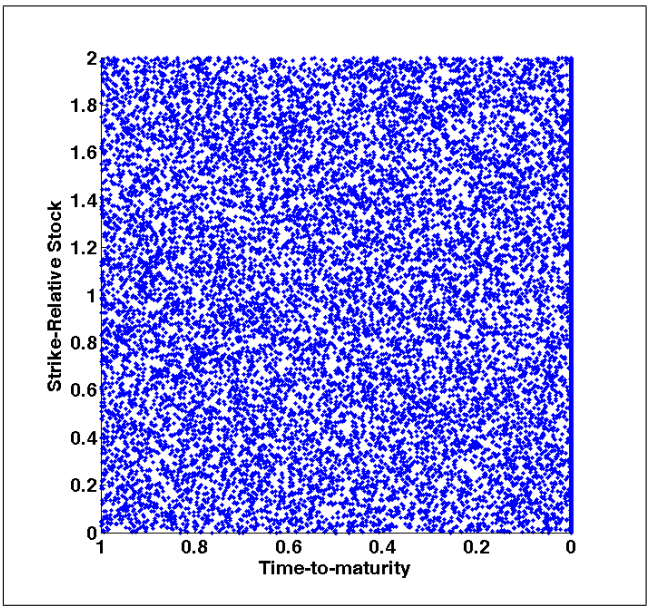


Figure 3.16: Stock vs Time-to-maturity - mapped model

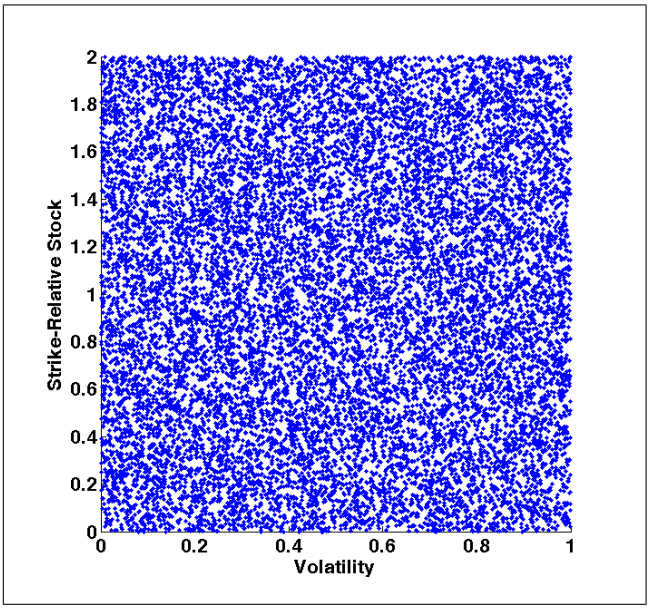
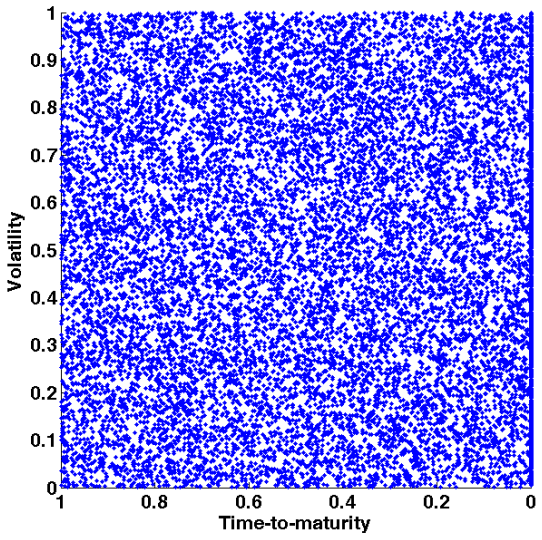


Figure 3.17: Stock vs Volatility - mapped model

to be calculated for a transition of (m, m'). From (2.4), Z can be defined as

$$Z = \ln\left(\frac{U'}{U} - \left(r - \frac{1}{2}\sigma^2\right)t\right) \tag{3.19}$$

where U' and U are the strike-relative stock price components of market states m' and m, respectively. The riskless interest rate is r, σ is the volatility



(a) Volatility vs. Time-to-maturity

Figure 3.18: Volatility versus Time-to-maturity - mapped model

component of market state m and the time lapse between market states m and m' is denoted by t where $t > 0$.

The standard Euclidean distance $d(U, U')$ between the strike-relative stock price components of the market states m and m' is defined as

$$d(U, U') = |U - U'|. \tag{3.20}$$

A lognormal distribution ($\bar{d}(U, U')$) may be obtained from $d(U, U')$ by multiplying the exponential component of the Black-Scholes stock price formula (from Equation (2.4)). More formally, $\bar{d}(U, U')$ is defined as

$$\bar{d}(U, U') = |U - U'| \exp\left(\left(r - \frac{1}{2}\sigma^2\right) t + \sigma Z\sqrt{t}\right) \tag{3.21}$$

where Z is calculated from (3.19). Figure 3.19 illustrates the nature of this curve for constant volatility ($\sigma = 0.4$) which reflects the same distribution as the standard Black-Scholes stock price distribution.

3.3.3 Generating Transition Probabilities

The transition probabilities are calculated (see Figure 3.20) from the current market state to every market state using the amended kernel function (from §3.3.2).

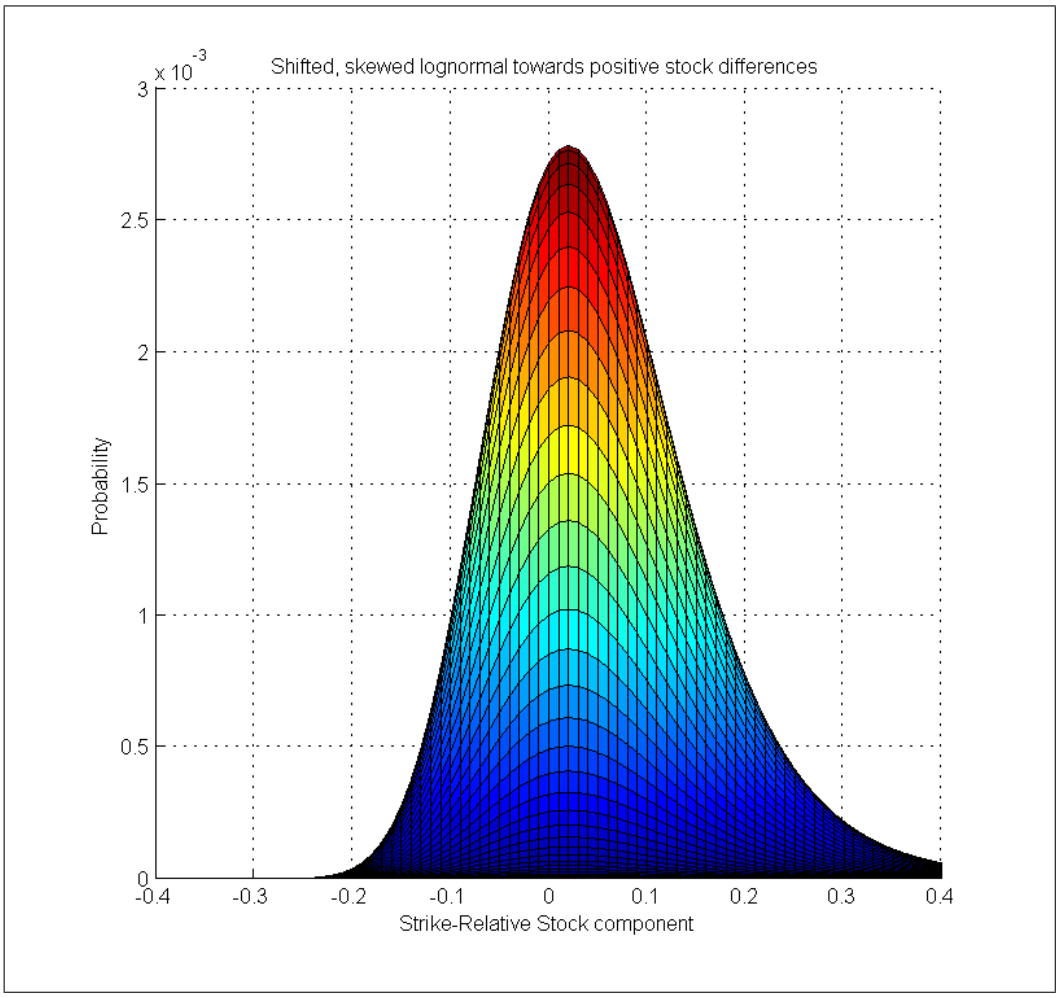


Figure 3.19: Shifted, lognormal Strike-Relative stock distribution

3.3.4 Experiment (Mapped model using separated features and only hold action): Pricing a European call

This experiment considers pricing a European call using the Mapped model discussed in previous sections. The following parameter values are used in solving for the value function using Kernel-Based Reinforcement Learning:

- sampled states using the procedure described in Figures 3.15 (results in roughly $16,000$ market states);
- the number of sampled stock prices n: $n = 1,450$;
- a riskless interest r: $r = 0.02$; and

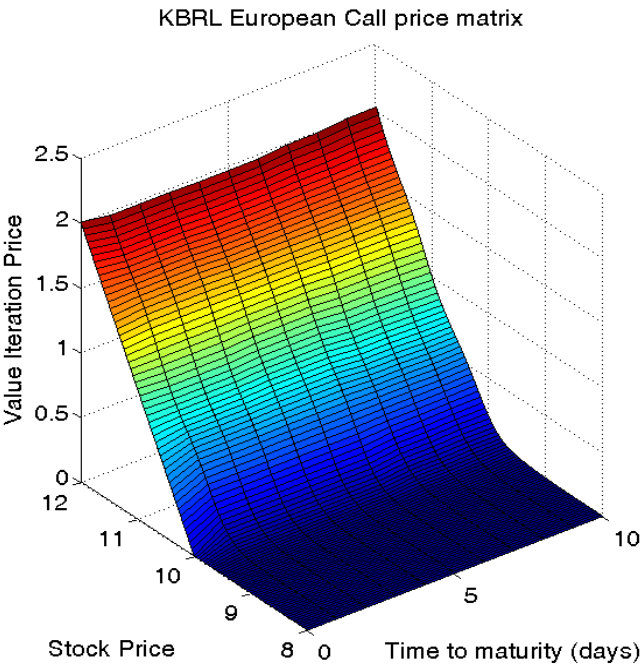
```
MAPPED-STATES-TRANSITION-PROBABILITIES(n, r, lower, upper)
    n ← number of random options to generate
    r ← risk-free interest rate
    [lower, upper] ← range of transitions to generate from

 1  M ← MAPPED-MARKET-STATES(n, r, lower, upper)
 2  m ← length of M
 3  T ← initialize to zero matrix of size m × m
 4  for i ← 1 to m do
 5      Z ← 0
 6      for j ← 1 to m do
 7          T(i, j) ← PHI(M(i), M(j), 0.07, ∞)
 8          Z ← Z + T(i, j)
 9      for j ← 1 to m do
10          T(i, j) ← T(i, j)/Z
11  return (T, M, m)
```

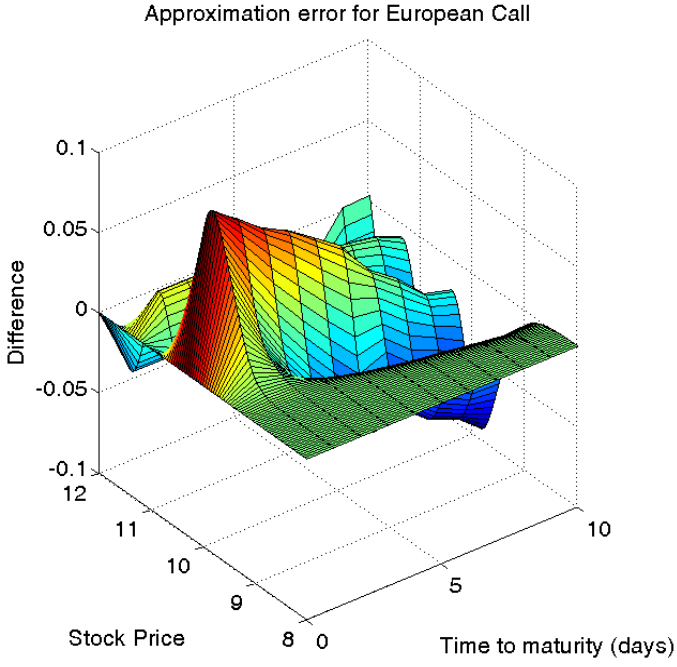
Figure 3.20: Mapped state space Transition Probability calculation

- the standard deviation b of the kernel functions: $b = 0.05$;

The results of the above experiment are shown in Figure 3.21 which shows the European call price matrix is consistently lower than the analytic Black-Scholes price matrix for in-the-money regions while the near-strike regions suffers from a high approximation error. However, this model exhibits a maximum error relative to the option price of approximately 1% which is significantly better than the Monte-Carlo method.



(a) European call price matrix (KBRL-derived mapped model)



(b) European call approximation error (KBRL-derived mapped model)

Figure 3.21: European call price matrix with mapped state space and approximation error to analytic Black-Scholes price matrix

3.4 Value Iteration procedure to obtain the value function and an optimal policy for the Option Pricing KBRL MDP

The VALUE-ITERATION procedure in Figure 3.23 is used to obtain the value function and an optimal policy for the KBRL MDP for each state and takes in eight input arguments:

- the number of random derivatives n to sample;
- the risk-free interest rate r;
- the *type* of the derivative (i.e. 'american' or 'european');
- $callPut$ flag indicating whether the option is a call or a put;
- a tolerance level ϵ for the value function;
- the minimum number of samples to have on a trajectory *lower*;
- the maximum number of samples to have on a trajectory *upper*; and
- set of actions available (only *hold* for 'european' and both *hold* and *exercise* for 'american').

The algorithm makes use of the calculated transition probability matrix, reward function, market states and actions to solve for an optimal value function. For European options, this is achieved by awarding rewards for all terminal states and progressing through each epoch assigning probabilistic returns for each state until the value function changes by a small amount (less than ϵ). The REWARD function used in VALUE-ITERATION is illustrated in Figure 3.22 and takes in two input arguments: the state s and the action a from the Kernel-Based Reinforcement Learning MDP. This procedure handles every scenario from initial Trajectory model to the improved models and also handles European and American options. The derivative d is extracted from the state s and the time-to-maturity of d is calculated. The following rules dictate the value returned to an agent:

- If the time-to-maturity is zero or the action a is to *exercise*, the (future) payoff is returned.

- If the action a is to *sell*, the future anayltic Black-Scholes price for the option is returned - this is the major reason for removing the *sell* action for American options.
- Otherwise, the reward is zero since the agent either does not have the derivative or the agent has decided to *hold* onto the derivative.

```
REWARD(s, a)
    s ← a state from the set of sampled market states
    a ← the action chosen in state s

1   d ← extract derivative from state s
2   T̄ ← d(T) − d(t)              // Time to maturity for derivative d
3   if T̄ = 0 or a = A_E then     // At maturity or exercise chosen ⇒ get
                                 //                              future payoff
4       return e^{rT̄} PAYOFF(d)
5   if a = A_S then              // sell action chosen ⇒ get future
                                 //                      Black-Scholes price
6       return e^{rT̄} BLACK-SCHOLES-PRICE(d)
7   return 0
```

Figure 3.22: Reward function

VALUE-ITERATION$(n, r, b, \nu, type, callPut, \epsilon, lower, upper, actions)$

$n \leftarrow$ number of random options to generate
$r \leftarrow$ risk-free interest rate
$b \leftarrow$ standard deviation of kernels
$\nu \leftarrow$ egg-shaped parameter
$tpye \in$ { 'american', 'european' }
$callPut \in$ { 'call', 'put' }
$\epsilon \leftarrow$ indicates the value iteration stopping condition
$[lower, upper] \leftarrow$ range of transitions to generate from
$actionSet \leftarrow$ if European $optionType \Rightarrow \{A_H\}$,
if American $optionType \Rightarrow \{A_H, A_E\}$

```
1  (T, M, m) ← TRANSITION-PROBABILITIES(n, r, lower, upper, b, ν)
2  V ← initialize to zero array of size m        // value function
3  Λ ← initialize to empty array of size m       // optimal policy
4  repeat
5          Δ = 0
6          for i ← 1 to m do
7             v ← V(i)
8             if M(i) is a terminal state then
9                    V(i) ← REWARD(M(i), A_H)
10                   A(i) ← A_H
11            else
12                   maxValue ← 0
13                   for a ∈ actionSet do
14                      if a ≠ A_H then
15                         p ← Σ_{j=1}^{m} T(i, j) REWARD(M(j), a)
16                      else
17                         p ← Σ_{j=1}^{m} T(i, j) V(j)
18                      if p > maxValue then
19                         Λ(i) ← a
20                         maxValue ← p
21                   V(i) ← maxValue
22            Δ ← max{Δ, |v − V(i)|}
23     until Δ < ε
24 return (V, Λ)
```

Figure 3.23: Value Iteration

3.5 Calculating a price for unseen option

The PRICE-SEPARATE-FEATURES function in Figure 3.24 is exactly the same as PRICE function except in the way the market state for the derivative is constructed. Since this is pricing a derivative using the separation of the risk feature, the constructed market state of the derivative needs to reflect this. Once the market state has been constructed, obtaining a price is the same methodology as calculating transition probabilities and then calculating the sum of those transition probabilities multiplied by the corresponding value from the value function.

PRICE-SEPARATE-FEATURES(M, V, d)
    $M \leftarrow$ sampled states of the market
    $V \leftarrow$ resulting value function (for each market state) of Value Iteration
    $d \leftarrow$ the derivative to price

```
 1  m ← initialize to zero array of size 4
 2  m(s̄_0) ← d(s_0)/d(K)
 3  m(σ̄) ← d(σ)
 4  m(T̄) ← d(T) − d(t)
 5  m(d̄) ← d
 6
 7  P ← 0
 8  Z ← 0
 9  for i ← 1 to m do
10      φ ← PHI(m, M(i), 0.13, 3.6)
11      P ← P + φ × V(i)
12      Z ← Z + φ
13  T̄ ← d(T) − d(t)
14  return e^{−rT̄} d(K) × P/Z
```

Figure 3.24: Determing price of previously unseen option

3.6 Summary

This chapter has formalised the European option pricing problem as a Markov Decision Process that is solvable via Kernel-Based Reinforcement Learning. An initial method was investigated and possible improvements presented due to the problem of the initial method requiring option prices for the sell action - these option prices are what the model is trying to learn, therefore the model should not have a sell action. The results from these improvements were closer to the analytic Black-Scholes price matrix than the initial method. In the following chapter, these methods form the basis of a solution to the American option pricing problem.

Chapter 4

American options

American options are amongst the hardest options to price as traders could exercise the option at any point during the life of the option. This makes American options more expensive than their European counterparts. Due to the complexity of American options there is no closed form solution available for pricing the option and thus practitioners have to resort to numerical approximations. American options have the structure whereby the exercise action receives a reward but no future return and the hold action receives no reward but *may* receive a future return. As previously discussed, the hold action can only benefit from future return because the trader only receives remuneration upon exercising.

Monte Carlo simulation is frequently used to price American options. The main problem with this approach is that for every option that needs to be priced, Monte Carlo simulation needs to be re-run to obtain that option's price. Another problem is that simulated paths cannot be re-used even if there are only slight changes to the volatility and time-to-maturity. Therefore new paths have to be generated which makes Monte Carlo simulation an expensive procedure.

This thesis will show that extending the KBRL-derived MDP used for European option pricing produces a viable model for pricing American options. This is attractive as once an optimal value function is obtained, the price for any American option can be calculated by a weighted average of the obtained value function by using the kernel function on the current option's market state and the existing market states. Essentially, this model allows us to reuse the generated kernel to price unseen options.

4.1 Changes to European option MDP

In order to price American options, we will modify the MDP shown in sections 3.2 and 3.3. Because American options allow the option to be exercised at any time before maturity, traders have one more action at their disposal. This *exercise* action gives the trader the ability to receive the payoff at any point on or before maturity. However, for the case where there are no dividends paid on a stock, it is never optimal to *exercise* prior to maturity.

If we were to have a *sell* action, the reward for selling the American option will be the true price of the American option (similar to that in the European case). However, the *sell* action is redundant. Consider an infinite state space for the American option MDP. The combination of the *hold* and *exercise* actions will produce American option prices that are equivalent to the true prices. The Monte Carlo procedure is based off the same principle of using *hold* and *exercise* actions: at each back track of the Monte Carlo simulation, either the discounted future value (i.e. *holding*) is higher than the *exercising* value or vice versa[12]. After Monte Carlo procedure terminates (i.e. after the complete back track from maturity to 'today'), a price for 'today' will be returned for a single American option[12]. Now, consider the finite state space for the American option MDP. When using the combination of the *hold* and *exercise* actions in the finite state space, the optimal value function will contain an approximate price to the true price. This is similar to the European option optimal value function which contains approximate prices to the true prices when using only the hold action. Therefore, the action space needs to include only the *hold* and *exercise* actions without any detriment to the effectiveness of the algorithm. That is,

$$a_t = \begin{cases} A_E & \text{if } D \text{ is being exercised,} \\ A_H & \text{if } D \text{ is being held.} \end{cases} \tag{4.1}$$

With the addition of the *exercise* action, the corresponding reward has been added to the reward structure (Equation (4.2))

$$R(s,a) = \begin{cases} D_T = f(U_T) & \text{if } t = T, \\ e^{r(T-t)} f(U_t) & \text{if } a = A_E \text{ and } t < T, \\ 0 & \text{if } a = A_H \text{ and } t < T. \end{cases} \tag{4.2}$$

4.2 Trajectory model using separated features

The improved trajectory model is discussed in §3.2. The only change here to that MDP lies in the action space and reward function as discussed in the previous section. The former has an extra action, namely *exercise* (A_E), and the latter includes the reward for choosing the *exercise* action in a state.

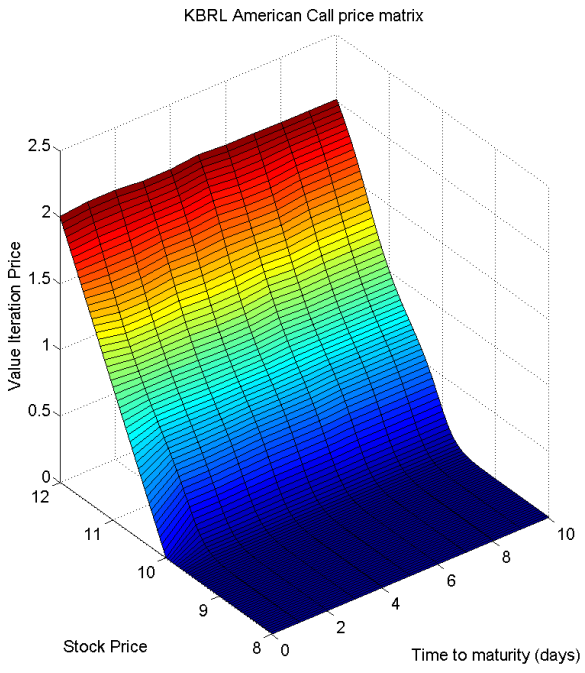
4.2.1 Experiment (Trajectory model using separated features): Pricing an American call option

This experiment considers pricing an American call option using the improved trajectory method discussed in §3.2 with the altered reward structure and action space as mentioned in §4.1. The KBRL MDP is constructed by generating samples in the manner outlined in §3.2.

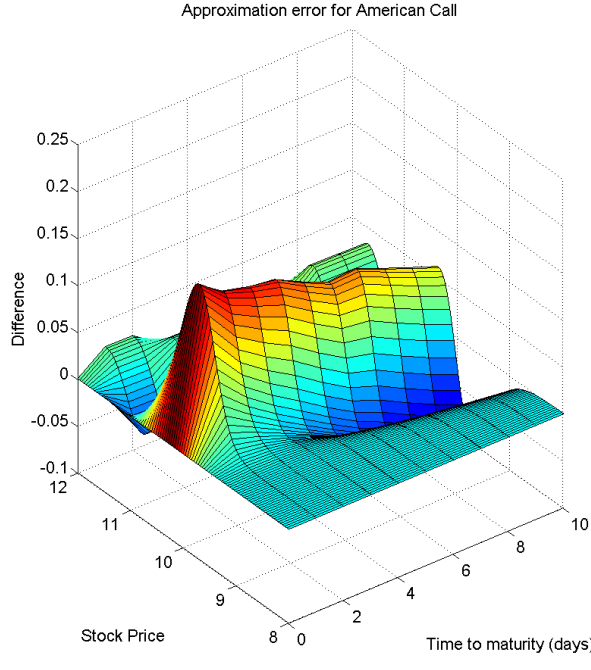
The following parameter values are used in solving for the value function using Kernel-Based Reinforcement Learning (KBRL):

- the number n of randomly sampled derivatives: $n = 1,500$;
- the riskless interest r: $r = 0.02$;
- the standard deviation b of the kernel functions: $b = 0.026$;
- the egg-shaped parameter ν applied to the distance function is given by (3.18);
- number of transitions per trajectory is an integer between 1 (inclusive) and 19 (inclusive); and
- the actions to use for this model in pricing an American call are the *hold* (A_H) and the *exercise* (A_E) actions.

The results of the above experiment are shown in Figure 4.1 which shows the American call price matrix is consistently higher than the Monte-Carlo price matrix for in-the-money regions and the approximation error rapidly increases the further from maturity the option is.



(a) American call price matrix (KBRL-derived model)



(b) American call approximation error (KBRL-derived model)

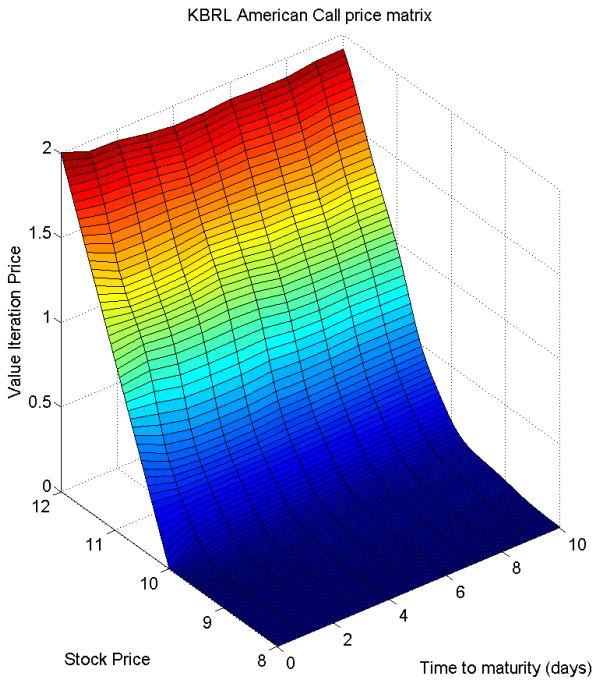
Figure 4.1: American call price matrix using KBRL with approximation error to the European call price matrix using the Analytic Black-Scholes approach

4.3 Experiment (Mapped model): Pricing an American call

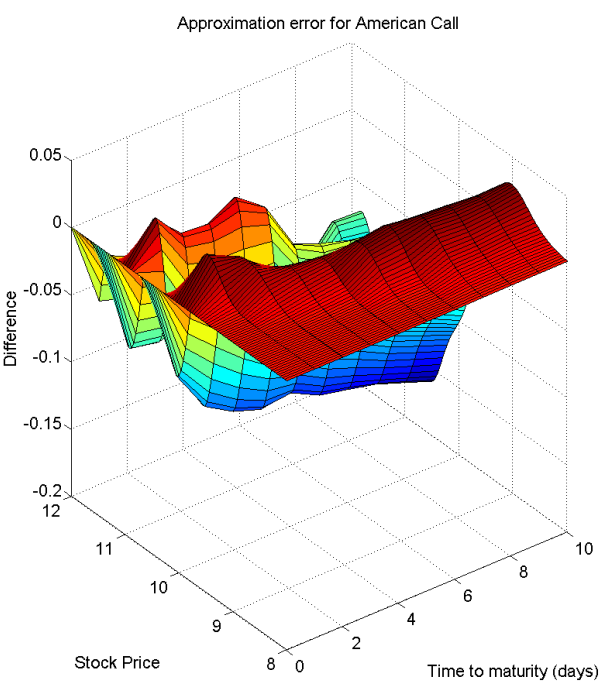
This experiment considers pricing an American call using the Mapped model discussed in §3.3 with the altered reward structure and action space as mentioned in §4.1. The following parameter values are used in solving for the value function using Kernel-Based Reinforcement Learning (KBRL):

- sampled states using the procedure described in Figure 3.15 (results in roughly $16,000$ market states);
- the number of sampled stock prices n: $n = 1,450$;
- a riskless interest r: $r = 0.02$;
- the standard deviation b of the kernel functions: $b = 0.05$;
- the egg-shaped parameter ν applied to the distance function is given by (3.18);
- the actions to use for this model in pricing an American call are the *hold* (A_H) and *exercise* A_E actions.

The results of the above experiment are shown in Figure 4.2 which shows the American call price matrix is consistently lower than the analytic Black-Scholes price matrix for in-the-money regions, however the approximation error relative to the option price is consistently below 1%.



(a) American call price matrix (KBRL-derived mapped model)



(b) American call approximation error (KBRL-derived mapped model)

Figure 4.2: American call price matrix using KBRL with mapped state space and approximation error to the European call price matrix using the Analytic Black-Scholes approach

Number of options priced	Monte Carlo simulation (on 10^5 paths)	Improved Trajectory model (on 10^5 states)	Mapped Model (on 10^5 states)
1	0.007196	0.000629	0.000644
10	0.065010	0.003947	0.003919
50	0.308740	0.015552	0.015611
100	0.622130	0.029282	0.027742
200	1.241843	0.055740	0.055961
500	3.092100	0.136024	0.136866
1000	6.175531	0.271860	0.271989
2000	12.341683	0.542383	0.544496
5000	30.878162	1.357726	1.363485
10000	61.861307	2.705847	2.728143
20000	123.862644	5.379056	5.507110
50000	310.130675	13.505553	13.732129

Table 4.1: Execution times (in seconds) of the Monte Carlo simulation approach in comparison to the improved Trajectory model and Mapped model

4.4 Analysis of execution time

One of the motivations for this research is Monte Carlo simulation is time consuming. This section examines how effective the KBRL models are in reducing the execution time and ultimately obtaining a similar American option price to the price calculated using Monte Carlo. Table 4.1 and Figure 4.3 show the execution times of the KBRL models in comparison to the Monte Carlo approach. The former models show similar results with a reduction over the latter approach. This reduction is significant because this means traders will be able to obtain an American option price in 4% of the time of the Monte Carlo approach and still obtain a similar price.

Therefore, the KBRL models are effective procedures for obtaining an accurate American option price (i.e. similar to Monte Carlo simulation) because these models are able to calculate the price in significantly less time.

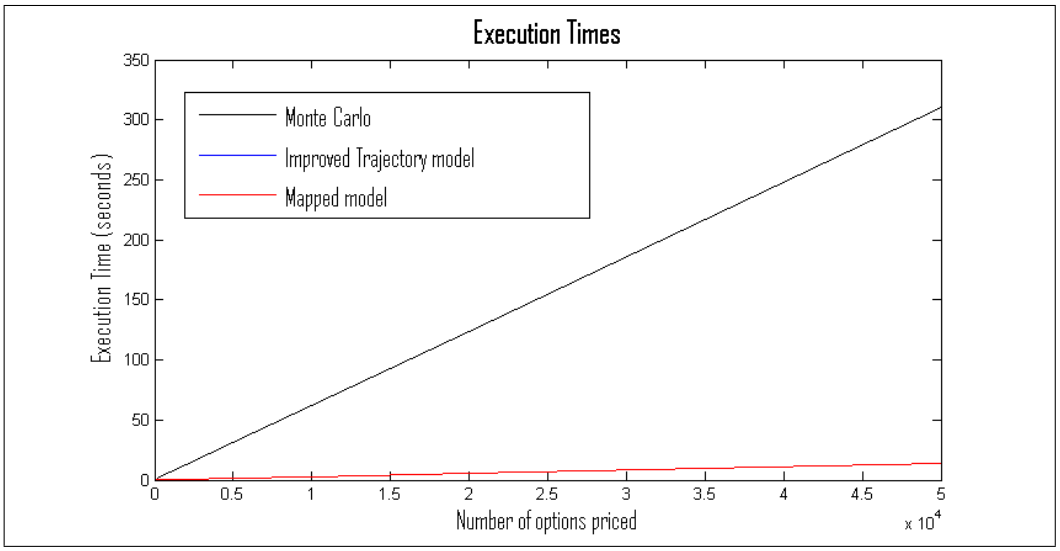


Figure 4.3: Execution times (in seconds) of the Monte Carlo simulation approach in comparison to the improved Trajectory model and Mapped model

4.5 Pricing an American call option on a dividend-paying asset

The previous sections have discussed American call options on non-dividend-paying assets which have equivalent prices to their European call option counterparts. American call options have a different price when the underlying asset pays out dividends. The simplest case is a stock that pays out continuous dividends. However, dividends are paid out at discrete times and generally dividend payouts are known in advance. Therefore, to convert the non-dividend-paying American option pricing MDP into a dividend-paying MDP, the present value of all dividends occurring from the value date to the expiry date of the option must be subtracted from the stock price.

4.5.1 Experiment (Trajectory model using separated features): Pricing an American call option on a dividend-paying asset

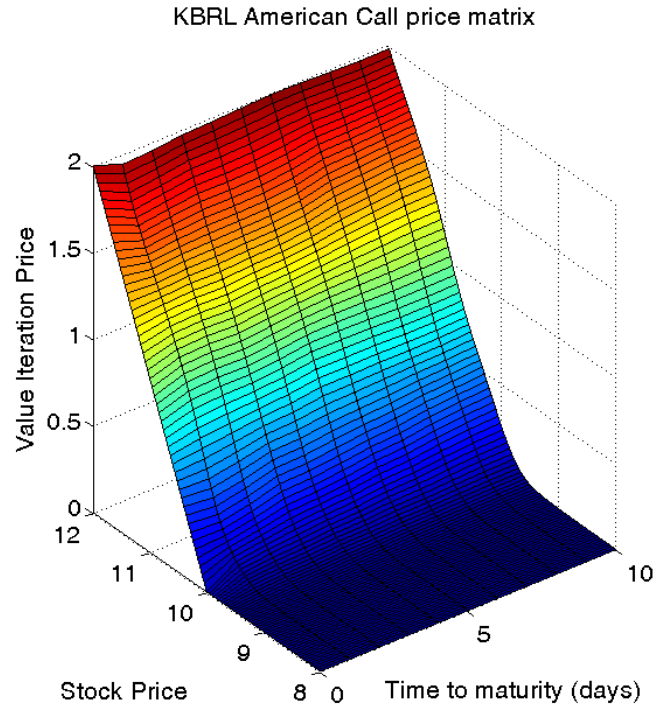
This experiment considers pricing an American call option on a dividend-paying asset using the improved trajectory method discussed in §3.2. The

KBRL MDP is constructed by generating samples in the manner outlined in §3.2.

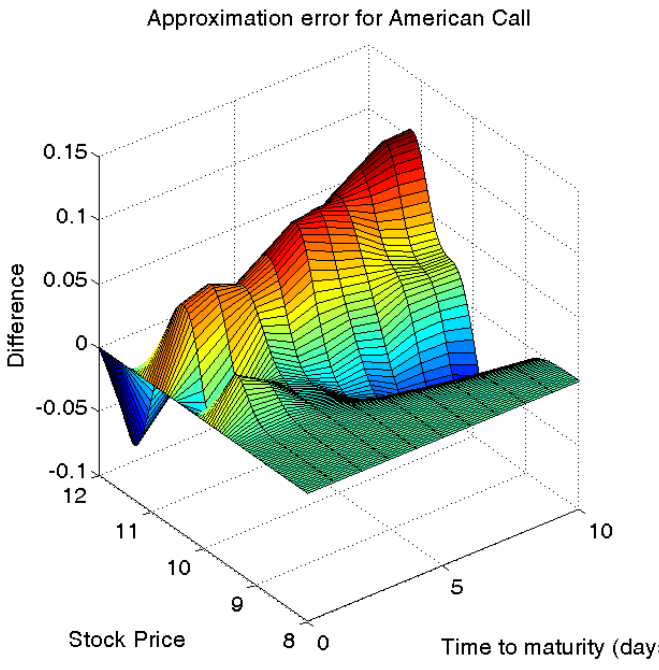
The following parameter values are used in solving for the value function using Kernel-Based Reinforcement Learning (KBRL):

- the number n of randomly sampled derivatives: $n = 1,500$;
- the riskless interest r: $r = 0.02$;
- a dividend yield q paid out quarterly: $q = 0.03$;
- the standard deviation b of the kernel functions: $b = 0.026$;
- the egg-shaped parameter ν applied to the distance function is given by (3.18);
- number of transitions per trajectory is an integer between 1 (inclusive) and 19 (inclusive); and
- the actions to use for this model in pricing an American call are the *hold* (A_H) and the *exercise* (A_E) actions.

The results of the above experiment are shown in Figure 4.4 which shows the American call price matrix is consistently higher than the Monte-Carlo price matrix for in-the-money regions and the approximation error rapidly increases the further from maturity the option is.



(a) American call price matrix (KBRL-derived model)



(b) American call approximation error (KBRL-derived model)

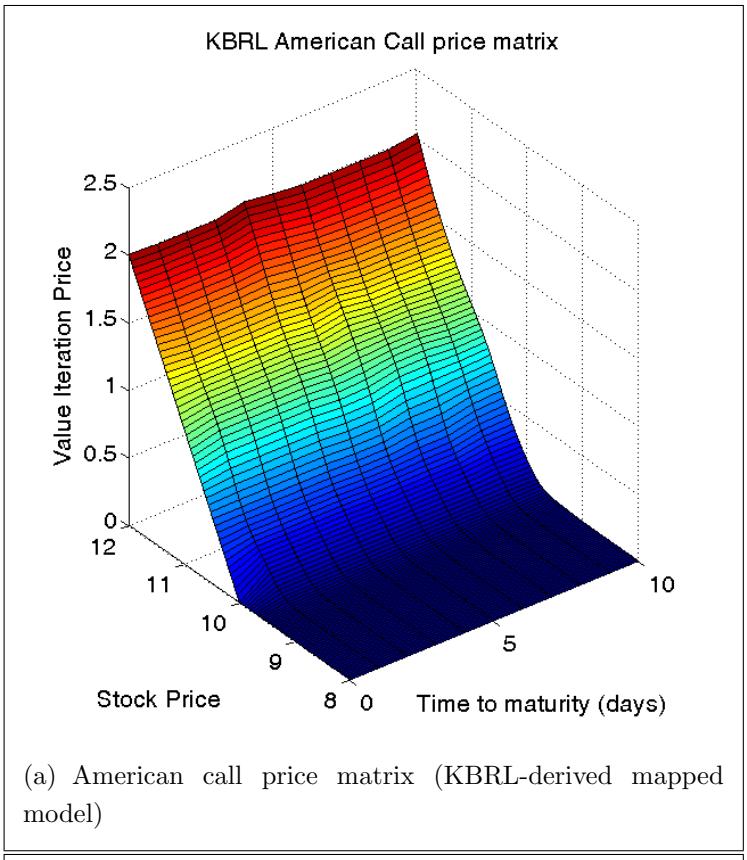
Figure 4.4: American call price matrix using KBRL with approximation error to the American call price matrix using Monte Carlo simulation

4.5.2 Experiment (Mapped model): Pricing an American call option on a dividend-paying asset

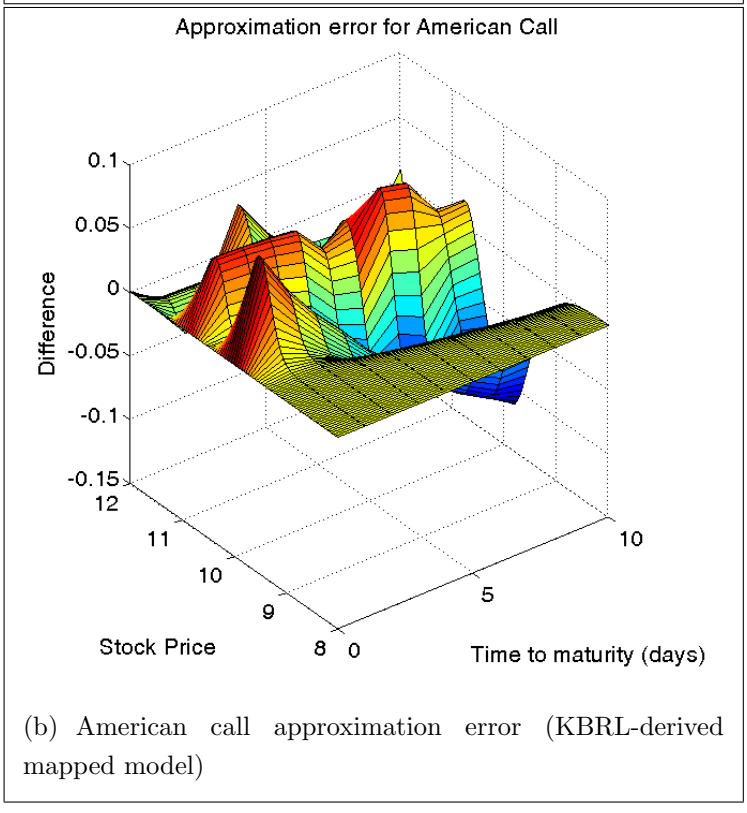
This experiment considers pricing an American call option on a dividend-paying asset using the Mapped model discussed in §3.3. The following parameter values are used in solving for the value function using Kernel-Based Reinforcement Learning (KBRL):

- sampled states using the procedure described in Figure 3.15 (results in roughly $16,000$ market states);
- the number of sampled stock prices n: $n = 1,450$;
- a riskless interest r: $r = 0.02$;
- a dividend yield q paid out quarterly: $q = 0.03$;
- the standard deviation b of the kernel functions: $b = 0.05$;
- the egg-shaped parameter ν applied to the distance function is given by (3.18);
- the actions to use for this model in pricing an American call are the *hold* (A_H) and *exercise* A_E actions.

The results of the above experiment are shown in Figure 4.5 which shows the American call price matrix is consistently higher than the Monte-Carlo price matrix for in-the-money regions, however the approximation error relative to the option price is consistently below 1%.



(a) American call price matrix (KBRL-derived mapped model)



(b) American call approximation error (KBRL-derived mapped model)

Figure 4.5: American call price matrix using KBRL with mapped state space and approximation error to the American call price matrix using Monte-Carlo simulation

4.6 Summary

This chapter has shown that a simple addition to the action space and reward function transforms the European option pricing MDP into an American option pricing MDP. Experiments were performed on the improved trajectory model and the mapped model illustrating desirable results when compared to Monte Carlo simulation. Also, the execution times of these two models were compared to the Monte Carlo simulation approach. Promising results were found showing the KBRL models are effective procedures for reducing the time needed to obtain a price for an American option. Furthermore, an American call option price matrix on a dividend-paying asset was compared to the equivalent Monte Carlo price matrix indicating the similarities in price between the two models.

The following chapter investigates the use of inputting market data into the models instead of generated market states and comparing the results to the respective existing methods.

Chapter 5

Calibrating models

Calibration of a model in finance is a term used to describe the insertion of market data into a financial model to "transform" the price the model calculates into a market price. For both European and American options, dividends need to be incorporated into the KBRL MDP to calculate a market price. The market data used for the improved trajectory model is the daily historical stock prices and the historical dividends for the list of companies in Table 5.1. This market data was retrieved from Yahoo Finance who provide adjusted daily close prices based on dividends from inception of the company to 8-Jan-2013. For the mapped model, pricing a call option requires European call option market data which was retrieved from Yahoo Finance for 8-Jan-2013 for the list of companies in Table 5.1. The maturity dates of the European call options is in Table 5.2.

5.1 Calibrating improved trajectory model using underlying stock trajectories

To calibrate this model, existing stock price paths from the market need to be extracted. The historical volatility on the returns of the stock can then be calculated (Figure 5.1) allowing us to replace the trajectories generated by the Black Scholes process with recorded market trajectories.

Companies	Symbol
Apple Incorporated	AAPL
Amazon.com Incorporated	AMZN
Ford Motor Company	F
Facebook Incorporated	FB
Google Incorporated	GOOG
Hewlett-Packard Company	HPQ
JPMorgan Chase & Company	JPM
McDonald's Corporation	MCD
Morgan Stanley	MS
Microsoft Corporation	MSFT
Starbucks Corporation	SBUX
Yahoo! Incorporated	YHOO

Table 5.1: List of companies for which market data was extracted from Yahoo Finance

Maturity dates
10-Jan-2013
18-Jan-2013
24-Jan-2013
31-Jan-2013
15-Feb-2013
15-Mar-2013
17-Mar-2013
19-Apr-2013
21-Jun-2013
19-Jul-2013
16-Aug-2013
20-Sep-2013
18-Oct-2013
17-Jan-2014
16-Jan-2015

Table 5.2: Maturity dates of European call options from 8-Jan-2013

HISTORICAL-VOLATILITY-ON-RETURNS-OF-STOCK(m, S)

$S \leftarrow$ a stock price path

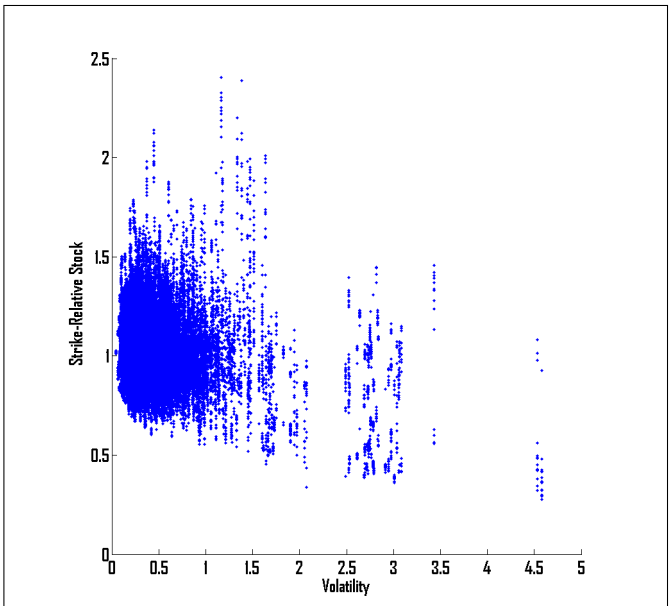
1 $m \leftarrow$ the number of stock prices in S
2 $log\text{-}results \leftarrow$ initialize to a vector of size $m-1$
3 $mean \leftarrow 0$
4 **for** $i \leftarrow 1$ **to** $m-1$ **do**
5 $\quad log\text{-}results(i) \leftarrow \ln\left(\frac{S(i+1)}{S(i)}\right)$
6 $\quad mean \leftarrow mean + log\text{-}results(i)$
7 $mean \leftarrow mean/(m-1)$
8 $\sigma \leftarrow 0$
9 **for** $i \leftarrow 1$ **to** $m-1$ **do**
10 $\quad \sigma \leftarrow \sigma + (log\text{-}results(i) - mean)^2$
11
12 **return** $\sqrt{\frac{252\sigma}{m-1}}$ // there are 252 average
// number of business days a year

Figure 5.1: Procedure to calculate Historical Volatility

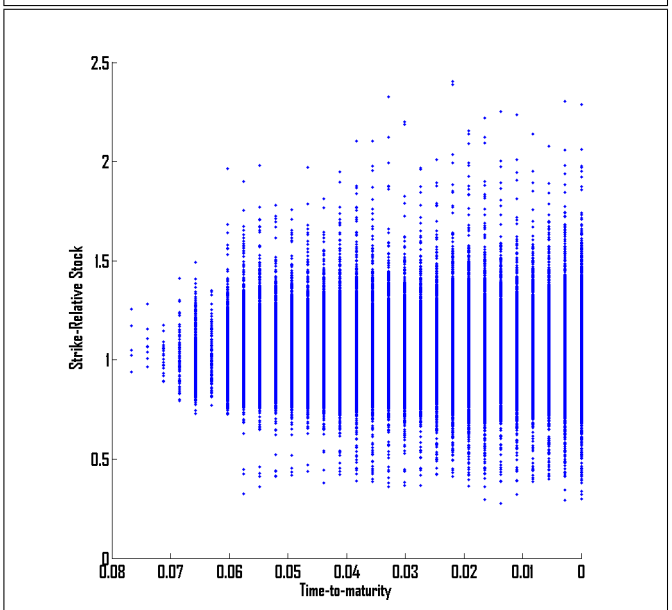
5.1.1 Replacing simulated Market State trajectories

Historical prices for the major stocks on the New York stock exchange were used for constructing the trajectories. Each trajectory has sixteen *daily* stock prices and historical volatility is calculated on each trajectory. The trajectory, together with the differing time-to-maturities of each option on the trajectory, including the historical volatility are merged together to form the market states. The strike prices for the options generated from the underlying stocks are chosen from a normal distribution centered around the stock price, as shown in Section 5.1.2

The stock versus volatility graph (Figure 5.2(a)) indicates approximately sixty percent of the volatility is close to zero whereas both the stock versus time-to-maturity graph (Figure 5.2(b)) and the volatility versus time-to-maturity graph (Figure 5.3) show the uniformity that we would expect when extracting daily stock price trajectories.



(a) Stock vs. Volatility



(b) Stock vs. Time-to-maturity

Figure 5.2: Market Data Stock state space mapping - separated risk/time components

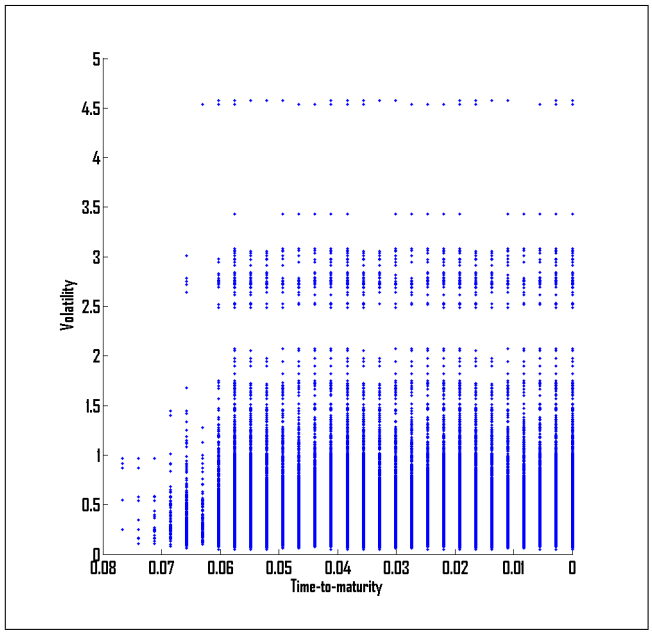


Figure 5.3: Volatility versus time-to-maturity state space mapping - separated risk/time components

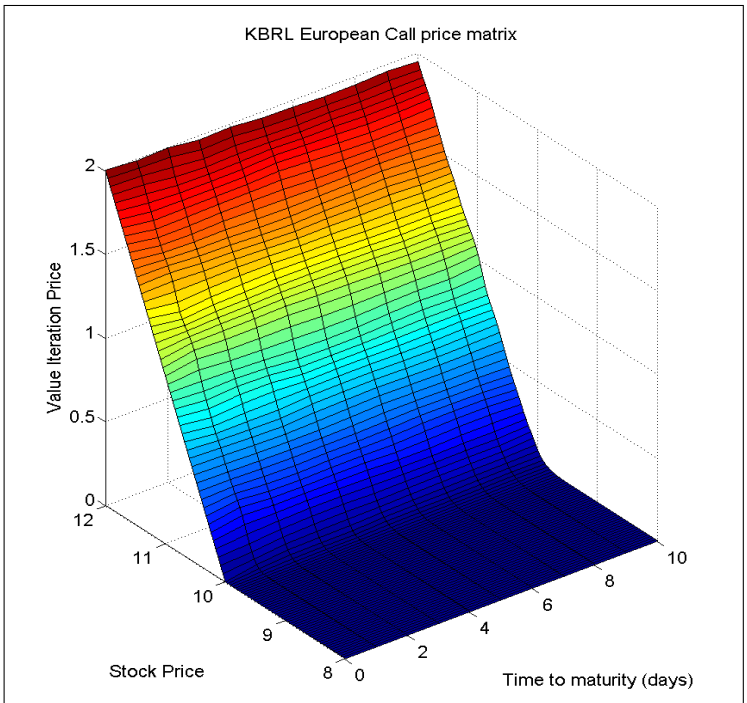
5.1.2 Experiment (calibrating the improved Trajectory model): Pricing a European call and an American call

The experiment considers calibrating the improved Trajectory model to existing historical market data for a European call and an American call using the improved Trajectory model (§3.2). The following parameter values are used in solving for the value function using Kernel-Based Reinforcement Learning (KBRL):

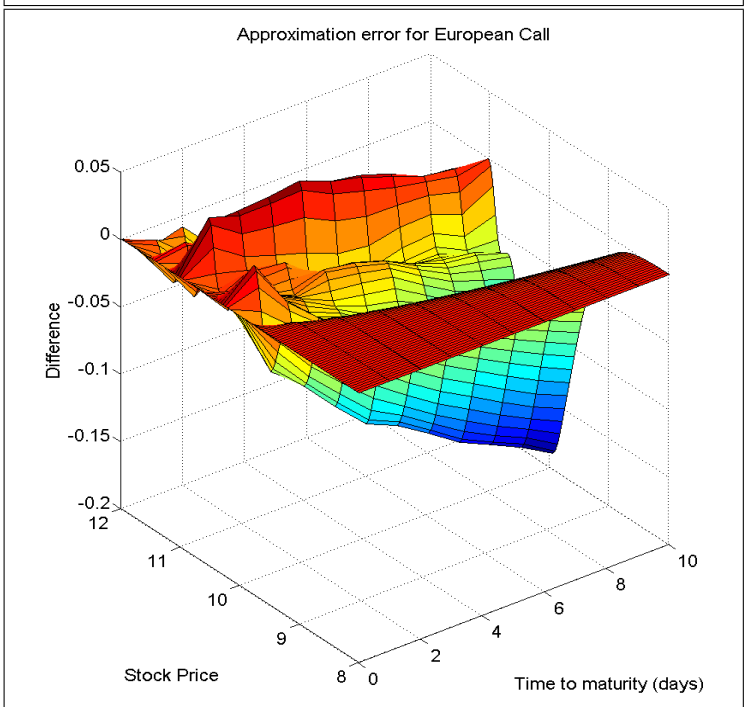
- existing historical market data on stocks are used to get $n = 3,000$ stock price trajectories;
- interval used to determine number of transitions per trajectory: $[1, 19]$ (with $n = 3000$, results in roughly $36,000$ market states);
  - number of time steps t_s for each trajectory is $t_s = \frac{19+1}{2} + 1 = 11$;
  - $t_s = 11$ time steps means that there are $m \approx 12$ stock prices on a trajectory; and
  - $n \times m \approx 3,000 \times 12 = 36,000$.

- the USD zero curve is used to calculate riskless interest rates r (see Figure B.3 for the USD zero curve);
- the known and forecast dividend payouts for each stock can be found in Figure B.4 and Figure B.5;
- volatility of each trajectory is calculated by using the procedure defined in Figure 5.1;
- the strike price K : $K = U_0 \times (1 - 0.12 \times \mathcal{N}(0, 1))$;
- the standard deviation b of the kernel functions: $b = 0.013$;
- the egg-shaped parameter ν applied to the distance function is given by (3.18); and
- the action to use for this model in pricing a European call is only the *hold* (A_H) action; the actions to use for this model in pricing an American call is the *hold* (A_H) and *exercise* A_E actions.

The results of the above experiment are shown in Figure 5.4 for European calls and Figure 5.5 for American calls. The maximum error relative to the option price is approximately 5% due to the overestimated prices the further the option is from maturity.

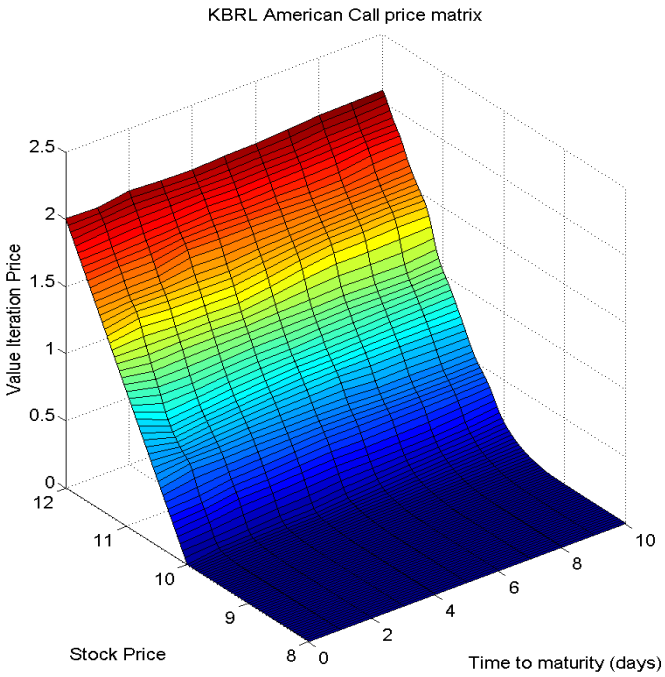


(a) Calibrated European call price matrix (KBRL-derived model)

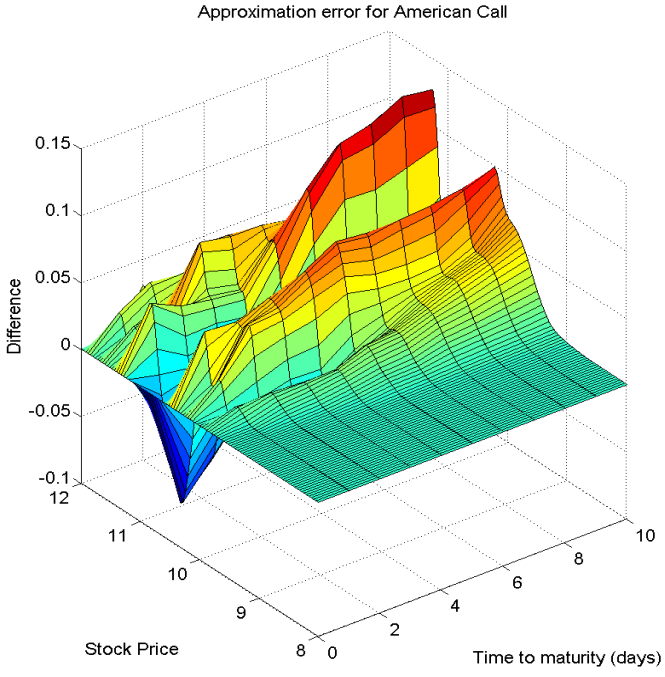


(b) Calibrated European call approximation error (KBRL-derived model)

Figure 5.4: Calibrated European call price matrix using market data trajectories and historical volatility with approximation error to Analytic Black-Scholes European call



(a) Calibrated American call price matrix (KBRL-derived model)



(b) Calibrated American call price matrix (KBRL-derived model)

Figure 5.5: Calibrated American call price matrix using market data trajectories and historical volatility with approximation error to Monte-Carlo Black-Scholes American call

5.2 Calibrating Mapped model using Option data

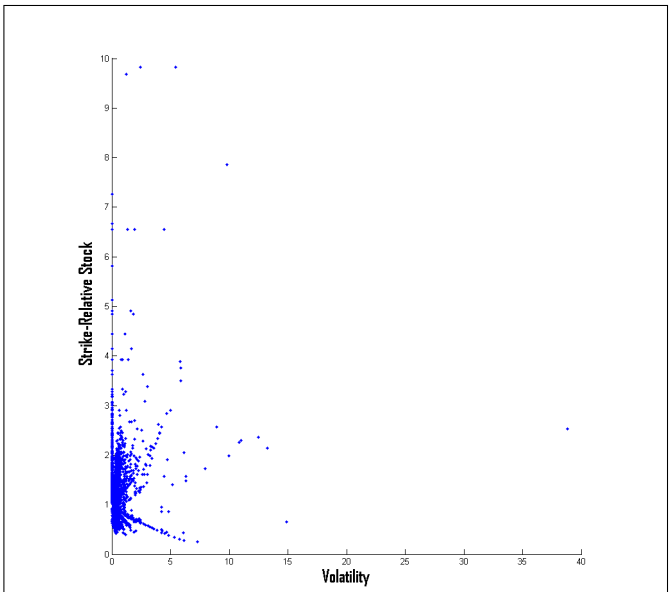
To calibrate this model, current market prices for various options can be used as input to the model. The market prices are not used directly in our kernel as these prices are susceptible to mispricings. Instead, we use these prices to calculate the implied volatility of the stock using the Black-Scoles formula, and we use value iteration on the option to estimate its fair price.

Once the implied volatility has been calculated, parameters visible from the market together with the calculated volatility can be used as a market state. The collection of these market states makes up the state space. The model requires no additional change to achieve calibration to the market. Figures 5.6(a), 5.6(b) and 5.7 illustrates the coverage of the European option price market data.

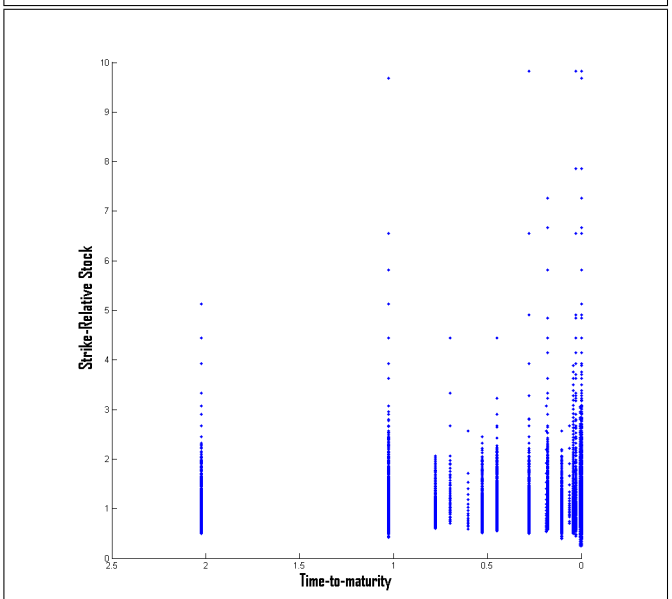
5.2.1 Experiment (calibrating Mapped model): Pricing a European call and an American call

The experiment considers calibrating the Mapped model (§3.3) to existing European option prices for a European call and American call. The following parameter values are used in solving for the value function using Kernel-Based Reinforcement Learning (KBRL):

- existing market data on European options are used to map the state space (the number of resultant market states is roughly $n = 11,000$);
- the USD zero curve is used to calculate riskless interest rates r (see Figure B.3 for the USD zero curve);
- the known and forecast dividend payouts for each stock can be found in Figure B.4 and Figure B.5;
- volatility of each trajectory is calculated by using the procedure defined in Figure 5.1;
- the standard deviation b of the kernel functions: $b = 0.026$;
- the egg-shaped parameter ν applied to the distance function is given by (3.18); and



(a) Stock vs. Volatility



(b) Stock vs. Time-to-maturity

Figure 5.6: Market Data stock state space mapping - separated risk/time components

- the action to use for this model in pricing a European call is only the *hold* (A_H) action; the actions to use for this model in pricing an American

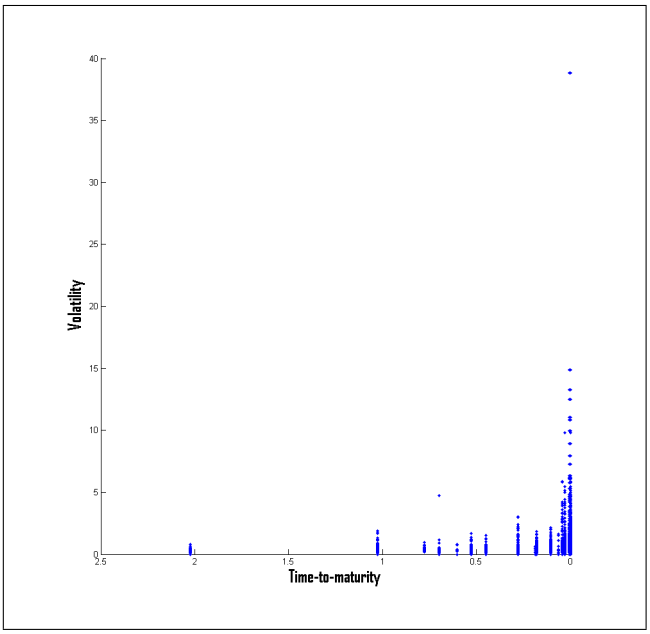
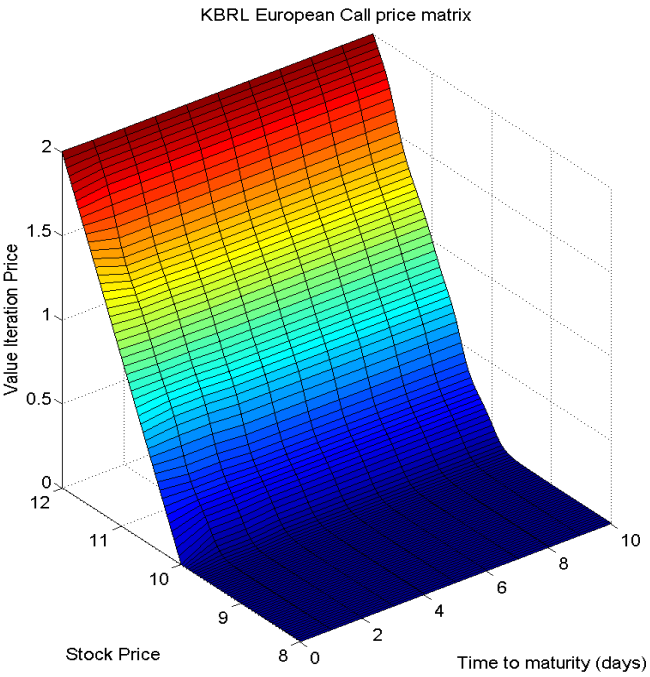


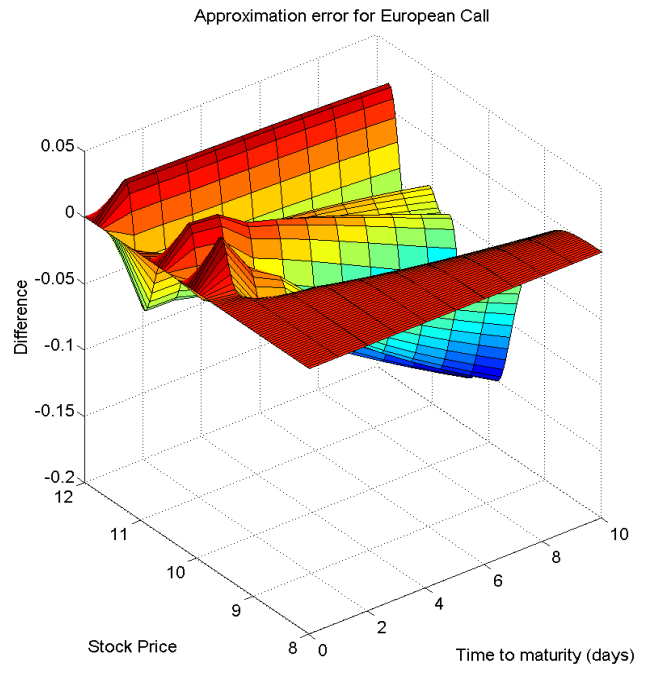
Figure 5.7: Volatility versus time-to-maturity state space mapping - separated risk/time components

call is the *hold* (A_H) and *exercise* A_E actions.

The results of the above experiment are shown in Figure 5.8 for European calls and Figure 5.9 for American calls. The approximation error for the European call price matrix remains below 1% for in-the-money regions, however near-strike has an approximation error of around 5%. Whereas the approximation error for the American call price matrix is significantly better having a maximum relative error of less than 1% across all regions.

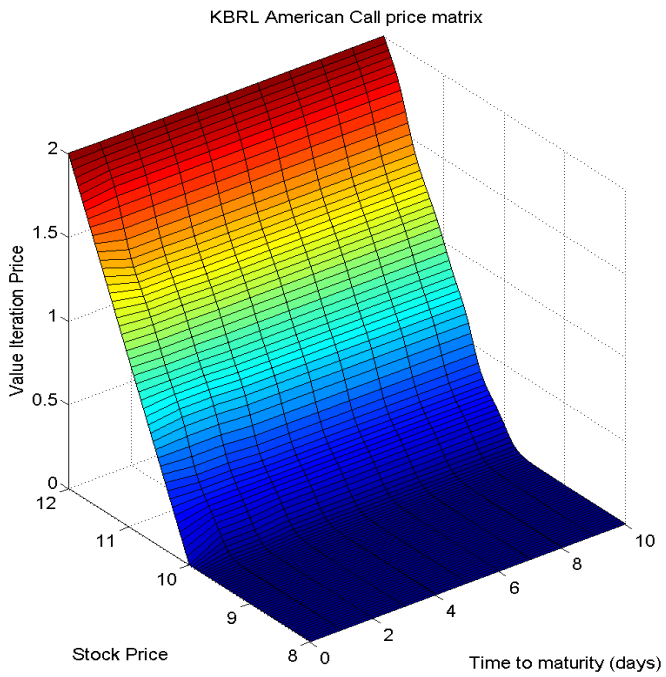


(a) Calibrated European call price matrix (KBRL-derived model)

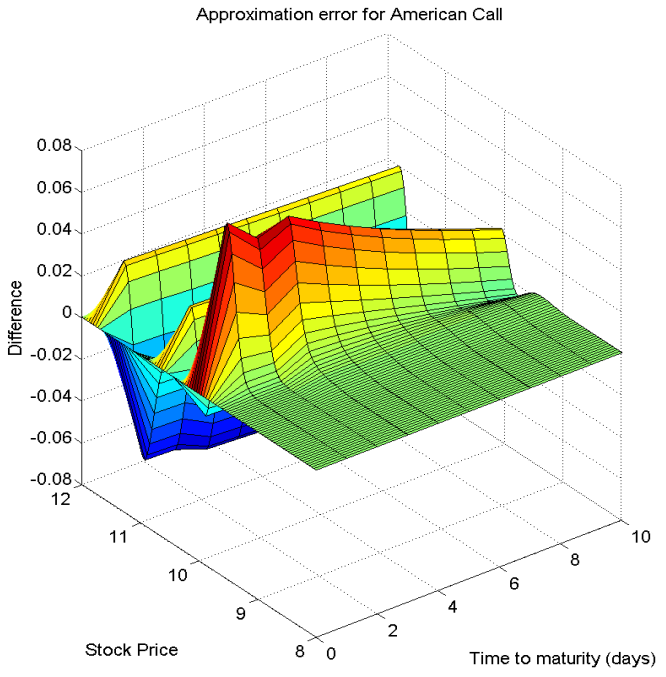


(b) Calibrated European call approximation error (KBRL-derived model)

Figure 5.8: Calibrated European call price matrix with approximation error to analytic Black-Scholes price



(a) Calibrated American call price matrix (KBRL-derived model



(b) Calibrated American call approximation error (KBRL-derived model

Figure 5.9: Calibrated American call price matrix with approximation error to Monte-Carlo Black-Scholes price

5.2.2 Comparison to market data prices

The market states used in the calibrated model are constructed directly from market data. Absolute and relative errors, Equations (5.1) and (5.2), respectively, may be calculated between the learned market states' prices and their market data prices to determine the effectiveness of the model. The absolute and relative errors are defined by

$$\epsilon = |v - v_{approx}| \tag{5.1}$$

and

$$\eta = \frac{\epsilon}{|v|} = \frac{|v - v_{approx}|}{|v|}, \tag{5.2}$$

respectively. The market data prices exhibit a maximum absolute error of 13.93 and a maximum relative error of 21.47. Furthermore, statistical analysis is applicable here to find the mean of the market data price differences to KBRL prices and the standard deviation around the mean. The mean is defined by

$$\mu = \frac{1}{n}\sum_{i=1}^{n} |P_i - V_i| \tag{5.3}$$

where n is the total number of market data option prices and $|P_i - V_i|$ is the absolute difference between the market data price and KBRL price of the ith market data option. The standard deviation around the mean μ is given by

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left(|P_i - V_i| - \mu\right)^2}. \tag{5.4}$$

Given Equations (5.3) and (5.4), the mean of the absolute differences in prices is 0.14 and the standard deviation around the mean is 8.2436×10^{-9}. The attributing factor to these numbers is the consistency of the gathered market data (shown in Figure 5.10).

The solid line is the payoff function for a call option and also represents the intrinsic value for any option with a strike price of one. Without considering dividend payouts made out by the stocks, inconsistencies will arise in the resultant date around intrinsic value versus fair value. That is, the intrinsic value of a derivative may be higher than the option's market price. This is evident in certain pieces of market data such as the Apple stock option with an unadjusted closing share price (i.e. not considering dividends) of $525.31

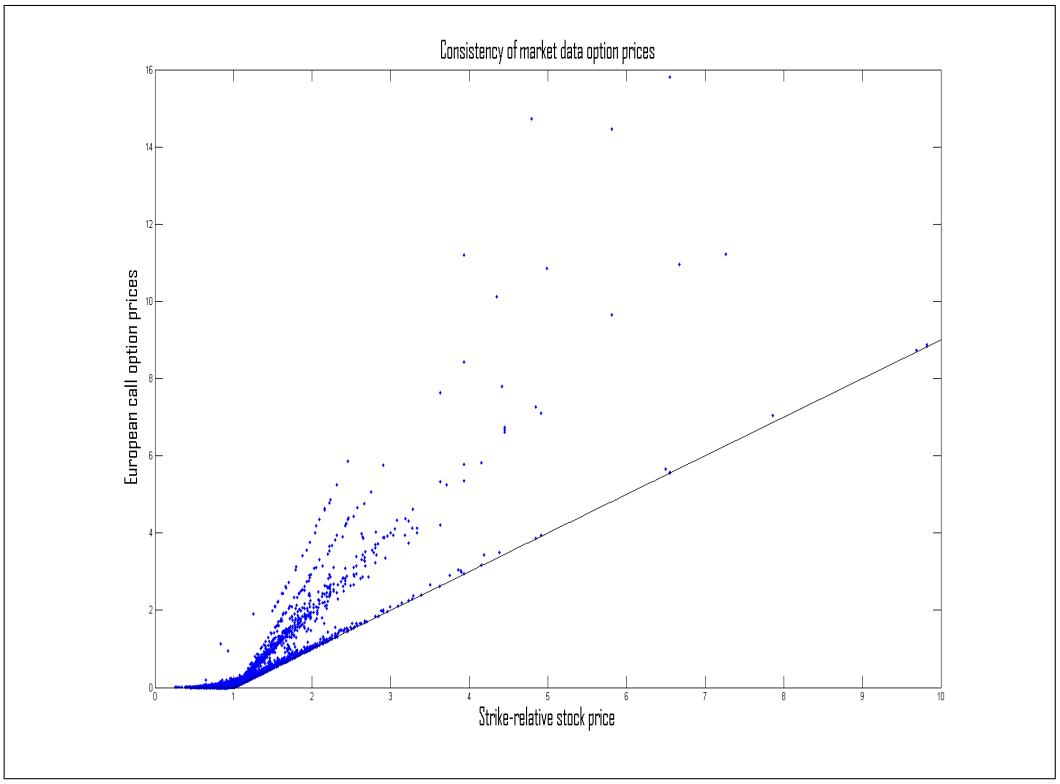


Figure 5.10: Strike-relative market data prices illustrating the consistency of the market data

for 8 January 2013, a strike price of \$320, a time-to-maturity of 0.2767 years and a call option price of \$202.83. Here, the intrinsic value (i.e. payoff) at this time is \$205.31 which is \$2.48 higher than the market's price for the call option. After taking dividends into consideration, the market is providing fair values higher than the intrinsic value for all of the gathered options.

The use and construction of the market states from the market data is just as easy as sampling the market data using random number generators. This is more difficult to do in current numerical procedures as a least squares approximation is used to minimise the error.

5.3 Summary

Calibration has been explained to be a technique used in finance to "transform" a price calculated by a model into a market price by inserting market data into the model. The improved Trajectory model and the Mapped model

have been calibrated to solve for a European and American option market price. Comparisons of European option prices were done to the existing market prices illustrating prices accurate to within 1% of the actual value. This was evident through statistical analysis by calculating the mean of the absolute differences in the prices calculated by the Mapped model and the prices from the market. The standard deviation around the mean was also calculated to gain insight into the extent the prices deviate from the mean. The following chapter investigates the limitations of all the models by changing the KBRL parameters to examine the effect of these changes on the price matrix of an option.

Chapter 6

Effect of changing KBRL parameters

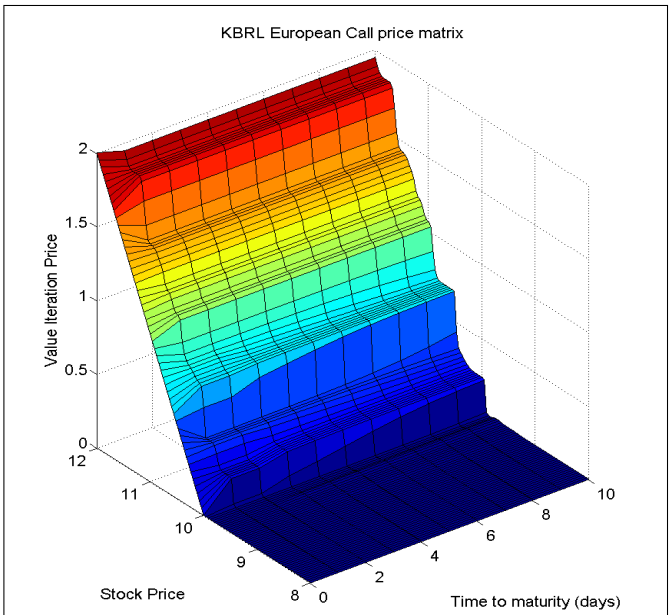
The KBRL approach to the option pricing problem has one significant parameter that may be adjusted to increase (or decrease) the effectiveness and accuracy of the models. KBRL makes use of a kernel function, in this case Gaussian, which has a standard deviation parameter b.

Section 6.1 describes and illustrates the impact of changing this parameter.

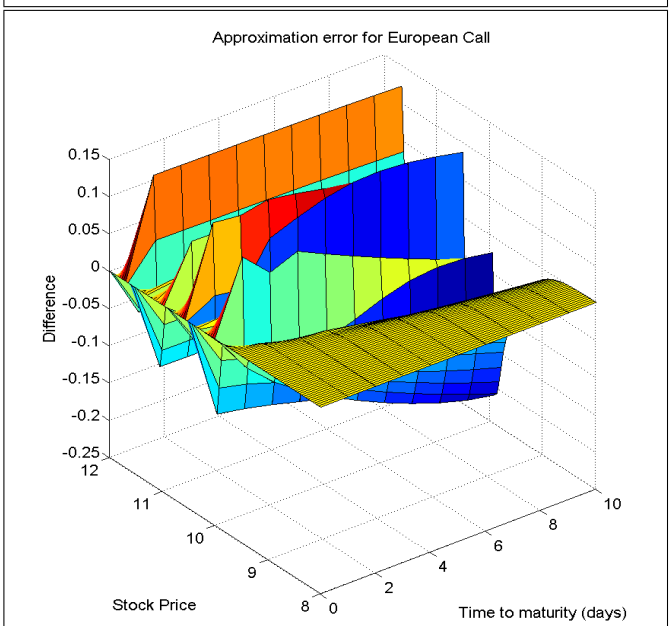
6.1 Standard deviation of kernels (b value)

This parameter is a constant number describing how quickly the weight of a potential successor state decreases as its distance from the unseen state increases. Intuitively, the higher the standard deviation, the wider the area of effect of each state. The standard deviation of the kernel is a highly volatile parameter in the KBRL model since a seemingly small change to this parameter can potentially cause significantly inaccurate prices.

The irregular nature of the price matrix in Figure 6.1 is caused by a lower than optimal standard deviation parameter. There are distinct regions in Figure 6.1 where contours are not smooth. Since the parameter is lower than optimal, too few neighbouring sampled states contribute to the value of the unseen states thereby causing these irregularities to occur.



(a) European call price matrix (KBRL-derived model)



(b) European call approximation error (KBRL-derived model)

Figure 6.1: European call price matrix with mapped state space and approximation error to analytic Black-Scholes using $b = 0.007$

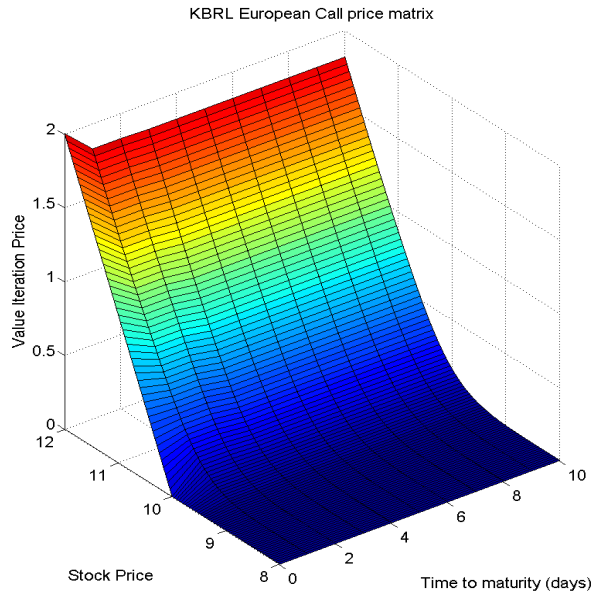
The price matrix is smoothened when this parameter is higher than the optimal value. The price of an unseen stock will be affected by states that are

very dissimilar to it, and will thus end up with an inaccurate value. Figure 6.2 illustrates how a higher b-value affects the overall price matrix for a European call.

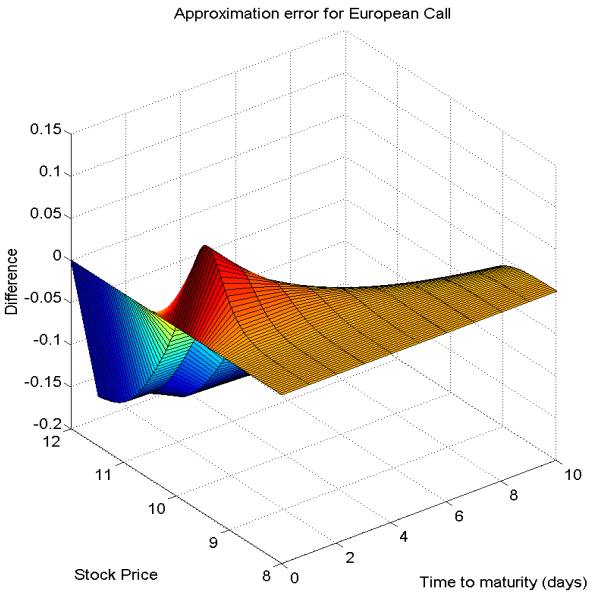
Setting the b value to infinity (equivalently, a large value) implies being able to reach *every* state from the current state with the same probability leaving the result as a *flat non-zero* surface (Figure 6.3). To the other extreme, setting the b value to zero implies that the current state that is being priced must have been seen before in order for a price to be obtained for the option. Therefore, the price matrix surface for any option would be undefined for the majority of the states.

6.2 Summary

This chapter has investigated the effects of changing the standard deviation of the kernels (b) value. The results indicate three distinct price matrix surfaces that arise for differing levels of the b-value. For b-values close to zero, the surface is irregular and bumpy in nature indicating that either more samples need to be generated or the b-value needs to be increased. The smooth surface shows that the in-the-money option regions underestimated the fair price and the out-the-money option regions overestimated the fair price. This indicates that the b-value must be decreased without changing the state space size. Finally, the flat surface resulted due to a large b-value being used which illustrates the upper bound of the price matrix. That is, the out-the-money option regions cannot have a higher price than the in-the-money regions and the in-the-money regions cannot have a lower price than the out-the-money regions.

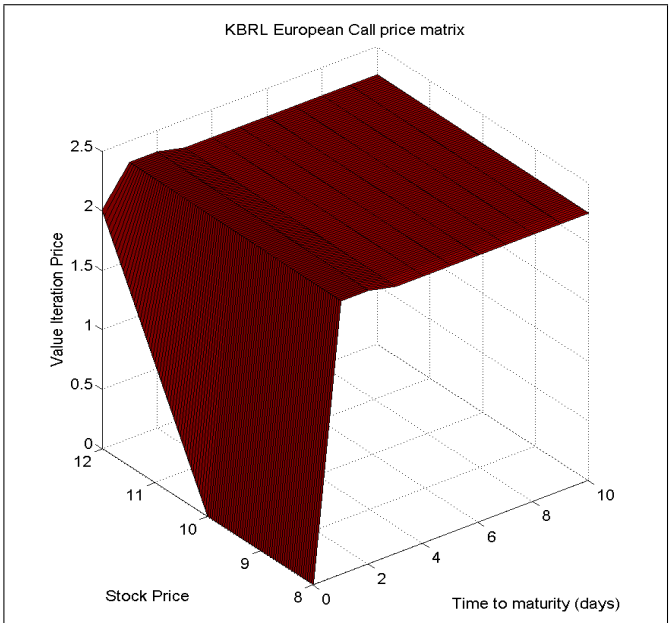


(a) European call price matrix (KBRL-derived model)

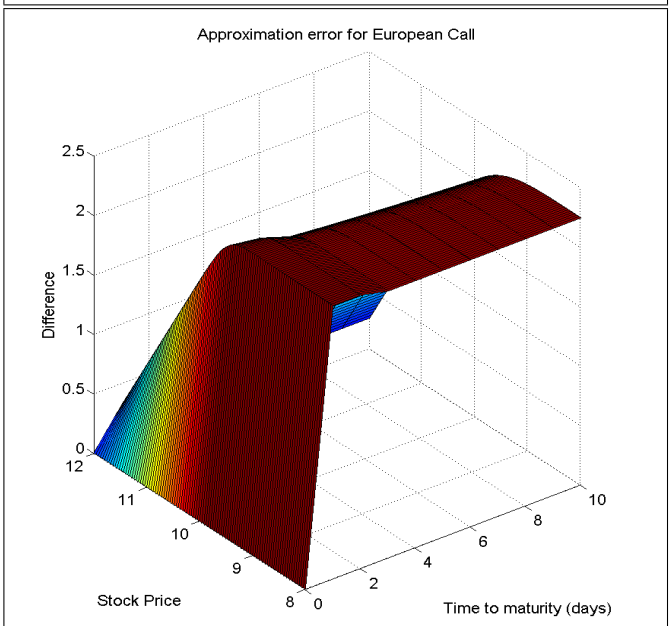


(b) European call approximation error (KBRL-derived model)

Figure 6.2: European call price matrix with mapped state space and approximation error to analytic Black-Scholes using $b = 0.07$



(a) European call price matrix (KBRL-derived model)



(b) European call approximation error (KBRL-derived model)

Figure 6.3: European call price matrix with mapped state space and approximation error to analytic Black-Scholes using large b value

Chapter 7

Conclusion

This dissertation has investigated the application of Kernel-Based Reinforcement Learning (KBRL) to solve the American option pricing problem. Existing numerical procedures are time consuming because simulations need to be generated for each option that requires a price. In addition, a trader requires accurate prices to mitigate the risk of loss in option trading.

7.1 Significance of research

The analytic Black-Scholes price for European options has been used since 1973 and has become the standard European option pricing formula. However, there is no standard option pricing procedure for American options since no closed-form solution exists for these types of options. Therefore, numerical approximations are frequently used to price these options. However, existing numerical procedures require running simulations to obtain a price for a single option. For an additional option, more simulations need to be generated to obtain the price for that option.

The significance of this research is that samples only need to be generated once to learn the American option pricing model and to calculate the value function. Following this, pricing any unseen American option requires the use of that value function eliminating the need to relearn the entire model and therefore drastically reducing the time it takes to obtain a price.

7.2 Research Findings

The results of the two experiments were compared to the closed-form solution in the case of European options and Monte Carlo simulation in the case of American options. The methods showed two significant benefits. The first benefit is the ability to price American options in 4% of the time of Monte Carlo simulation and the second is the ability to calibrate the models to market prices using market data.

7.3 Future Research

Only two option classes were investigated in this research, namely European and American options. These option classes are well-defined for the KBRL models shown in this research. Other option classes need to be analysed to further improve on these KBRL models. For example, Bermudan options have a prespecified number of dates on which the buyers of the option may exercise their right to obtain the payoff for the option. Unlike American options, Bermudan options are harder to price under the current KBRL models because the exercise dates of the option must be taken into account when calculating the price of unseen options.

In this research, a constant standard deviation parameter was used for the kernel function. This parameter should decrease as the number of market states increases. As was shown in chapter 6, the changes to this parameter caused large differences when compared to the existing pricing models. The standard deviation parameter determines how close a state needs to be in order to influence the price. If this is set incorrectly, states that are too far away could have an unwarranted impact. If it is set too low, we could have too few states, resulting in statistical anomalies. This poses the problem of determining what the correct value should be for any sampled state space size.

Further, the Matlab random number generator was used in all stock price and state space samples. Previous contributions to option pricing using Monte Carlo simulation have shown that the use of Sobol sequences drastically improves the convergence rate over the crude Monte Carlo approach which uses pseudo-random numbers[1, 3, 5, 10, 22]. An investigation into how Sobol sequences can improve the KBRL models to obtain consistent and accurate option prices would be worthwhile.

Another area of potential research is investigating the extension of the coverage of market data. Market data extracted from Yahoo Finance was limited to twelve large corporations from the United States of America which had a vast amount of European options traded in the US market. Including other corporations' market information to increase the sample size may lead to an increase in accuracy in the KBRL models.

In finance, pricing models must make financial sense (plausible) and must be easy to control (tractable). Therefore, surveys can be done on how traders interact with these models to assess whether these models are plausible and tractable.

Bibliography

[1] P. Acworth, M. Broadie, and P. Glasserman, *A comparison of some Monte Carlo and quasi Monte Carlo methods for option pricing*, 1st ed., Springer, New York, 1998.

[2] R. Bellman, *Dynamic Programming*, 1st ed., Princeton University Press, Princeton, NJ, USA, 1957.

[3] J.R. Birge, *Quasi-Monte Carlo approaches to option pricing*, Tech. report, Department of Industrial and Operations Engineering, University of Michigan, 1994.

[4] F. Black and M. Scholes, *The Pricing of Options and Corporate Liabilities*, Journal of Political Economy **81** (1973), no. 3, 637 – 659.

[5] P. Bratley and B.L. Fox, *Algorithm 659: Implementing Sobol's quasirandom sequence generator*, ACM Transactions on Mathematical Software **14** (1988), 88–100.

[6] Aristotle (Translated by Jowett. B), *Politics*, 1st ed., 350 B.C.

[7] J. C. Cox, J. E. Ingersoll, and S. A. Ross, *A theory of the term structure of interest rates*, Econometrica **53** (1985), no. 2, 385–407.

[8] J. C. Cox, S. A. Ross, and M. Rubenstein, *Option Pricing: A Simplified Approach*, Journal of Financial Economics **7** (1979), no. 3, 229–263.

[9] D. Duffie, *Dynamic Asset Pricing Theory*, 3rd ed., Princeton University Press, Princeton, NJ, 2001.

[10] S. Galanti and A. Jung, *Low-discrepancy sequences: Monte Carlo simulation of option prices*, Journal of Derivatives **5** (1997), 63–83.

[11] M. Gilli and E. Schumann, *Calibrating Option Pricing Models with Heuristics*, University of Geneva, Department of Economics (2010).

[12] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, New York, NY: Springer, 2003.

[13] T. Grassl, *A reinforcement learning approach for pricing derivatives*, 2010.

[14] P. S. Hagan and G. West, *Interpolation Methods for Curve Construction*, Applied Mathematical Finance **13** (2006), no. 2, 89 – 129.

[15] S. L. Heston, *A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options*, The Review of Financial Studies **6** (1993), no. 2, 327–343.

[16] R. A. Howard, *Dynamic Programming and Markov Decision Processes*, 1st ed., MIT Press, 1960.

[17] J. C. Hull, *Options, Futures and Other Derivatives*, 7th ed., Prentice Hall, 2008.

[18] J. C. Hull and A. White, *The Pricing of Options on Assets with Stochastic Volatilities*, Journal of Finance **42** (1987), no. 2, 281–300.

[19] ———, *An analysis of the bias in option pricing caused by a stochastic volatility*, Advances in Futures and Options Research **3** (1988), 29–61.

[20] H. Johnson and D. Shanno, *Option pricing when the variance is changing*, Journal of Financial and Quantitative Analysis **22** (1987), 143–151.

[21] N.K. Jong and P. Stone, *Kernel-Based Models for Reinforcement Learning*, ICML-06 Workshop on Kernel Methods and Reinforcement Learning (2006).

[22] C. Joy, P.P. Boyle, and K.S. Tan, *Quasi-Monte Carlo methods in numerical finance*, Management Science **42** (1996), 926–938.

[23] S. Junhua, *Pricing Multi-Dimension American Options by Simulation*, Master's thesis, MATHEMATICS, 2004.

[24] F.A. Longstaff and E.S. Schwartz, *Valuing American Options by Simulation: A Simple Least-Squares Approach*, The Review of Financial Studies **14** (2001), no. 1, 113 – 147.

[25] N. Moodley, *The Heston Model: A Practical Approach with Matlab Code*, Master's thesis, Computational and Applied Mathematics, 2005.

[26] D. Ormoneit and S. Sen, *Kernel-Based Reinforcement Learning*, Machine Learning **49** (2002), 161–179.

[27] E. Pashenkova, I. Rish, and R. Dechter, *Value iteration and policy iteration algorithms for Markov decision problem*, Department of Information and Computer Science, University of California at Irvine (1996).

[28] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*, 1st ed., John Wiley and Sons, New York, NY, 1994.

[29] J. Reisinger, P. Stone, and R. Miikkulainen, *Online Kernel Selection for Bayesian Reinforcement Learning*, Journal of Machine Learning, 2008.

[30] L. O. Scott, *Option pricing when the variance changes randomly: Theory, estimation, and an application*, Journal of Financial and Quantitative Analysis **22** (1987), no. 4, 419–438.

[31] E. M. Stein and J. C. Stein, *Stock Price Distributions with Stochastic Volatility: An Analytic Approach*, The Review of Financial Studies **4** (1991), no. 4, 727–752.

[32] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*, The MIT Press (1998).

[33] J. B. Wiggins, *Option values under stochastic volatilities*, Journal of Financial Economics **19** (1987), 351–372.

Appendix A

Pseudocode

The algorithm in Figure A.1 shows how to generate the stock price matrix using the Black-Scholes model. This procedure takes four input arguments, namely, the initial (or current) stock price U_0, the drift term μ (the riskless interest rate r is passed in for this value in a risk-neutral market), the volatility of returns on the stock σ and the time in years T. First, initialization of the number of stock price paths and number of timesteps in each path is done. Then, two matrices are created to store generated standard normal random numbers Z and the simulated stock prices U. The first column of U is initialized with the initial stock price U_0 to begin the simulation. Two loops are needed to simulate the stock prices: the outer loop controls the simulation for each stock price path while the inner loop controls the current time step. The Black-Scholes stock price formula is used to calculate each time step's simulated stock price using the previous simulated stock price. Plotting $U_{i,j}$ against t_j for each $i \in \{0, 1, \ldots, n-1\}$ and $j \in \{0, 1, \ldots, m\}$ will produce the result in Figure C.1.

The Monte-Carlo-Pricing-Model algorithm in Figure A.2 is generic enough that European and American option prices could be calculated by simply calculating a stock price matrix U using Figure A.1. The algorithm takes six input arguments, namely the stock price matrix U, the strike price of the option K, the riskless interest rate r, the time-to-maturity of the option T, the payoff function Payoff and the *optionType* (i.e. *American* or *European*). Calculation of a price begins by applying the $Payoff$ function to the last column in U. If the *optionType* is *European*, then the average of the values in the last column in U is calculated and discounted back to time 0 using $e^(- rT)$. This is the price for *optionType* = *European*. However, if the *optionType* is *American*, the $Payoff$ function is applied to every column in U. The algorithm progressively backtracks one row and one column at a time taking the maximum between the discounted payoff of one time-step ahead and the current payoff value (each column represents an equal time-step between the current time and the expiry of the option). Eventually, the algorithm gets

BLACK-SCOLES-STOCK-SIMULATION(U_0, μ, σ, T)

$U_0 \leftarrow$ initial stock price
$\mu \leftarrow$ drift of stock price process
$\sigma \leftarrow$ volatility of returns on the stock
$T \leftarrow$ maturity of the option

1 $n \leftarrow$ initialize the number of simulated stock price paths
2 $m \leftarrow$ initialize the number of time steps for each stock price path
3 $Z \leftarrow$ generate a matrix $(Z_{n\times m})$ of standard normal random variables
4 $U \leftarrow$ initialize a matrix $(U_{n\times(m+1)})$ to contain all zeros except for first column which contains U_0 - the initial stock price
5 $h \leftarrow \frac{T}{m}$

6 **for** $i \leftarrow 0$ **to** $n-1$ **do**
7 **for** $j \leftarrow 0$ **to** m **do**
8 $U_{i,j+1} \leftarrow U_{i,j} \exp\left[\left(\mu - \frac{1}{2}\sigma^2\right) h + \sigma Z_{i,j}\sqrt{h}\right]$

Figure A.1: Algorithm to generate Black-Scholes stock price paths

to first column in U and calculates the average of all values in that column which is the price for the American option.

MONTE-CARLO-PRICING-MODEL($U, K, r, T,$ payoff, *optionType*)

U - the stock price matrix with the first column being the initial stock price
K - the strike price of the option
r - the term structure of NACC forward interest rates (i.e. an array of forward interest rates)
T - the time-to-maturity of the option
payoff - the mathematical payoff function
optionType - European, Bermudan, American, etc.

```
1  m ← row dimension of S
2  n ← column dimension of S
3  t ← {T, T(n−2)/(n−1), ..., T/(n−1), 0}
4  for i ← 1 to m do
5      U_{i,T} = Payoff(U_{i,T}, K)
6  if optionType is European then
7      return exp(−rT) mean(U_T)
8  for j ← n − 1 to 1 do
9      for i ← 1 to m do
10         U_{i,t_j} ← max{exp(−r(t_{j+1} − t_j)) U_{i,t_{j+1}}, payoff(U_{i,t_j}, K)}
11 return mean(U_{t_0})
```

Figure A.2: Monte Carlo Pricing model

Appendix B

Market Data

B.1 Stock options data

Table B.1 shows an altered excerpt of Market Data stock options on Microsoft[1] shares taken from Yahoo Finance[2] for close of market on 8-Jan-2013. The table shows prices for European calls and puts at different expiration dates and strike prices. Transforming this data into Market States can be done by first calculating the implied volatility[3] ($\sigma_T^{(i)}$) and then using

$$m_t^{(i)} = \left(\frac{U_t}{K_T^{(i)}}, \sigma_T^{(i)}, T^{(i)} \right)$$

for the construction of each state.

B.2 Interest Rate Zero Curve (USD)

The zero curve (Table B.3) is stripped[4] from USD swap rate raw inputs (Table B.2). The deposits, futures and swaps data have the following characteristics:

Data Type	Reference Rate	Compounding	Daycount	Business Day Adjustment
Deposits	NONE	NONE	ACT360	Modified Following
Futures	LIBOR	-	ACT360	Following
Swaps	LIBOR	NONE	30/360	Modified Following

[1] MSFT stock price (U_t) was \$26.55 on 8-Jan-2013 at close of market

[2] `http://finance.yahoo.com` - used since it is a free service (this dissertation shows the simplicity of calibrating a model to market prices)

[3] Calculated by using the stock level, strike price, annualized interest rate, T (in years) and the price for the option using a modified Black-Scholes pricing formula

[4] see [14] for stripping a curve and interpolation methods that can be used

Expiration Date ($T^{(i)}$)	Strike Price ($K_T^{(i)}$)	Put Price ($P_T^{(i)}$)	Call Price ($C_T^{(i)}$)
18-Jan-2013	25	1.65	0.04
18-Jan-2013	25.5	1.08	0.02
18-Jan-2013	26	0.67	0.13
18-Jan-2013	26.5	0.21	0.17
18-Jan-2013	27	0.15	0.59
18-Jan-2013	27.5	0.01	0.98
15-Feb-2013	24	2.67	0.11
15-Feb-2013	25	1.78	0.22
15-Feb-2013	26	1.07	0.49
15-Feb-2013	27	0.52	0.96
15-Feb-2013	28	0.22	1.63

Table B.1: Excerpt Market Data of MSFT stock options for 8-Jan-2013

Deposits		Futures		Swaps	
Tenor	Rate	Product	Price	Tenor	Rate
1D	0.0164	EDH3	99.7	3Y	0.005025
1M	0.002077	EDM3	99.68	4Y	0.00678
2M	0.00252	EDU3	99.655	5Y	0.008985
3M	0.00305	EDZ3	99.63	6Y	0.011345
		EDH4	99.595	7Y	0.013585
		EDM4	99.545	8Y	0.0156
		EDU4	99.49	9Y	0.017375
		EDZ4	99.42	10Y	0.018979
				12Y	0.021665
				15Y	0.02439
				20Y	0.026655
				30Y	0.02858
				40Y	0.0289505

Table B.2: Market Data for USD swap rate raw data

Curve Dates	Days	Rate	Discount Factor
08-Jan-13	0	0.001641342	1
09-Jan-13	1	0.001641342	0.999995444
11-Feb-13	34	0.002053954	0.999806233
11-Mar-13	62	0.00249461	0.999571
10-Apr-13	92	0.003023042	0.999228908
19-Jun-13	162	0.002846259	0.99872182
18-Sep-13	253	0.002966648	0.997920358
18-Dec-13	344	0.003082847	0.997063022
19-Mar-14	435	0.00319556	0.996152276
18-Jun-14	526	0.003322347	0.995165452
17-Sep-14	617	0.003477802	0.994067442
17-Dec-14	708	0.003656138	0.992848402
18-Mar-15	799	0.003865626	0.991473556
11-Jan-16	1098	0.004964237	0.98501002
10-Jan-17	1463	0.00671298	0.973176666
10-Jan-18	1828	0.008937069	0.955826503
10-Jan-19	2193	0.011348337	0.933568472
10-Jan-20	2558	0.013670754	0.908027769
11-Jan-21	2925	0.015786488	0.880501791
10-Jan-22	3289	0.017683749	0.852016988
10-Jan-23	3654	0.019426024	0.822601468
10-Jan-25	4385	0.022407795	0.76343573
10-Jan-28	5480	0.025530515	0.681299658
10-Jan-33	7307	0.028155892	0.569162705
12-Jan-43	10961	0.030437789	0.401349804
10-Jan-53	14612	0.030579652	0.294465131

Table B.3: Market Data for stripped USD swap curve

Symbol	Date	Dividend
AAPL	07-Feb-13	2.65
AAPL	09-May-13	3.05
AAPL	08-Aug-13	3.05
AAPL	06-Nov-13	3.05
AAPL	06-Feb-14	3.05
AAPL	06-May-14	3.05
AAPL	06-Aug-14	3.05
AAPL	06-Nov-14	3.05
AAPL	06-Feb-15	3.05
AAPL	06-May-15	3.05
AMZN	07-Feb-13	0
F	28-Jan-13	0.1
F	01-May-13	0.1
F	31-Jul-13	0.1
F	30-Oct-13	0.1
F	29-Jan-14	0.125
F	29-Apr-14	0.125
F	29-Jul-14	0.125
F	29-Oct-14	0.125
F	29-Jan-15	0.125
F	29-Apr-15	0.125
FB	07-Feb-13	0
GOOG	07-Feb-13	0
HPQ	11-Mar-13	0.132
HPQ	10-Jun-13	0.145
HPQ	09-Sep-13	0.145
HPQ	09-Dec-13	0.145
HPQ	10-Mar-14	0.145
HPQ	10-Jun-14	0.145
HPQ	10-Sep-14	0.145
HPQ	10-Dec-14	0.145
HPQ	10-Mar-15	0.145
JPM	03-Apr-13	0.3
JPM	02-Jul-13	0.38
JPM	02-Oct-13	0.38
JPM	02-Jan-14	0.38
JPM	02-Apr-14	0.38
JPM	02-Jul-14	0.38
JPM	02-Oct-14	0.38
JPM	02-Jan-15	0.38
JPM	02-Apr-15	0.38

Table B.4: Dividend payouts per stock

MCD	27-Feb-13	0.77
MCD	30-May-13	0.77
MCD	29-Aug-13	0.77
MCD	27-Nov-13	0.81
MCD	27-Feb-14	0.81
MCD	27-May-14	0.81
MCD	27-Aug-14	0.81
MCD	27-Nov-14	0.81
MCD	27-Feb-15	0.81
MCD	27-May-15	0.81
MS	01-Feb-13	0.05
MS	26-Apr-13	0.05
MS	29-Jul-13	0.05
MS	29-Oct-13	0.05
MS	29-Jan-14	0.05
MS	29-Apr-14	0.05
MS	29-Jul-14	0.05
MS	29-Oct-14	0.05
MS	29-Jan-15	0.05
MS	29-Apr-15	0.05
MSFT	19-Feb-13	0.23
MSFT	14-May-13	0.23
MSFT	13-Aug-13	0.23
MSFT	19-Nov-13	0.28
MSFT	18-Feb-14	0.28
MSFT	18-May-14	0.28
MSFT	18-Aug-14	0.28
MSFT	18-Nov-14	0.28
MSFT	18-Feb-15	0.28
MSFT	18-May-15	0.28
SBUX	05-Feb-13	0.21
SBUX	07-May-13	0.21
SBUX	06-Aug-13	0.21
SBUX	12-Nov-13	0.26
SBUX	04-Feb-14	0.26
SBUX	04-May-14	0.26
SBUX	04-Aug-14	0.26
SBUX	04-Nov-14	0.26
SBUX	04-Feb-15	0.26
SBUX	04-May-15	0.26
YHOO	07-Feb-13	0

Table B.5: Dividend payouts per stock (continued)

Appendix C

Graphs

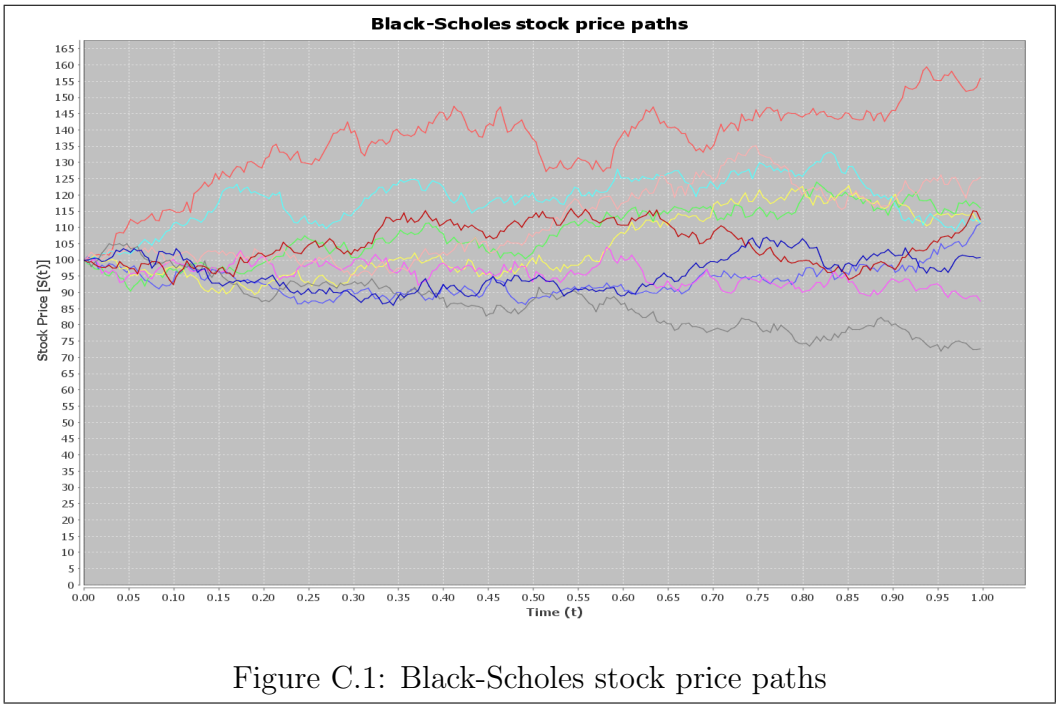


Figure C.1: Black-Scholes stock price paths

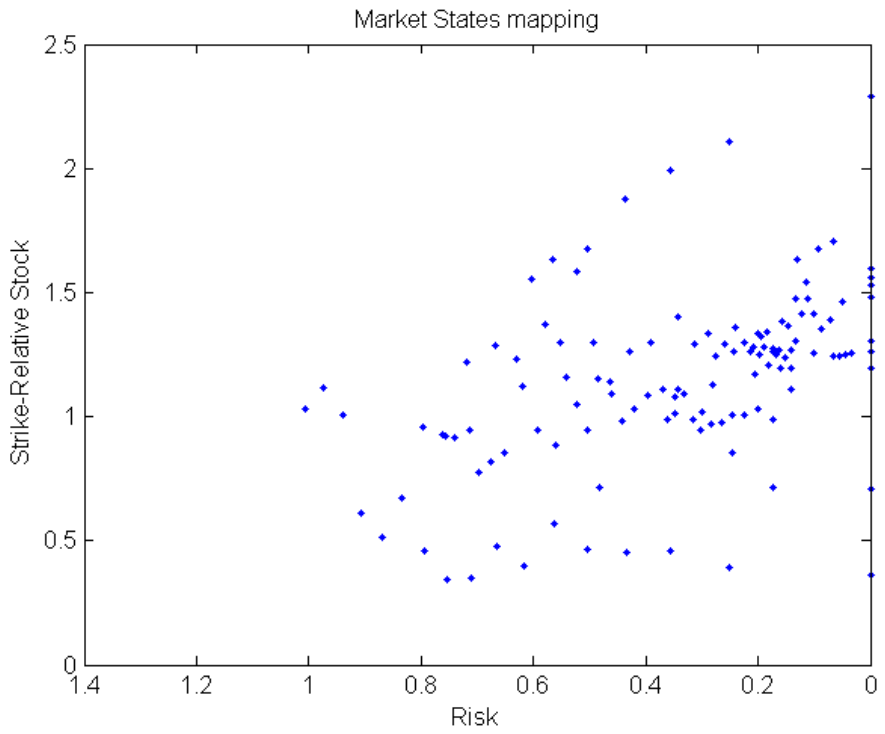


Figure C.2: Market States mapping on 10 sampled trajectories of strike-relative stock versus risk/time components

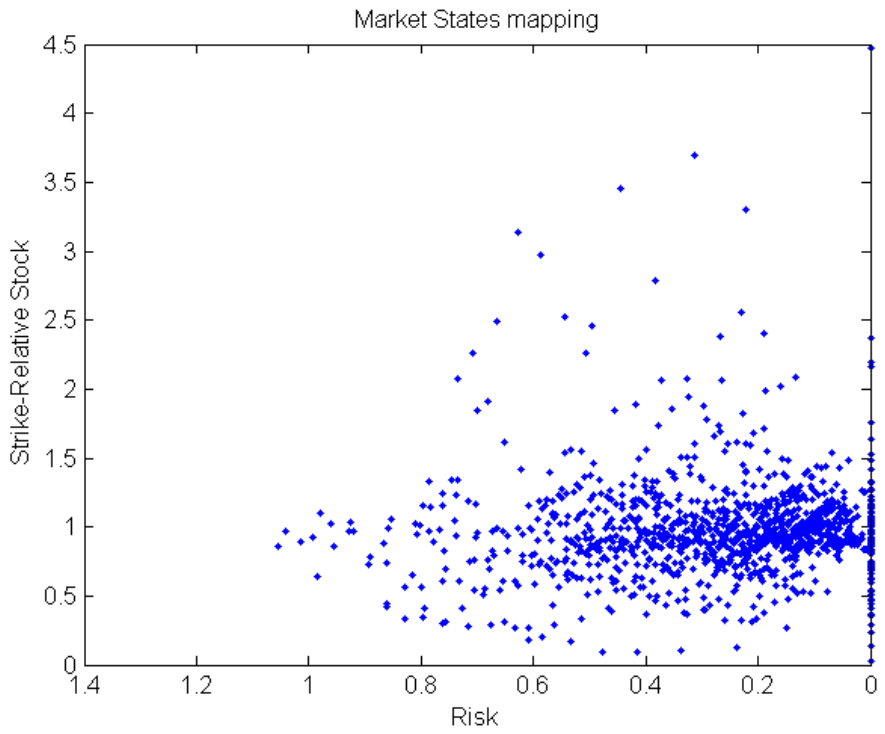


Figure C.3: Market States mapping on 100 sampled trajectories of strike-relative stock versus risk/time components

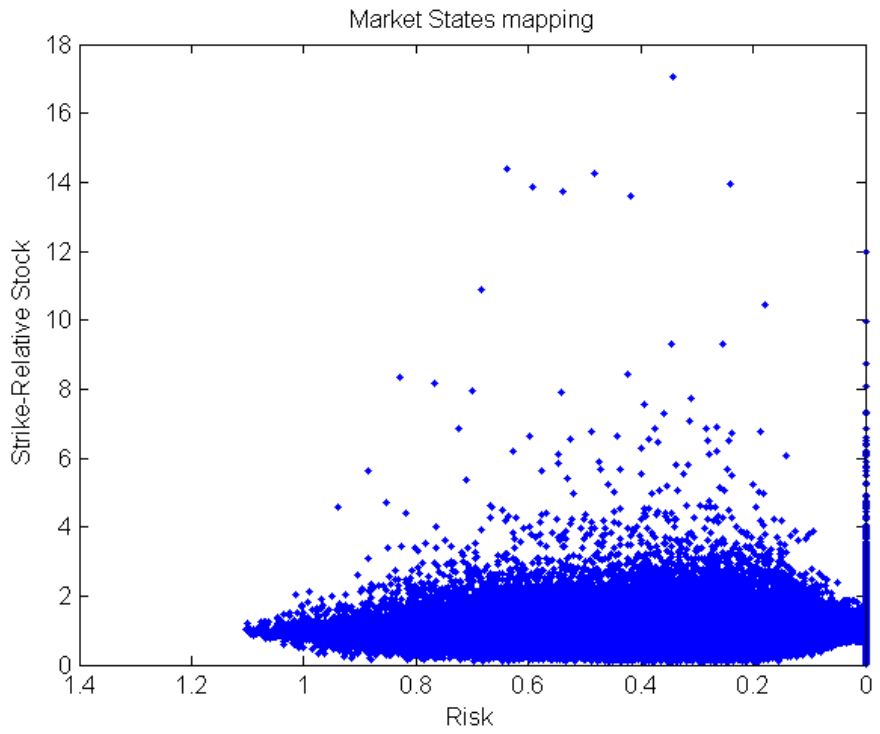


Figure C.4: Market States mapping on 10000 sampled trajectories of strike-relative stock versus risk/time components

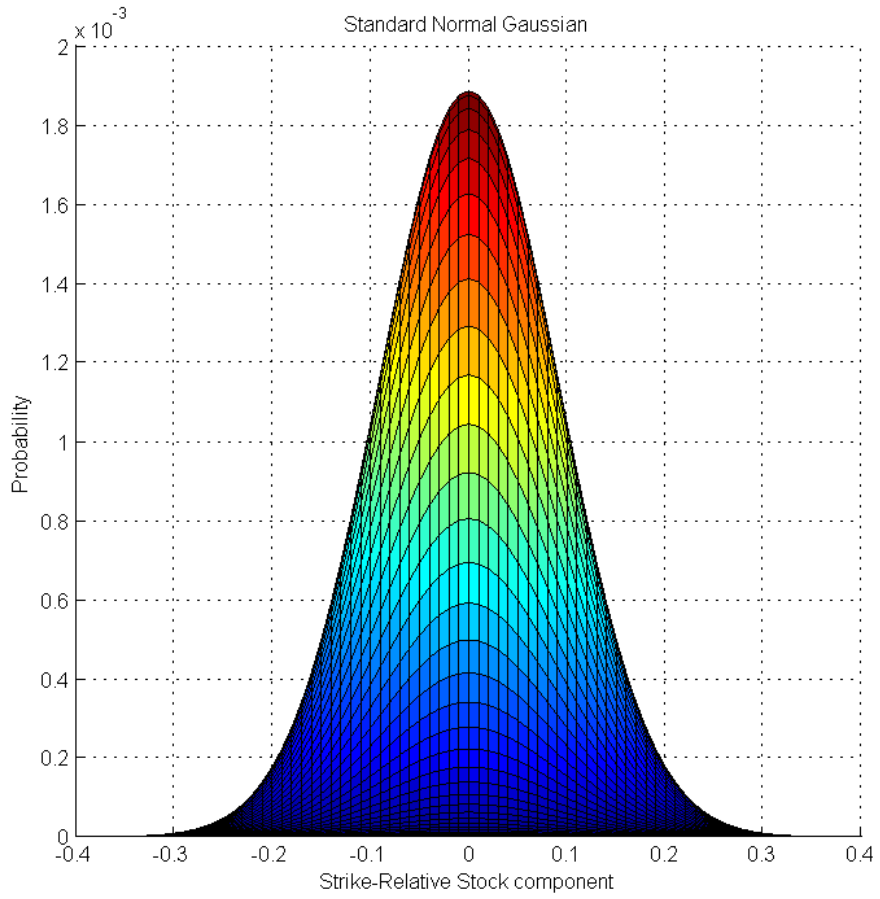
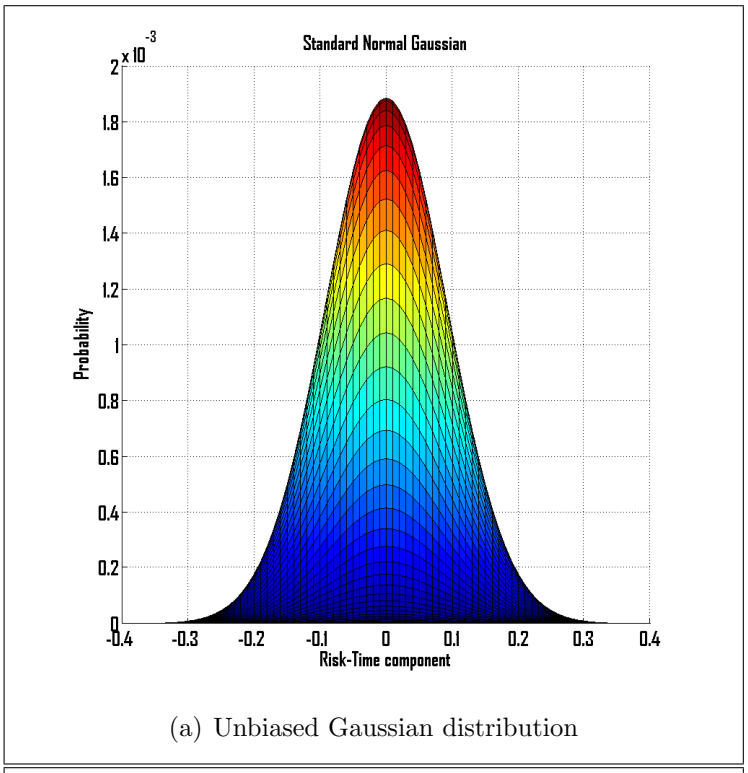
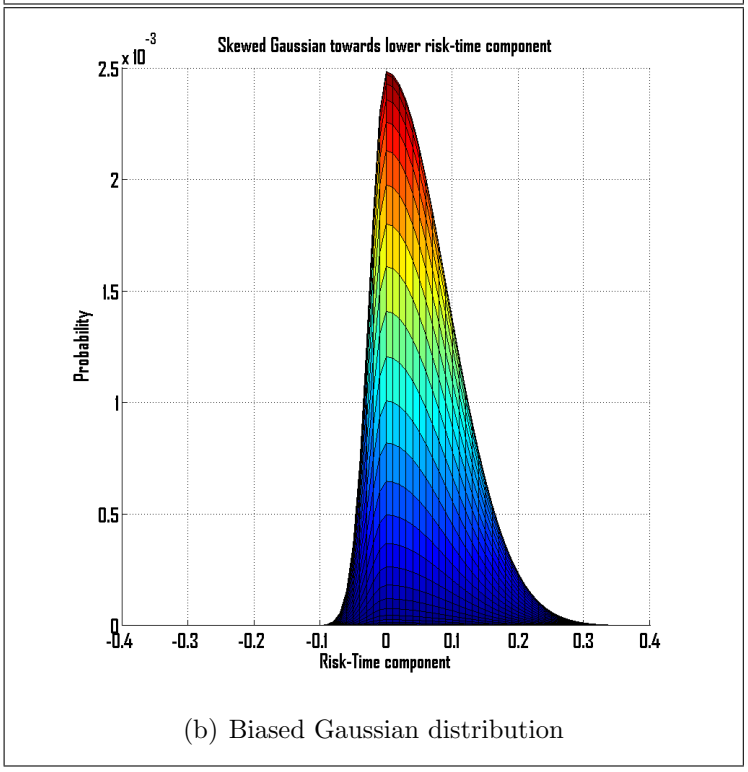


Figure C.5: Transition Probabilities reflecting unbiased Gaussian for Strike-Relative stock prices



(a) Unbiased Gaussian distribution



(b) Biased Gaussian distribution

Figure C.6: Transition Probabilities reflecting biased/skewed Gaussian for risk-time component

Appendix D

Tables

KERNEL	BASIC	EXTENDED
NORM	$k(\mathbf{x},\mathbf{x}') = 1 - \frac{\|\|\mathbf{x}-\mathbf{x}'\|\|^2}{\alpha}$	$k(\mathbf{x},\mathbf{x}') = 1 - \sum_i w_i(x_i - x_i')^2$
GAUSSIAN	$k(\mathbf{x},\mathbf{x}') = \exp\left[-\frac{\|\|\mathbf{x}-\mathbf{x}'\|\|^2}{\sigma^2}\right]$	$k(\mathbf{x},\mathbf{x}') = \exp\left[-\sum_i w_i(x_i - x_i')^2\right]$
POLYNOMIAL	$k(\mathbf{x},\mathbf{x}') = (\langle \mathbf{x},\mathbf{x}'\rangle + 1)^d$	$k(\mathbf{x},\mathbf{x}') = \left(\sum_i w_i x_i x_i' + 1\right)^d$
TANH NORM	$k(\mathbf{x},\mathbf{x}') = \tanh(v\|\|\mathbf{x}-\mathbf{x}'\|\| - c)$	$k(\mathbf{x},\mathbf{x}') = \tanh\left(\sum_i w_i(x_i - x_i')^2 - 1\right)$
TANH DOT	$k(\mathbf{x},\mathbf{x}') = \tanh(v\langle \mathbf{x},\mathbf{x}'\rangle - c)$	$k(\mathbf{x},\mathbf{x}') = \tanh\left(\sum_i w_i x_i x_i' - 1\right)$

Table D.1: Basic kernel functions and the corresponding extended parameterizations