Efficient Procedure for Valuing American Lookback Put Options

by

Xuyan Wang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Actuarial Science

Waterloo, Ontario, Canada, 2007

Abstract

Lookback option is a well-known path-dependent option where its payoff depends on the historical extremum prices. The thesis focuses on the binomial pricing of the American floating strike lookback put options with payoff at time t (if exercise) characterized by

$$\max_{k=0,\ldots,t} S_k - S_t,$$

where S_t denotes the price of the underlying stock at time t. Build upon the idea of Reiner Babbs Cheuk and Vorst (RBCV, 1992) who proposed a transformed binomial lattice model for efficient pricing of this class of option, this thesis extends and enhances their binomial recursive algorithm by exploiting the additional combinatorial properties of the lattice structure. The proposed algorithm is not only computational efficient but it also significantly reduces the memory constraint. As a result, the proposed algorithm is more than 1000 times faster than the original RBCV algorithm and it can compute a binomial lattice with one million time steps in less than two seconds. This algorithm enables us to extrapolate the limiting (American) option value up to 4 or 5 decimal accuracy in real time.

Acknowledgements

It would be ridiculous to claim the success, if any, all to myself. Any obstacle could render my effort in vain. Thanks are due to all the people mentioned below or the people I failed to mention at all: it is their cohort effort that makes this progress happen.

First I want to thank Professor Jock MacKay for all the long hours he tried to help me enroll or made it easy for me to enroll in courses.

I also want to thank Professor Mary Hardy for promising me admittance upon my performance on undergraduate study and then permitting me directly into graduate study.

Big thanks are due to Professor Jiahua Chen, Mary Lou Dufton and my supervisor Professor Ken Seng Tan for their unrelenting efforts in helping me make through to enroll in graduate study.

Sincere thanks are due to Professor Colleen Cutler and Ken Seng Tan for their teaching, encouragement and help. I am very appreciative of the awarded financial support of NSERC Postgraduate Scholarship, Empire Life Insurance Company Graduate Scholarship in Actuarial Science through IQFI, President's Graduate Scholarship from the university and department funding.

I wish to thank Professor Weidong Tian for his kindly consideration which has helped me survive the exhausting course study and concurrent SOA exams.

Professor Peter Forsyth's course, "Numeric Computation For Financial Modelling", gave me many helpful ideas, such as the concept of memory saving, which is utilized in my algorithm: even though he may not favor lattice method in general to PDE method.

The KMV model project is a great programming task, and the result is of general interest, both are valuable to me: thank Professor Harry H. Panjer for the course "Enterprise Risk Management".

I wish to thank Professor Phelim P. Boyle because the thesis topic started from observations made while doing an assignment in his finance course.

I cannot overemphasize my gratitude to Professor Ken Seng Tan. While at undergraduate study I was led to pursue graduate study by his persuasion, at graduate study, his valuable insight regarding the topic finally gave me the direction to success. Also, his emphasis on algorithm efficiency and the tasks he assigned to me fully prepared me for this kind of problem.

Finally I want to thank my whole family here in Waterloo, Wenyi, Claire and Albert and my big family back in China for always being beside me whether at the success time or at the struggle time.

Contents

1 Introduction **1**

1.1 Option Valuation and Binomial Model 1

1.2 Literature review 5

1.3 Main idea of our approach 10

2 Algorithm and Numerical Results **13**

2.1 Binomial Valuation of Floating Strike Lookback Option 13

2.2 Proposed Algorithm for Pricing Floating Strike Lookback Option 26

2.2.1 RBCV Algorithm 26

2.2.2 Proposed enhanced algorithm 27

2.3 Numerical results 29

3 Proofs and Discussions **36**

3.1 Proof of Corollary 2.1.4 36

3.2 Proof of Theorem 2.2.2 37

3.3 Proof of Lemma 2.2.4 40

3.4 Proof of Proposition 2.2.5 41

3.5 Proof of Theorem 2.2.6 42

3.6 Proof of Corollary 2.2.7 42

4 Other Types of Lookback Options **46**

4.1 Floating strike lookback call options 46

4.2 Fixed strike lookback call options 48

4.3 Fixed strike lookback put options 51

5 Conclusion **52**

List of Tables

2.1 State matrix for a binomial lattice with time periods N=6 18
2.2 State triangles: $N = 6, n = 7$ and $n = 6$ 20
2.3 State triangles: $N = 6, n = 6$ and $n = 5$ 21
2.4 State triangles: $N = 6, n = 4$ and $n = 3$ 22
2.5 State triangles: $N = 6, n = 2$ and $n = 1$ 24
2.6 Comparison of computing time (in seconds): Matlab code implementation . 29
2.7 Quotient of difference, note that $\frac{1}{\sqrt{2}} = 0.70710678$ 32
2.8 Predicted continuously exercisable American lookback put option value . . 33

4.1 Computing time (C=20.5521826180488,P=18.7232860368255) 49

List of Figures

2.1 Graphical Depiction of a Binomial lattice with $N = 6$ 15
2.2 State triangle for a lattice with time periods N=6 23
2.3 State triangle for a lattice with time periods N=20 25
2.4 Matlab code computing time of RBCV algorithm 30
2.5 Matlab code computed value of RBCV algorithm 31
2.6 C code computing time of our revised algorithm 31
2.7 C code computed value of our revised algorithm 32
2.8 Predicted continuously exercisable American lookback put option value . . 34
2.9 Power function weighted least square fit of American lookback put option . 35

3.1 Continue region of L_2^{N-1}: $x = a, y = u, f(x, y) < 0$ 44
3.2 Continue region of L_2^{N-1}: $x = a, y = u, y > f(x)$ 45
3.3 Continue region of L_2^{N-1} for sufficient large N 45

Chapter 1

Introduction

1.1 Option Valuation and Binomial Model

First, we will give a brief description of the terminology that will be used in this thesis. Most of the terminology is taken from the free encyclopedia Wikipedia. For example, the definition of option as given in the Wikipedia `http://en.wikipedia.org/wiki/Option_%28finance%29` is:

> In finance options are types of derivative contracts, including call options and put options, where the future payoffs to the buyer and seller of the contract are determined by the price of another security, such as a common stock. More specifically, a call option is an agreement in which the buyer (holder) has the right (but not the obligation) to exercise by buying an asset at a set price (strike price) on (for a European style option) or not later than (for an American style option) a future date (the exercise date or expiration); and the seller (writer) has the obligation to honor the terms of the contract. A put option is an agreement in which the buyer has the right (but not the obligation) to exercise by selling an asset at the strike price on or before a future date; and the seller has the obligation to honor the terms of the contract. Since the option gives the buyer a right and the writer an obligation, the buyer pays the option premium to the writer.

Option valuation is the determination of an option contract's premium. Two factors that determine the complexity of the option valuation most are the option style and payoff forms.

> The option style determines when the buyer may exercise the option. Generally the contract will either be American style — which allows exercise up to the expiration date — or European style — where exercise is only allowed on the expiration date — or Bermudan style - where exercise is allowed on several, specific dates up to the expiration date.

When the payoff function not only depends on the on-the-spot underlying asset price, but also on some function of the history of the underlying (asset price), it is called a path-dependent option. When an extreme function is used, it is called a lookback option, and when an average function is used, it is called an Asian option.

Another factor affecting the valuation most is the model or dynamics assumption about the underlying asset. The classical assumption is that the asset follows a geometric Brownian motion (GBM). Nowadays, it is extended to jump-diffusion processes or even more general stochastic differential equations (SDE) [42]. The no-arbitrage (or no free lunch) principle — portfolios with the same payoffs must cost the same — is applied to obtain a partial differential equation (PDE) or variational inequality governing the option price. This is the major approach to option valuation as vast tools exist in PDE: various transformation [43] and approximation methods as well as numerical solutions including error analysis and stabilization techniques [45]. This kind of method is so advanced that now even the C code of option valuation can be automatically generated [6, 29]. It is shown by analysis and numerical example that the PDE method is better than other methods in efficiency (at least in low dimensions, but also see the counter example: [32]) [25, 22].

The Feynman-Kac (FK) formula gives another expression of the option price as the path-integral (expectation) of the underlying, which is the foundation of Monte-Carlo (MC) methods — the competing methods to PDE. The early exercisable and path-dependent options are two trying cases for these two methods. To solve the first problem, PDE methods exploit the exercise region and exercise boundary concept and use approximation of the boundary shape from various function types to give a lower or upper bound for the option price. MC methods had a counterpart of estimated continuation value at each time step [35, 38]. It seems a tie in this case in their competition. As for the second problem, PDE methods used enhanced state space to handle the extra dependence on the path and then tried to reduce the dimension of the enhanced state space by stratification or aggregation by cells, which were cut out by using some characteristic function on the enhanced state space [11]. What the MC method faced here was just a high-dimensional integration problem, and a natural application place of the new development in quasi-Monte Carlo or lattice rule method [41, 39, 10]. In this respect, the MC methods seem to have great advantage over PDE methods.

Both the PDE and MC methods seem to complement each other and have the common property of general applicability — can be applied to almost every type of options — and have theoretical analysis tools to give error bands [8, 9]. Thus, there is little room for other methods. But there is a third choice of lattice model, the simplest of which is the binomial pricing model. Here is a description in Wikipedia `http://en.wikipedia.org/wiki/Binomial_options_pricing_model` :

> The binomial pricing model uses a "discrete-time framework" to trace the evolution of the option's key underlying variable via a binomial lattice (tree), for a given number of time steps between valuation date and option expiration.
>
> Each node in the lattice, represents a possible price of the underlying, at a particular point in time. This price evolution forms the basis for the option valuation.
>
> The valuation process is iterative, starting at each final node, and then working backwards through the tree to the first node (valuation date), where the calculated result is the value of the option.
>
> Option valuation using this method is, as described, a three step process:
> 1. price tree generation
> 2. calculation of option value at each final node
> 3. progressive calculation of option value at each earlier node; the value at the first node is the value of the option.

The tree is a simple plane geometry object and the recursion is simple algebra and yet already capable of valuing early exercise options. This simple nature gave it a niche of a bottom-up approach. Instead of starting from theory analysis and working to algorithm design to implementation, in a binomial model, we can do it the other way. First, we do the implementation and then do the analysis through observation of the pattern in the calculated number, perhaps by also doing some experiment — manipulating the place that can be adjusted, for example, to alter the data representation method or use different but equivalent recursion formulas. For the very primitive intention of reducing the unneeded calculations or reducing the memory usage, we can reinvent the "wheel" from scratch, and even go beyond the known "wheel". To deal with the path-dependent option valuation, we can use a brute-force method of calculating all the possible paths, which amounts to adding a third dimension to the binomial tree, and see what we can do to reduce the calculations. By stepping through the calculations, we find that the calculated prices at different nodes have some nice properties: many values remain unchanged across different time steps, and once they change in a time step, they will increase monotonically thereafter. In addition, through observing the recursion formula, we find that we can divide it by some quantity and the formula is still valid. Thus, the recursive calculations in different part of the tree are in a sense repetitive calculations. All these observations gave us some general principles, or just general guidelines. We summarize them as: "uniformity" — which means do not repeat the calculations if they are just proportional to each other — a counterpart to the change of numeraire concept in SDE and transformation method in PDE; "exercise barrier" — which means if a value remains unchanged then we do not need to do the calculation again and again to get the same number in each time step — a counterpart to the exercise region or exercises boundary concept in PDE, even if we do not know a priori it exists and is convex (our numerical experiment shows in Geometric Asian option case the boundary is not convex and is instead a jigsaw); and "monotonicity" and "exercises propagation" — some inequality relationship of the prices that is obvious in the simple recursion algebra formula. Abstruse theory and methods do work very well, but the converse is not true. In some cases, very simple and even very primitive methods also work well. Our simple observations will be shown to work thousands of times faster than without utilizing these additional observations.

This bottom-up approach sheds new light on the concept that other methods used, which enables us to understand better what is really working if at a first reading of those "higher methods" we cannot understand what is really going on. And in some special cases, it can even work better. The drawback to the lattice method is that we need a case-by-case analysis of each kind of option. And even if we find these nice properties in such a case, we usually cannot give a thorough analysis for the error bounds. We can only use numerical examples to show its efficiency. But also due to its fully tailored nature, if it can be applied somewhere, it may be more efficient than the general applicable methods. Such is the case of lookback option valuation.

1.2 Literature review

According to Babbs(2000) [4], the lookback or "hindsight" option, which gives the owner the right to buy (sell) a non-divided paying security at the lowest (highest) price achieved by that security over the lifetime of the option, was first proposed by Goldman, Sosin and Gatto [30] in the year 1979 as pure theoretical research. This option is now called the floating strike European lookback call (put) option, as the option can be exercised only at the maturity time T. More explicitly, the payoff of the call is given by $S_T - m_0^T$ where

$$m_0^T = \min\{S_s \mid s \in [0,T]\},$$

and S_t is the security price at time t. On the other hand, the payoff of a European put option is $M_0^T - S_T$ where

$$M_0^T \quad = \quad \max\{S_t \mid t \in [0,T]\}.$$

The counterpart fixed strike European lookback option is defined similarly. For example, the call option's payoff becomes $(M_0^T - K)_+$ while the put option's payoff is $(K - m_0^T)_+$, where K is the fixed strike price.

The closed-form formulae for valuation of these European lookback options were given by Conze and Viswanathan(1991) [15]. There, the floating strike lookback options were named "standard lookback", and the fixed strike lookback options were named "options on extrema". For the floating strike lookback call (put) option issued at time 0 and with maturity at time T, the value is respectively C and P:

$$C = S_0[1 - N(-d_1)(1 + \frac{\sigma^2}{2r}) + e^{-rT}N(d_2)(-1 + \frac{\sigma^2}{2r})], \tag{1.2.1}$$

$$P = S_0[-1 + N(d_1)(1 + \frac{\sigma^2}{2r}) + e^{-rT}N(-d_2)(1 - \frac{\sigma^2}{2r})], \tag{1.2.2}$$

where

$$d_1 = (\frac{r}{\sigma} + \frac{\sigma}{2})\sqrt{T}, \tag{1.2.3}$$

$$d_2 = (\frac{r}{\sigma} - \frac{\sigma}{2})\sqrt{T}, \tag{1.2.4}$$

$$N(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{u^2}{2}}du, \tag{1.2.5}$$

and r is the risk-free interest rate, σ is the volatility of the security. Similar formulae for the fixed strike (and other kind of) lookback options were also given in their paper.

The authors also considered for the first time the American version of these lookback options: the exercise time of the options can be any moment from 0 to T. They proved that for floating strike lookback options:

$$C^a = C,\ P \leq P^a \leq Pe^{rT} + S_0(e^{rT} - 1),$$

where $C(P)$ denotes the time-zero value of the European call (put), $C^a(P^a)$ denotes the time-zero value of the American call (put), for the fixed strike lookback option:

$$C \leq C^a \leq Ce^{rT},\ P \leq P^a \leq Pe^{rT}.$$

The paper used Martingale theory and the concept of Snell envelope to arrive at these conclusions, while in our paper we will also give a proof of $C = C^a$ for floating strike lookback call option, but only elementary algebra calculations are used.

Please also note that the famous Global Derivatives Lookback Options website
`http://www.global-derivatives.com/options/lookback-options.php`
gives a misleading (at least for me once upon a time) introduction regarding this formula in saying that "American *fixed* lookback calls are always equal in value to their European counterparts" *etc.*

Since no analytic formulae exist for floating strike American lookback put options and all kinds of fixed strike American lookback options, many numerical methods have been proposed, including for example, the Laplace/Gaussian transform method by Petrella and Kou(2004) [37], Yamamoto(2005) [43]; the Monte Carlo/Quasi-Monte Carlo method by Boyle, Lai and Tan(2001) [10], Papatheodorou(2005) [36], and Athanassios, Avramidis and L'Ecuyer (2005) [1]; PDE methods by the paper of Forsyth, Vetzal and Zvan(1999) [28], Foufas and Larson(2004) [24], Yan(2005) [44], especially, Barraquand and Pudet(1994) [11], who proposed a Forward Shooting Grid (FSG) method that can be used to compute the exact price of American lookback options, see also Forsyth, Vetzal and Zvan(1998) [27] and (2002) [26] regarding convergence problems of the FSG method; and finally the lattice method.

John Hull in his Technical Note 13 gives a succinct and elementary introduction to the "transformed" lattice method: a new quotient variable of the maximum stock price divided by terminal stock price is defined, a tree and recursion equation is given for this quotient. No mention is made of change of numeraire. Hull attributes this idea to unpublished work of Eric Reiner, S. Babbs, T. H. F. Cheuk and T. C. F. Vorst (RBCV). Hull's note takes only about two pages:

> We define $G(t)$ as the maximum stock price achieved up to time t and set
>
> $$Y(t) = \frac{G(t)}{S(t)}.$$
>
> We next use the Cox, Ross, and Rubinstein binomial lattice for the stock price to produce a tree for Y. Initially, $Y = 1$ because $G = S$ at time zero. If there is an up movement in S during the first time step, both G and S increase by a proportional amount u and $Y = 1$. If there is a down movement in S during the first time step, G stays the same, so that $Y = 1/d = u$.
>
> The rules defining the geometry of the tree are
> 1. When $Y = 1$ at time t, it is either u or 1 at time $t + \Delta t$.
> 2. When $Y = u^m$ at time t for $m \geq 1$, it is either u^{m+1} or u^{m-1} at time $t + \Delta t$.
> An up movement in Y corresponds to a down movement in the stock price, and vice versa. The probability of an up movement in Y is, therefore, always $1 - p$ and the probability of a down movement in Y is always p.

We use the tree to value the American lookback option in units of the stock price rather than in dollars. In dollars, the payoff from the option is

$$SY - S.$$

In stock price units, the payoff from the option, therefore, is

$$Y - 1.$$

We roll back through the tree in the usual way, valuing a derivative that provides this payoff except that we adjust for the differences in the stock price (*i.e.*, the unit of measurement) at the nodes. If $f_{i,j}$ is the value of the lookback at the jth node at time iΔt and $Y_{i,j}$ is the value of Y at this node, the rollback procedure gives

$$f_{i,j} = \max(Y_{i,j} - 1, e^{-r\Delta t}[(1-p)f_{i+1,j+1}d + pf_{i+1,j-1}u]),$$

when $j \geq 1$. Note that $f_{i+1,j+1}$ is multiplied by d and $f_{i+1,j-1}$ is multiplied by u in this equation. This takes into account that the stock price at the node (i,j) is the unit of measurement. The stock price at the node $(i+1, j+1)$, which is the unit of measurement for $f_{i+1,j+1}$, is d times the stock price at the node (i,j) and the stock price at the node $(i+1, j-1)$, which is the unit of measurement for $f_{i+1,j-1}$, is u times the stock price at the node (i,j). Similarly, when $j = 0$ the rollback procedure gives

$$f_{i,j} = \max(Y_{i,j} - 1, e^{-r\Delta t}[(1-p)f_{i+1,j+1}d + pf_{i+1,j}u]).$$

His reasoning may not sound very rigorous in stating that the up and down probability of the quotient of the maximum stock price over the current stock price is the same as the original down and up probability, also the addition of a factor of d and u for the rollback procedure seems very heuristic. But he does emphasize the essence of the transformation method is the recursion on the quotient that gives the major reduction to the calculation.

We followed Hull's approach, but instead of working on the tree for the quotient, we reverse to the original binomial tree of the underlying. By analysis of the form of the values of the maximum of stock price of all the possible paths and the movement of these values when time goes on, we find other properties of the quotient which can be used as “stop” criteria for the recursion and thus gives improvement upon the original RBCV recursion.

If we adopted Hull’s ingenious tree for the quotient, and did not use our cumbersome and clumsy inflated binomial tree with all the possible path maximum values attached to each node, it may not be possible for us to find other properties of this recursion. Even though these concepts are well known in other methods, only have I spent much efforts in the complicated state matrix manipulation do I know how the counterparts of those concepts work in the binomial case. In the next chapter we will go through the same journey as how these concepts are originally discovered. At first sight they may seem much wired because they stem from the Matlab data structures to represent the complete path maximum of each node and then do recursion on these data structures. If we do not interested on how these concepts are discovered we may ignore this part as these principles once established their algebraic description and deduction are very simple: just simple elementary algebra — these are the contents of the second section of the next chapter and of chapter 3.

Cheuck and Vorst(1997) [16] as well as Babbs(2000) [4] published their works later. In the latter paper, Babbs gives proof of the convergence of the modified lattice method as the time step goes to infinite. Using our revised algorithm and possibility for calculation of wide range of time steps (up to 1,024,000,000) we can see clearly the convergence pattern and the power function relationship form of option values with respect to time steps (an example of reinvent the “wheel” of the Richardson extrapolation from our naked eyes). This is also used in our paper to make extrapolation for the continuous exercisable option value.

Boyle and Tian(1999) [12] and Boyle, Tian and Imai(1999) [13] give a trinomial lattice approach built also upon the change of numeraire idea, see also Broadie, Glasserman and Kou(1999) [7]. In Boyle et al’s paper, the minor difference between Cheuck and Vorst’s approach and Babbs’ approach is mentioned. Some new development closely related to the RBCV method can be found in the paper of Lai and Lim(2004) [33] and [34], see also Dai and Kwork(2005) [19] and (2006) [20].

Dai(2000) [18] used PDE method to establish a modified RBCV recursion formula by using different grid and showed by numerical experiments that his modified method has better convergence rate than the original RBCV recursion formula. By inspecting our algebraic proof of exercise barrier property, we can see our idea of revised algorithm with stop criteria imposed by exercise barrier property can be also applied to Dai's modified binomial tree method (then a modified modified RBCV). Numerical calculation confirms our claim and also confirms the vast improvement of accuracy of Dai's method over the original RBCV.

The RBCV type of algorithm (in our terminology "uniformity" principle and in PDE term scalability or homogeneous) is only one method of reducing the calculation. If we do not limit our scope to this one method, we can find many other papers of different methods of calculation or storage reduction. First is Boyle's ingenious trinomial tree method [5], because it gives us extra freedom to manipulate upon. This freedom is utilized in [21] (see also [23]) to give the first time a viable exact valuation method of Asian option — the parameter of the trinomial lattice is adjusted to make each underlying price integer and thus effectively reduce the exponential increase of possible path states. Another good example of "bottom-up" method is [14, 17] and [40], apparently out of their efforts in overcoming the overwhelm need of memory for storing the underlying price path.

Finally we have to mention the work of Andricopoulos et al. [2, 3] because it does not fall under any category of the three kinds of methods, neither can it be ascribed as amenable to our guidelines, but it beats all other methods according to their numerical results — obviously our principles are not ubiquitous axioms!

1.3 Main idea of our approach

As mentioned in John Hull's Technical Note 13,
`http://www.rotman.utoronto.ca/%7Ehull/Technical%20Notes/TechnicalNote13.pdf`
the transformed lattice method by Reiner Babbs Cheuk and Vorst is very efficient: compare to the intuitive method introduced in [31, p. 570], this method has a computation complexity quadratically on time steps instead of cubically depends on time steps. An implementation of this algorithm in Matlab can be found in the following address: `http://www.math.nctu.edu.tw/finmath/C_FILES/Intro_to_FinMath/Projects/Team1/finmath(1).pdf`
Using Numerical Methods to Value Lookback Options -Based on Taiwan's Stock Markets

By utilizing the following five observations, we can greatly reduce the storage and computation requirement of this algorithm:

1. Except for the minimum possible path maximum state (to be introduced latter) values, the optimum strategy is to exercise if at the next time step the optimum strategy is to exercise regardless of up or down movements of the underlying. In other words, if the value of the price of the American lookback put option at time step n+1 as a maximum of either continue or exercise is attend at exercise, and that at time step n, the path state is not the minimum possible state value, in the exceptional minimum case, it is always optimum to not exercise, *i.e.*, to continue, then the option value is still attend at exercise.

2. With the same ratio of path maximum state value over underlying value, the ratio of option value over underlying value is the same at each fixed time step for different state values, *i.e.*, the ratio of option value over underlying value is unique over every $45°$ "line" of the state matrix (to be introduced latter): this idea is inspired from the tech note #13 of John Hull by his emphasis on the quotient, but we cannot find it directly from that document because we are using a different framework from his — in his setting the quotient is defined a priori and in our setting we need to proof the quotient is independent of the state and thus can be defined — his approach is simple and our approach is complex.

3. The initial ratio of option value to underlying asset value is $0, u-1$,u^2-1, ..., u^N-1, if exercised at time $N+1$. In the backward recursion, at time n, only the (at most) first $\min(n, N+1-n)$ ratios will change when we calculate the ratio for time n from the ratios at time $n+1$. So we only need to store the initial value up to $u^{[\frac{N+1}{2}]}-1$.

4. These ratios are increasing with respect to index and to time steps when reversely counted from time N to time 1. If at time n the ratio at an index j is not changed, *i.e.*, it is still $u^{j-1}-1$, then at time previous (here means greater) to n, the ratio at the index j is all $u^{j-1}-1$; and at time n, for all index that is greater than j, the ratio is also not changed, *i.e.*, for all index $k>j$, the ratio is still $u^{k-1}-1$. Hence at time step n, we only need to calculate the ratio up to the index where the discounted

back ratio (or ratio if continue) is greater than that if exercised immediately. And once the optimum strategy is continue, at the next time step (here means smaller) it will also continue.

5. The ratio at the index j needs to be calculated from the ratio at the index $j-1$ and $j+1$ at the previous step, when $j \geq 2$. We need the original ratio at the index j to calculate the new ratio at the index $j+1$. So we can renew the ratio at j after we calculated ratio at $j+1$ by store the new ratio at j at a temporary variable, because all the calculation for ratio index greater than $j+1$ will no longer depends on the original ratio at the index j.

The results are that we almost relieve the memory constraint for this algorithm (for a computer with 1G byte memory it can calculate up to 400 billion time steps without run out of memory in theory), and also can calculate the continuous exercising lookback American put option value by extrapolation to up to 4 or 5 decimal accuracy in real time (within one or two seconds).

Henceforward we will focus on the improvement of the translated lattice method and referring audience to other sources for related topics. Chapter 2 introduces the state matrix concept by example and then the concept of "option ratio" and "exercise ratio", the "uniformity" and monotonicity property of these ratios, and the "exercise propagation" and "exercise barrier" property of these ratios, the original RBCV algorithm and our revised algorithm upheld by these properties. Numerical results are presented to show the efficiency of the revised algorithm against the original algorithm. Chapter 3 collects the proofs of all the claims made in Chapter 2 and some preliminary works that I have conducted in the direction of analysis the complexity of our revised algorithm. Chapter 4 discusses the possibility of extending our revised algorithm to other types of lookback options. Chapter 5 summarizes our findings and principles and outlines two more possible situations of application of our approach.

Chapter 2

Algorithm and Numerical Results

In this chapter, we will present a binomial valuation framework for valuing the floating strike lookback American put option. The key of our proposed algorithm exploits many of the combinatorial structures inherent in pricing the floating strike lookback option in a binomial lattice setting. We describe these properties in details in Section 2.1. Section 2.2 presents our proposed algorithm and we also contrast it to the existing algorithm. Numerical results are conducted in Section 2.3.

2.1 Binomial Valuation of Floating Strike Lookback Option

In this chapter, we will mainly consider the floating strike lookback American put option valuation. The reason for only considering this particular type of option will become obvious by the end of Chapter 4. Recall that for the floating strike lookback option, the payoff of the option if it is exercised at time t is the difference between the maximum stock price todate (i.e. between time 0 and time t) and the current stock price. We will denote this maximum value as the *path maximum state* variable and the current stock price as the underlying state variable; hence the option value depends on these two state variables.

Recall also that the classic Cox, Ross, and Rubinstein binomial lattice model is a discrete approximation of the continuous-time geometric Brownian motion. This model is known for its capability of capturing the early exercise feature of the American options. The binomial model works by first partitioning the time space into N time periods with the time (index) takes on value $n = 1, 2, ..., N + 1$. (Here we adopt the Matlab's usage of indexing that starts from 1 instead of starting from 0 as in the C programming language's convention.) At each time index, the stock price in the next time period can either go up with a multiplicative factor u and with probability p or go down with a multiplicative factor d and with probability $1 - p$. Furthermore, the classic way of defining u, d and p are:

$$\begin{aligned} u &= e^{\sigma\sqrt{\frac{T}{N}}}, && (2.1.1) \\ d &= \frac{1}{u}, && (2.1.2) \\ p &= \frac{a-d}{u-d}, && (2.1.3) \\ a &= e^{r\frac{T}{N}}, && (2.1.4) \end{aligned}$$

where σ is the volatility of the underlying state variable, T is the maturity of the option in years, and r is the risk-free interest rate.

A graphical depiction of a binomial model is given in Figure 2.1 with $N = 6$ time periods. Before we describe our approach to pricing a lookback option using a binomial lattice, we first introduce several terminologies associated with the structure of the binomial model as depicted in Figure 2.1. These will be useful for our subsequent discussions. First note that the time index starts from 1, instead of the usual convention of 0. Second, there are n possible stock prices at time index n. Third, the nodes that lie on the same horizontal level have the same stock prices. With $N = 6$, there are 13 possible such prices and we label these stock values as $j = 1, 2, \ldots, 13$. Consequently, the root (or the first node) of the lattice is labeled as $(1, 7)$ and the underlying value index $j = 7$ corresponds to the stock level S_0, which we denote as the S_0-level. Similarly, using the same labeling system, node A is denoted as (5,9). In the next time step, node A evolves to the up-state (6,10) (i.e. node C) with probability p or the down-state (6,8) (i.e. node B) with probability $1 - p$.

The binomial lattice in Figure 2.1 also provides another equivalent way of labeling the nodes. By defining k as the *state index* so that in our example, the state index admits values $k = 1, 2, \ldots, 7$. Also, nodes $(n, 1)$ and (n, n) denote, respectively, the lowest and

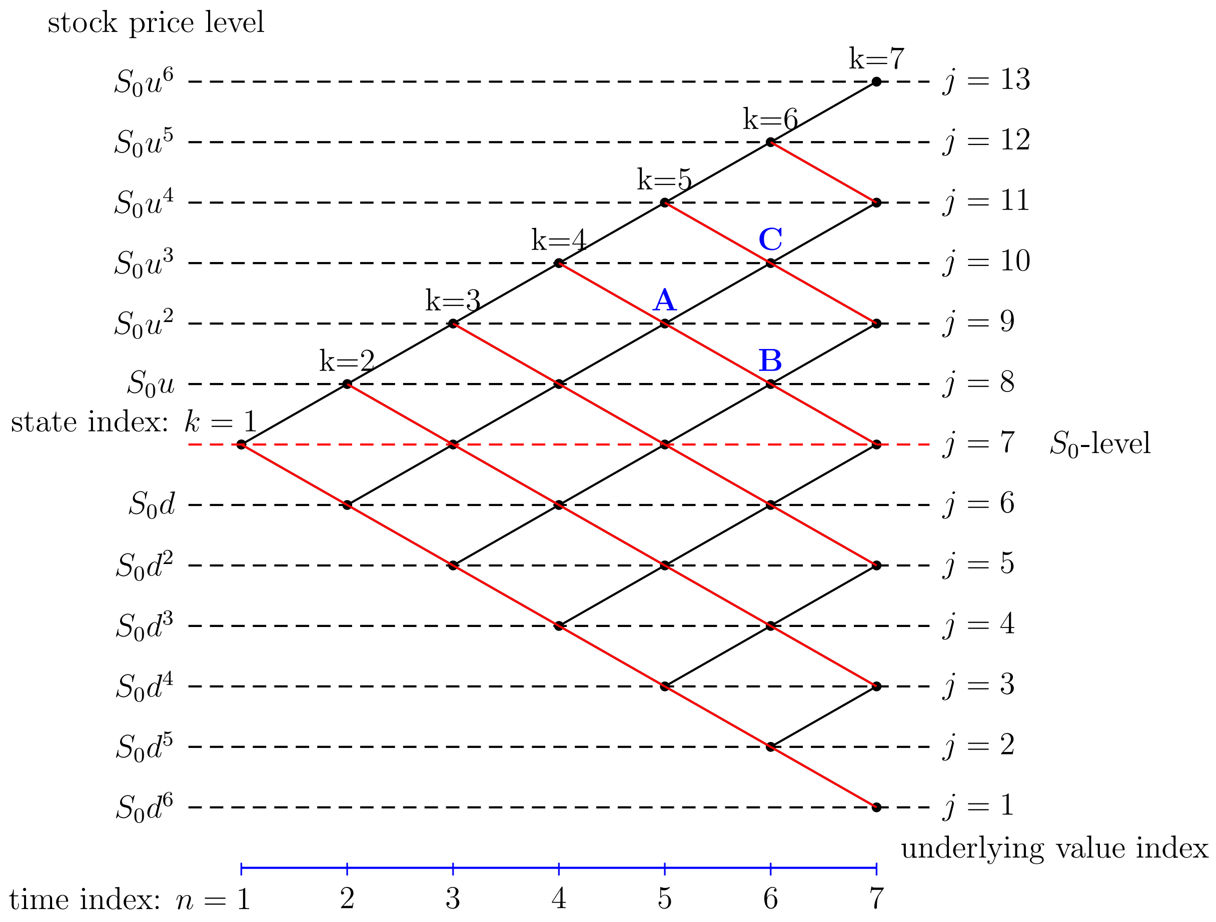

Figure 2.1: Graphical Depiction of a Binomial lattice with $N = 6$

upper most nodes at each time step n. Nodes A, B, and C can be re-labeled respectively as (5,4), (6,4) and (6,5). Furthermore, starting from node $(1,1)$ with initial stock price S_0, it takes $k-1$ up movements and $n-k$ down movements to reach node (n,k). This implies that the stock price for a general node (n,k) is given by $S_0 u^{k-1} u^{-(n-k)} = S_0 u^{2k-n-1}$.

Note that for a fixed n, the state index k and the underlying value index j are one-to-one correspondence and satisfies the relationship $j = 2k - n + N$. This implies that for a given fixed n and N, we can uniquely label the nodes using either the state index k or the underlying value index j.

We now proceed to considering the path maximum state variable in the binomial lattice framework. Let us first consider the number of paths that can be traced from node $(1,1)$ to node (n,k). When $k=1$ or $k=n$, there is only a single path since these nodes represent the extreme nodes in the binomial lattice. For general node (n,k), it can be shown easily that there are $\begin{pmatrix} n-1 \\ k-1 \end{pmatrix}$ paths emanating from node $(1,1)$ to node (n,k). This implies that for each node (n,k), there are at most $\begin{pmatrix} n-1 \\ k-1 \end{pmatrix}$ possible path maximum values and hence at most $\begin{pmatrix} n-1 \\ k-1 \end{pmatrix}$ calculations. A crude way of pricing the lookback option is to rely on the standard backward recursive approach and enumerate all the possible paths at each node. This is computationally not feasible, even for modest size N, since the number of paths grows exponentially with N.

One approach of reducing the computational burden is to reduce the number of paths that needs to be considered at each node. For example, for all the paths that do not traverse above the S_0-level, the path maximum is S_0, and hence has zero payoff for the floating strike lookback option. Recognizing this simple fact already reduces the number of paths quite substantially. For all other paths, it turns out that there exists a recursive relationship between the path maximum at time step n and the path maximum at time step $n+1$. This property can be exploited to enhance the binomial recursive valuation and this is the essence of our proposed enhancement.

We now describe these properties. Let us first revisit the node A in Figure 2.1. For this particular node, there are $\begin{pmatrix} 4 \\ 3 \end{pmatrix} = 4$ admissible paths emanating from the initial node S_0. These four paths are enclosed in the parallelogram $(1,1), (4,4), A, (2,1)$ and the maximum stock price along such a path is attained at the highest point in each path. Among these paths, the highest one is the upper edge traced by $(1,1) - (4,4) - A$ and the lowest one is the lower edge traced by $(1,1) - (2,1) - A$ of the parallelogram. For this particular node, the path maximums can only spanned between the value index 9 and 10 even though there are a total of 4 paths. We denote the possible range of the maximum stock prices as the *path maximum state interval*. For node A, its path maximum state interval is $(9,10)$.

Remark 2.1.1. For any node below the S_0-level, the left end point of the path maximum state interval must be S_0.

With regard to our example in Figure 2.1, the remark above implies that for any node that lies below the value index 7, the lowest value of the path maximum state interval 7.

We can use the *state matrix* to represent all the path maximum state intervals for all time index n and all state index k. The state matrix corresponding to a binomial lattice with $N = 6$ is illustrated in Table 2.1.

Let us now interpret this table. For time index $n = 7$, the state index is defined for $k = 1, 2, 3, \ldots, 7$ that corresponds to stock price value index $j = 1, 3, 5, \ldots, 13$. For each k, the path maximum state interval is indicated by the range [ ] in the state matrix in Table 2.1. For example when $k = 7$, the path maximum state interval is [13] since there is only one admissible path from (1,1) to (7,7) and its maximum stock price must be $S_0 u^6$, which corresponds to value index 13. Similarly when $k = 5$, even though there are 30 admission paths from (1,1) to (7,5), the state matrix indicates that its path maximum state interval is [9, 11].

For time index $n = 6$, the state index is defined for $k = 1, 2, 3, ..., 6$ with corresponding value index $j = 2, 4, 6, \ldots, 12$. This represents the other half of our state matrix rows. For each k the possible path maximum state variable takes values in an interval which is indicated by ( ) in our state matrix.

Similarly for $n = 5$ and $n = 4$, the path maximum state intervals are denoted by [} and ($\rangle$, respectively. Note that the path maximum state intervals for $n = 5$ and $n = 4$ are subset of the path maximum state intervals at time index 7 and 6, respectively.

Table 2.1: State matrix for a binomial lattice with time periods N=6

13	7	8	9	10	11	12	[13]		7
12	7	8	9	10	11	(12)	13	6	
11	7	8	9	10	[11}	12]	13		6
10	7	8	9	(10 ⟩	11)	12	13	5	
9	7	8	[9	10}	11]	12	13		5
8	7	(8	9 ⟩	10)	11	12	13	4	
7	[7	8	9}	10]	11	12	13		4
6	(7	8 ⟩	9)	10	11	12	13	3	
5	[7	8}	9]	10	11	12	13		3
4	(7 ⟩	8)	9	10	11	12	13	2	
3	[7}	8]	9	10	11	12	13		2
2	(7)	8	9	10	11	12	13	k=1	
j=1	[7]	8	9	10	11	12	13		k=1
m=	7	8	9	10	11	12	13		
	Path maximum state interval							n=6	n=7

Remark 2.1.2. For a binomial lattice with N time steps, the path maximum state interval corresponds to node (n, k) is given by

$$(\ max(N-n+2k, N+1),\ \ N+k\ ), \tag{2.1.5}$$

which is located at row $j = N - n + 2k$ in the state matrix.

To demonstrate explicitly the relationship between the successive path maximum state intervals, Table 2.2) extracts the relevant path maximum state intervals from Table 2.1 for $n = 7$ and $n = 6$. This table shows that the combined path maximum state intervals produce a *slant triangle* bounded by a perpendicular line, a 45° and a 67.5°. We refer this as the *state triangle.*

We now make two observations regarding the state triangle. The first observation corresponds to nodes below the S_0-level while the second observation pertains to nodes at or above the S_0-level.

- It is easy to explain the first observation by considering node (6,2) with the path maximum state interval is spanned by $(7, 8)$ and the node is below the S_0-level. The state triangle indicates that if the stock price were to move down to node (7,2), then the path maximum state interval becomes $(7, 8)$ while if the stock price were to move up to node (7,3), then the path maximum state interval expands to $(7, 9)$. This implies that for a node that is below the S_0-level, the path maximum state interval at time step $n+1$ always encompasses the path maximum state interval at time step n, regardless of the underlying stock moving up or moving down.

- To explain the second observation, let us consider node (6,4) with the path maximum state interval $(8, 10)$. In the next time period, the path maximum interval either evolves to $(7, 10)$ in the down-state or to $(9, 11)$ in the up-state. This implies that if the underlying stock price drops in the next time period, the revised path maximum state interval still spans the path maximum from the previous time step. On the other hand, if the current path maximum at node (6,4) is the smallest admissible underlying value index 8 (i.e. S_0u) and followed by an up movement, the new attainable maximum increases to S_0u^2 or the value index increases to 9. Consequently, for nodes above the S_0-level, as long as the current path maximum is not the left-most end point of the path maximum state interval, the path maximum is part of the path maximum state interval in the next time interval.

We now formally summarize the above observations in the following lemma:

Lemma 2.1.3.

(i) Suppose the nodes are below the S_0-level. The path maximum state interval at time step n is a subset of the path maximum state interval at time step $n+1$, irrespective of whether the underlying stock moving up or moving down.

(ii) Suppose the nodes are at or above the S_0-level. The path maximum state interval at time step n is a subset of the path maximum state interval at time step $n+1$, except when the current path maximum equals to its smallest admissible path maximum and the stock price goes up. In the latter case, the path maximum increases by one.

Tables 2.3, 2.4, and 2.5 demonstrate the evolution of the state triangles as we recursively move backward in time. As we move one time step backward, the state triangle shrinks by moving up one level the 67.5° edge of the triangle until we collapse to only one path maximum when $n = 1$ as shown in Tables 2.3, 2.4, and 2.5.

Table 2.2: State triangles: $N = 6, n = 7$ and $n = 6$

13							13		7
12						12		6	
11					11	12			6
10				10	11			5	
9			9	10	11				5
8		8	9	10				4	
7	7	8	9	10					4
6	7	8	9					3	
5	7	8	9						3
4	7	8						2	
3	7	8							2
2	7							k=1	
j=1	7								k=1
m=	7	8	9	10	11	12	13		
	Path maximum state interval							n=6	n=7

In the original binomial lattice method for American lookback put option evaluation, we start from time N+1, calculate the exercise value of the option for all the state k and all the path state at k; for time N and each state k at time N and each path state M at the node (N, k),we calculate the continue value as the discounted back average option value at time $N + 1$ due to up or down movement of underlying asset, using the elongated path to calculate the corresponding path state at time N+1 and using this path state and underlying state to get option value at time $N + 1$; if this continue value is greater than the immediate exercise value, we will continue and using the continue value as the option value at time N, otherwise using exercise value; the same procedure is repeated as we do

Table 2.3: State triangles: $N = 6, n = 6$ and $n = 5$

13									
12						12			6
11					11			5	
10				10	11				5
9			9	10				4	
8		8	9	10					4
7	7	8	9					3	
6	7	8	9						3
5	7	8						2	
4	7	8							2
3	7							k=1	
2	7								k=1
j=1									
m=	7	8	9	10	11	12	13		
	Path maximum state index							n=5	n=6

from $N+1$ to N to other time index n until we arrive at $n = 1$, when we will have unique state and path state, and the option value is the option price at time 1.

At time n if the option value is the immediate exercise value, we say the optimum strategy is to exercise, otherwise we say the optimum strategy is to continue.

Corollary 2.1.4. *If the optimum strategy at time step $n+1$ is exercised regardless of up or down movement when emanating from time step n, then the optimum strategy at time step n is also exercised, except for the special case where the node is above the S_0-level and the path maximum equals to its smallest admissible path; in that case, the optimum strategy is to continue.*

At some path states, the option value is the exercise value while the other states are the continuation value. At time step $n = N+1$, all path maximums, as represented by the largest slant triangle, must correspond to the exercise value. It also follows from the above

Table 2.4: State triangles: $N = 6, n = 4$ and $n = 3$

13									
12									
11									
10				10					4
9			9					3	
8		8	9						3
7	7	8						2	
6	7	8							2
5	7							k=1	
4	7								k=1
3									
2									
j=1									
m=	7	8	9	10	11	12	13		
	Path maximum state index							n=3	n=4

corollary that at time N, only the upper $45°$ edge of the slant triangle is the "continuation state" while all other states are the "exercise state". By induction we have the following corollary:

Corollary 2.1.5. *At time period n, at most the upper $N + 1 - n$ $45°$ parallel lines of the state triangle can be the continuation state, while all the states on the lower lines are optimal to exercise early.*

This pattern is illustrated in Figure 2.2. The state triangle is somewhat like a tie (This pattern is more obvious with bigger N, see also another Figure 2.3 with $N = 20$). Some of the $45°$ parallel lines are drawn on the figure. For example, the upper most edge of the $45°$ line is the continue path state at time $n = 6$ (or the first backward iteration), and counting downwards, the second parallel line is the boundary of possible continue states at time $n = 5$, the second backward iteration, the thirds parallel line is the boundary of

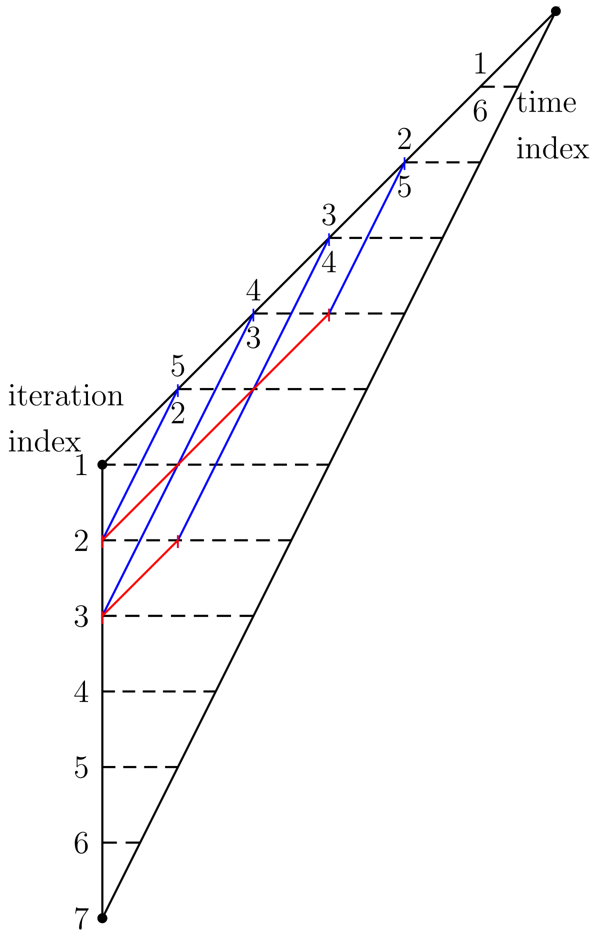

Figure 2.2: State triangle for a lattice with time periods N=6

Table 2.5: State triangles: $N = 6, n = 2$ and $n = 1$

m=	7	8	9	10	11	12	13	n=1	n=2
13									
12									
11									
10									
9									
8		8							2
7	7							k=1	
6	7								k=1
5									
4									
3									
2									
j=1									
	Path maximum state index								

possible continue states at time $n = 4$, and so on.

Remark 2.1.6. From Remark 2.1.2 we also know that at time n we only need to calculate path states that are at or to the left of the $(N + 2 - n)^{\text{th}}$ 67.5° parallel line counting from right leftwards (or the n^{th} line counting from left rightwards). The effect is that, at time n, we only need to calculate continue value at most for the upper $\min(n, N + 1 - n)$ 45° parallel lines of the state triangle.

In our example with $N = 6$ and for $n = 3$ or 4, we need to calculate to the 3rd position, for $n = 2$ or 5, we need to calculate to the 2nd position, and for $n = 1$ or 6, we only need to calculate continue value for the 1st line.

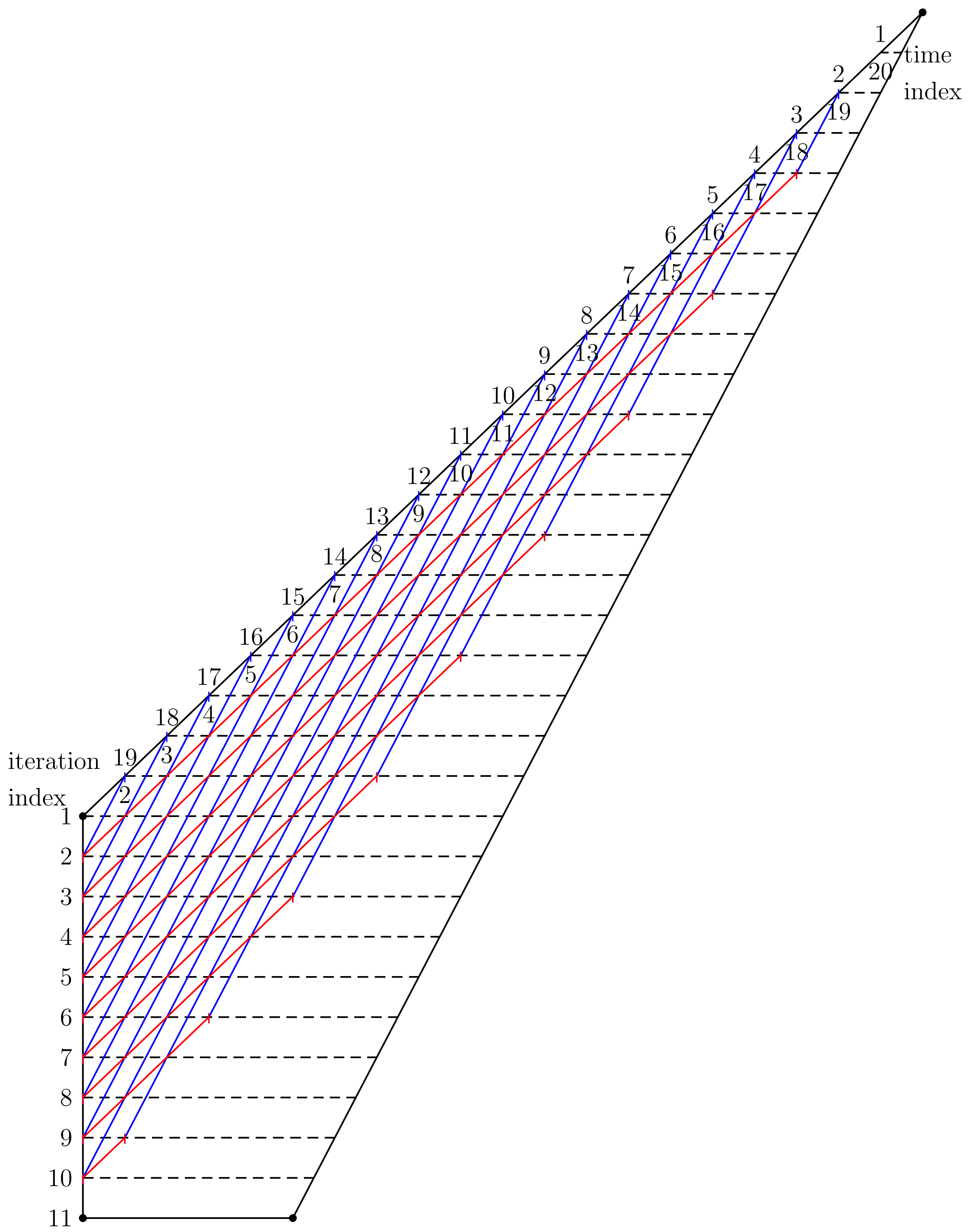

Figure 2.3: State triangle for a lattice with time periods N=20

2.2 Proposed Algorithm for Pricing Floating Strike Lookback Option

In this section, we will demonstrate how to exploit the properties established in the previous section to enhance the binomial valuation of the floating strike lookback put option. We will see that by using these properties, we can reduce the calculation complexity of the original RBCV algorithm to half. Subsection 2.2.1 reproduces the algorithm proposed by RBCV and Subsection 2.2.2 describes the proposed enhanced algorithm.

2.2.1 RBCV Algorithm

For a general node A(n,k) with path maximum state M_A, the option value will be a function of M_A and the underlying stock price P_A:

$$O_A = O_A(M_A, P_A).$$

Let us first define the following two ratios:

Definition 2.2.1.

$$L_A \equiv \frac{O_A}{P_A}, \tag{2.2.1}$$

$$Y_A \equiv \frac{M_A}{P_A}. \tag{2.2.2}$$

We can restate RBCV's algorithm in the following theorem:

Theorem 2.2.2 (RBCV,1992). *L_A and Y_A are constant along every 45° parallel lines of the state triangle and have the following recursion relationship:*

$$L_1^n = \frac{1}{a}[L_1^{n+1}up + L_2^{n+1}d(1-p)], \tag{2.2.3}$$

$$L_k^n = \max\left(Y_k - 1, \frac{1}{a}[L_{k-1}^{n+1}up + L_{k+1}^{n+1}d(1-p)]\right) \text{ when } k > 1, \tag{2.2.4}$$

where k is the index of the line and n is time index.

By combining this theorem and Remark 2.1.6, we can achieve a slight improvement of the above algorithm as follows:
RBCV algorithm:

- $L^{N+1} \leftarrow$ calculate initial exercises value: $0, u-1, ..., u^{[\frac{N+1}{2}]}-1$,
- for $n = 1 : [\frac{N}{2}]$ and for k=1:n using equation 2.2.3 and 2.2.4 to update L_k^{N+1-n},
- for $n = N - [\frac{N}{2}] : 1$ and for k=1:n using equation 2.2.3 and 2.2.4 to update L_k^n,
- L_1^1 is option price.

Remark 2.2.3. Every step in the iteration we have a constant times of floating point (multiplication) calculations, if for simplicity we normalize it as 1, then the total floating point calculation in our algorithm will be:

$$1 + 2 + ... + [\frac{N}{2}] + N - [\frac{N}{2}] + N - [\frac{N}{2}] - 1 + ... + 1 = [\frac{N+1}{2}][\frac{N+2}{2}], \tag{2.2.5}$$

which is approximately half of the original calculations:

$$N + N - 1 + N - 2 + ... + 1 = \frac{N(N+1)}{2}. \tag{2.2.6}$$

2.2.2 Proposed enhanced algorithm

In this subsection, we first provide some key results and then demonstrate how to use these results in improving the binomial valuation of the lookback options. The prove of these results are collected in the next chapter.

Lemma 2.2.4. *L_k^n is strictly increasing with respect to k and nonincreasing with respect to n.*

Proposition 2.2.5. *L_1^n is strictly decreasing with respect to n.*

Theorem 2.2.6. *If $L_k^n = Y_k - 1$, then for $\forall j > k$, $L_j^n = Y_j - 1$.*

Corollary 2.2.7. *If $\exists$ n such that $L_k^n > Y_k - 1$, then for $\forall j < n$, L_k^j is strictly decreasing with respect to j, that is to say $L_k^{n-1} > L_k^n$.*

Corollary 2.2.7 implies that for a fixed k, L_k^n will remain to be the "exercise ratio", but once it reaches above that level it will then strictly increasing when n decreasing. Theorem 2.2.6 means there exits an exercise boundary: once the option ratio hits this boundary, it will remain to be the exercised ratio (for a fixed n); this is in a sense duality to the corollary. While the RBCV algorithm 2.2.2 only utilizes the "ratio uniformity" property, we will further utilize all the other properties: the "exercise barrier" as expressed by Theorem 2.2.6 and Corollary 2.2.7, "ratio monotonicity" of this section's Lemma 2.2.4 and the "exercise propagation" property as represented by Corollary 2.1.4 and Corollary 2.1.5. Here is our revised algorithm:
Revised algorithm:

```
• $L^{N+1} \leftarrow$ calculate initial exercises value: $0, u-1, ..., u^{[\frac{N+1}{2}]}-1$
• for $n = 1 : [\frac{N}{2}]$ and
    – for k=1 using equation 2.2.3 to update $L_1^{N+1-n}$
    – for k=2:n using equation 2.2.4 to update $L_k^{N+1-n}$ until $L_k^{N+1-n} = Y_k - 1$
• for $n = N - [\frac{N}{2}] : 1$ and
    – for k=1 using equation 2.2.3 to update $L_1^n$
    – for k=2:n using equation 2.2.4 to update $L_k^n$ until $L_k^n = Y_k - 1$
• $L_1^1$ is option price
```

Remark 2.2.8. A further simplification to our revised algorithm is possible by using the following relationship:

$$\begin{aligned} L_k^n &= [L_{k-1}^{n+1}up + L_{k+1}^{n+1}d(1-p)]/a \\ &= L_{k-1}^{n+1} + [L_{k+1}^{n+1} - L_{k-1}^{n+1}]d(1-p)/a, \end{aligned} \tag{2.2.7}$$

instead of the equation 2.2.4 in the above algorithm. In the comparison we can use either L_k^{n+1} or L_k^{n+2} instead of the $Y_k - 1$; and in each time step we need only to perform one floating point comparison due to the monotonicity property.

Table 2.6: Comparison of computing time (in seconds): Matlab code implementation

Time periods (N)	Proposed algorithm	RBCV algorithm
1 million	2	8313
2 million	13	22776

2.3 Numerical results

In this section, we provide some numerical illustrations on our proposed algorithm. The reduction of the proposed algorithm in calculation load is related to the question of when L_k^n will begin "continue value" (for fixed k); some discussions are given in Chapter 3. Here are some numerical results: for the following parameters

$$S_0 = 100, N = 1000, \sigma = 0.25, r = 0.05, T = 1, \tag{2.3.1}$$

the largest k reached in our algorithm is 48, which means in place of the 1000 lines in the original RBCV algorithm we only need to calculate up to 48 lines. When $N = 10,000,000$, maximum k reached is about 10,000, a reduction of 1000. And the actual run time of our algorithm for N=2,000,000 is 13 seconds vs. the original's 22776 seconds, a reduction of 1751 times. This little extra "until" has done an astounding job with thousands of time reduction (see Table 2.6).

The memory requirement is now reduced to depend linearly on the maximum k: if we use one temporary variable to store L_{k-1}^n while calculating L_k^n and after that update L_{k-1}^n, then for a different n we can reuse the same variable L_k. In my laptop with 1 GB memory, we can calculate a lattice with up to 432 million time periods if we use $N/2$ storage. If we use max k storage, we tried and guess it can deal with more than 432 billion time periods (we will need almost 5000 years to finish this calculation if we permit it to run), thus leave the memory constraint a non-constraint. (But if we do not use this swap method, and instead, using double amount of memory and alternating between two whole sequences, the computing speed will be boosted somewhat: a trade off between memory space and time.)

The original RBCV algorithm has quadratic computing complexity that is obvious from

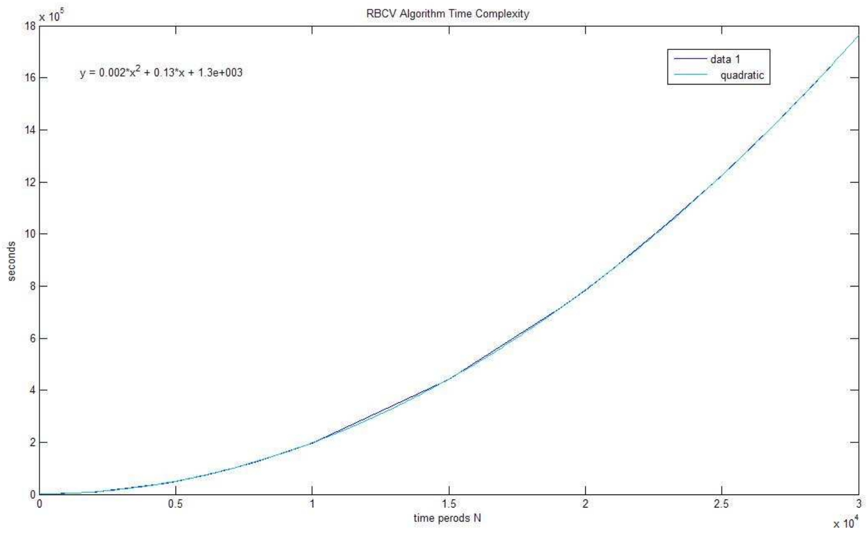

Figure 2.4: Matlab code computing time of RBCV algorithm

equation 2.2.6 and can also be shown by the actual calculating time graph in Figure 2.4.

The computed value is in Figure 2.5.

Also see our revised algorithm time usage in Figure 2.6

and the computed value in Figure 2.7.

The value graph is very much like a power function. And from Table 2.7's almost constant ratio $\frac{f(4x)-f(2x)}{f(2x)-f(x)}$

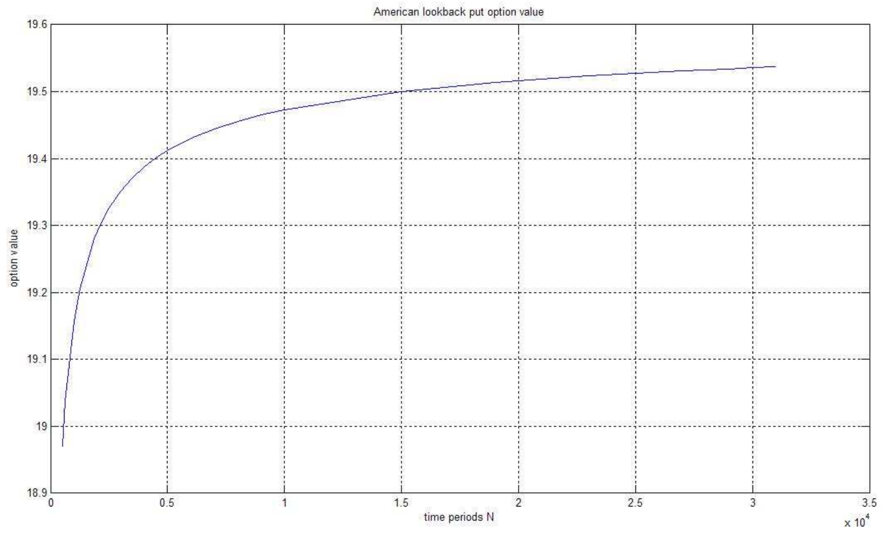

Figure 2.5: Matlab code computed value of RBCV algorithm

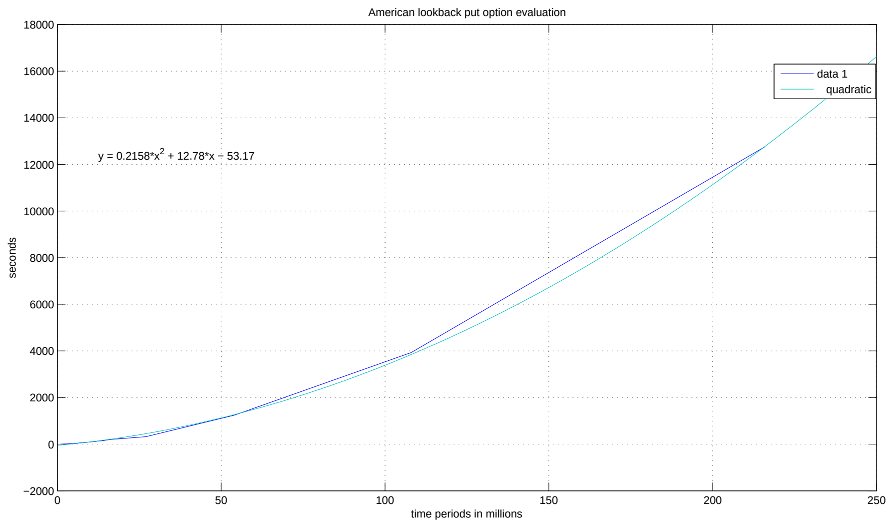

Figure 2.6: C code computing time of our revised algorithm

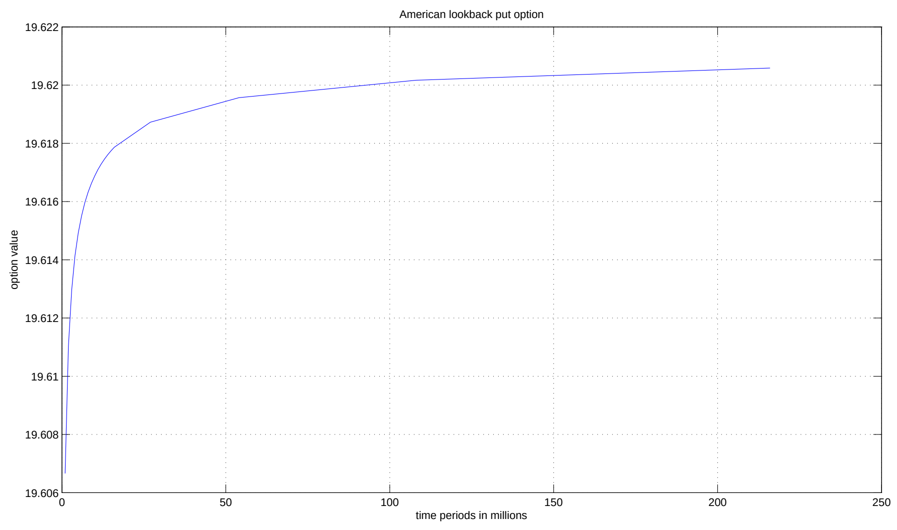

Figure 2.7: C code computed value of our revised algorithm

Table 2.7: Quotient of difference, note that $\frac{1}{\sqrt{2}} = 0.70710678$

x	625	1250	2500	5000	10000	20000
$\frac{f(4x)-f(2x)}{f(2x)-f(x)}$	0.715114917	0.712786765	0.711165286	0.7099835	0.709148086	0.708559778
x	40000	80000	160000	320000	640000	1280000
$\frac{f(4x)-f(2x)}{f(2x)-f(x)}$	0.70813393	0.7078355577	0.707623202	0.707472162	0.707365604	0.707289945
x	1000000	2000000	4000000	8000000	16000000	32000000
$\frac{f(4x)-f(2x)}{f(2x)-f(x)}$	0.707314247	0.707250955	0.70720726	0.707228303	0.707160446	0.70690663

Table 2.8: Predicted continuously exercisable American lookback put option value

x	625	1250	2500	5000	10000	20000
prediction	19.62843339	19.62500506	19.62332641	19.62246474	19.62203527	19.62182185
x	40000	80000	160000	320000	640000	1280000
prediction	19.62171276	19.62165878	19.62163163	19.62161798	19.62161117	19.62160776
x	1000000	2000000	4000000	8000000	16000000	32000000
prediction	19.62160872	19.62160644	19.62160532	19.6216057	19.62160483	19.62160254

we guess the function form of our option value as a function of time periods will be:

$$f(x) = c - \frac{c_1}{x^m}. \tag{2.3.2}$$

Based on the asymptotic property of our numerical results and that $\frac{1}{\sqrt{2}} = 0.707106781186547$, we conjecture that m=$\frac{1}{2}$.

Then we can predict $f(\infty)$ by ("two point prediction"):

$$f(N) + (f(2N) - f(N))/(1 - 1/\sqrt{2}); \tag{2.3.3}$$

or predict just locally ("three point prediction"), with the only assumption of constant ratios, by:

$$\begin{aligned} f(\infty) &= f(N) + \frac{f(2N) - f(N)}{1 - \frac{f(4N)-f(2N)}{f(2N)-f(N)}} && (2.3.4) \\ &= \frac{f(2N)^2 - f(N)f(4N)}{2f(2N) - f(N) - f(4N)}. && (2.3.5) \end{aligned}$$

The predicted value is given in Table 2.8 and Figure 2.8.

We can see the prediction is unchanging for up to 5 decimal points when $N \geq 16$ million. And the prediction is already accurate to 4^{th} decimal points when N is 160000 that can be calculated in less than two seconds by our algorithm.

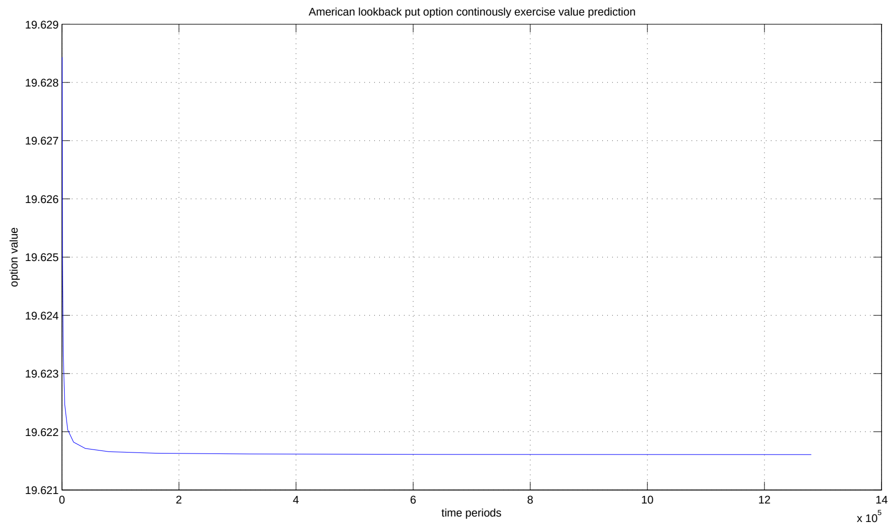

Figure 2.8: Predicted continuously exercisable American lookback put option value

Remark 2.3.1. In implementation we find when time steps are more than 10 million, the accuracy of floating point calculation will affect the final results to the 7^{th} decimal position: the double arithmetic in C is only accurate to 13^{th} decimal position, we must use high precision arithmetic to calculate the coefficient in equation 2.2.7.

Remark 2.3.2. Regarding the $m = 0.5$ conjecture, when we fit it in Matlab using power function $f(x) = ax^b + c$ and using time periods as weight function, heuristically more time steps getting closer to continue limit, we get $a = -0.01495172116627$, $c = 19.62160231517513$, and $b = -0.50028858674636$: different methods give similar results somewhat confirms our conclusion (see Figure 2.9).

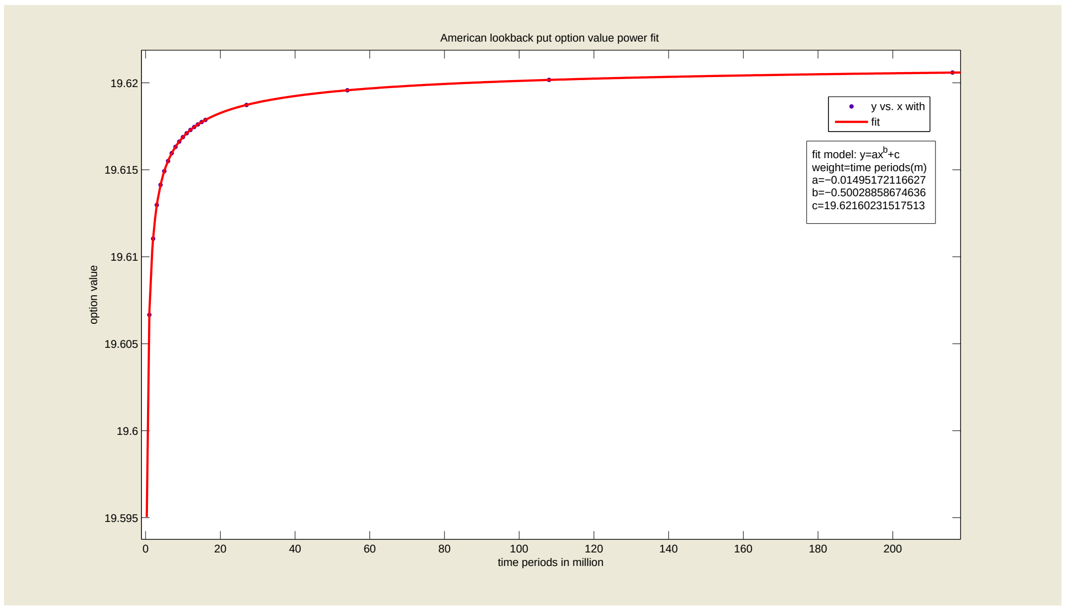

Figure 2.9: Power function weighted least square fit of American lookback put option

Chapter 3

Proofs and Discussions

In this chapter, we provide proofs to various of the results established in the last chapter. Some discussions on these results are also presented.

3.1 Proof of Corollary 2.1.4

Let us denote A as node (n,k), P_A the underlying stock price and M_A a path maximum state at A. At time $n+1$, node A can move up to C or move down to B. By Lemma 2.1.3 we know that in the general case,

$$M_C = M_B = M_A, P_C = P_A u, P_B = P_A d. \tag{3.1.1}$$

The continue value is

$$\begin{aligned}
[(M_C - P_C)p + (M_B - P_B)(1-p)]/a &= M_A/a - P_A[up + d(1-p)]/a \\
&= M_A/a - P_A[u\frac{a-d}{u-d} + d\frac{u-a}{u-d}]/a \\
&= M_A/a - P_A\frac{ua - ud + du - da}{u-d}/a \\
&= M_A/a - P_A 1 \\
&< M_A - P_A,
\end{aligned}$$

the exercise value, as $a > 1$.

In the special case of minimum path state value,

$$M_B = M_A = P_A, M_C = P_C = P_A u, P_B = P_A d. \tag{3.1.2}$$

We will use O to denote option value, E to denote exercise value and C to denote continue value, then

$$\begin{aligned}
E_A &= M_A - P_A \\
&= 0, \\
C_A &= [O_C p + O_B(1-p)]/a \\
&\geq [E_C p + E_B(1-p)]/a \\
&= [(M_C - P_C)p + (M_B - P_B)(1-p)]/a \\
&= P_A(1-d)(1-p)/a \\
&> 0, \\
O_A &= C_A \\
&> E_A,
\end{aligned}$$

as in general $u > 1, d < 1$, and $0 < p < 1$. So in this case, the optimum strategy is to continue. □

3.2 Proof of Theorem 2.2.2

The following proof of RBCV theorem of ours is solely based on the original binomial lattice and our state matrix concept and does not depend on the translated lattice framework that is outlined in John Hull's Technical Note 13.

By inspecting the state matrix, noting that the horizontal index is for M_A and vertical index is for P_A, we see Y_A is $1, u, ..., u^N$ along the parallel 45° diagonal lines calculate downwards starting from the upper edge of the state triangle, which is constant within each parallel line. Henceforth we will index these lines by $1, 2, ..., N+1$, and denote these values $Y_1, Y_2, ..., Y_{N+1}$.

At time N+1,

$$\begin{aligned}L_A &= \frac{O_A}{P_A}\\ &= \frac{M_A - P_A}{P_A}\\ &= Y_A - 1,\end{aligned}$$

is also constant within each of these lines.

This argument and conclusion are also valid for the portion of lines in the triangle where optimum strategy is exercise.

Then at time N, we need only to consider the first line, which are the only continue states by Corollary 2.1.4:

$$\begin{aligned}L_A^N &= \frac{O_A^N}{P_A}\\ &= \frac{[O_C^{N+1}p + O_B^{N+1}(1-p)]/a}{P_A}\\ &= \frac{[E_C^{N+1}p + E_B^{N+1}(1-p)]/a}{P_A}\\ &= \frac{[(M_C - P_C)p + (M_B - P_B)(1-p)]/a}{P_A}\\ &= \frac{P_A(1-d)(1-p)/a}{P_A}\\ &= (1-d)(1-p)/a,\end{aligned}$$

is also independent of the location of M_A or P_A within the same diagonal line, here we use the same notation as before, the only difference is we use an additional superscript to denote time index.

Thus we see for time N, L_A is also only depends on the position of the line.

In general, if L_A is only depends on the position of the line, we will denote its value on the kth line, which is characterized by

$$Y_A = \frac{M_A}{P_A} = u^{k-1} = Y_k. \tag{3.2.1}$$

L_k.

By induction, if at time $n+1$, all the L_A depends only on line index k, then at time n, if M_A is the minimum state, by equation 3.1.2,

$$\begin{aligned}
\frac{M_A}{P_A} &= 1, \\
\frac{M_C}{P_C} &= \frac{P_C}{P_C} \\
&= 1, \\
\frac{M_B}{P_B} &= \frac{M_A}{P_A d} \\
&= u, \\
L_A^n &= \frac{O_A^n}{P_A} \\
&= \frac{[O_C^{n+1}p + O_B^{n+1}(1-p)]/a}{P_A} \\
&= [\frac{O_C^{n+1}}{P_C}up + \frac{O_B^{n+1}}{P_B}d(1-p)]/a \\
&= [L_1^{n+1}up + L_2^{n+1}d(1-p)]/a,
\end{aligned}$$

which is also independent of values of M_A or P_A, so we may write

$$L_1^n = [L_1^{n+1}up + L_2^{n+1}d(1-p)]/a. \tag{3.2.2}$$

If M_A is not the minimum state, and is on the kth line, by equation 3.1.1 and 3.2.1,

$$\begin{aligned}
\frac{M_A}{P_A} &= u^{k-1} \\
&= Y_k, \\
\frac{M_C}{P_C} &= \frac{M_A}{P_A u} \\
&= u^{k-2} \\
&= Y_{k-1}, \\
\frac{M_B}{P_B} &= \frac{M_A}{P_A d} \\
&= u^{k} \\
&= Y_{k+1}, \\
L_A^n &= \max(E_A^n, C_A^n)/P_A \\
&= \frac{\max(M_A^n - P_A, [O_C^{n+1}p + O_B^{n+1}(1-p)]/a)}{P_A} \\
&= \max(Y_A^n - 1, [L_C^{n+1}up + L_B^{n+1}d(1-p)]/a) \\
&= \max(Y_k - 1, [L_{k-1}^{n+1}up + L_{k+1}^{n+1}d(1-p)]/a),
\end{aligned}$$

which is also independent of values of M_A or P_A, so we may write

$$L_k^n = \max(Y_k - 1, [L_{k-1}^{n+1}up + L_{k+1}^{n+1}d(1-p)]/a). \tag{3.2.3}$$

□

3.3 Proof of Lemma 2.2.4

As $u > 1$, $Y_k = u^{k-1}$ is strictly increasing with respect to k, so does $L_k^{N+1} = Y_k - 1$.

Using induction and suppose L_k^{n+1} is strictly increasing with respect to k,

$$\begin{aligned}
L_1^n &= [L_1^{n+1}up + L_2^{n+1}d(1-p)]/a \\
&< [L_1^{n+1}up + L_3^{n+1}d(1-p)]/a \\
&= L_2^n.
\end{aligned}$$

When $a < b$ and $c < d$, $a < b \leq max(b,d), c < d \leq max(b,d)$, then $max(a,c) < max(b,d)$,

$$\begin{aligned} L_k^n &= \max(Y_k - 1, [L_{k-1}^{n+1} up + L_{k+1}^{n+1} d(1-p)]/a) \\ &< \max(Y_{k+1} - 1, [L_k^{n+1} up + L_{k+2}^{n+1} d(1-p)]/a) \\ &= L_{k+1}^n \ when\ k > 1, \end{aligned}$$

$$\begin{aligned} L_1^n &= [L_1^{n+1} up + L_2^{n+1} d(1-p)]/a \\ &> [L_1^{n+1} up + L_1^{n+1} d(1-p)]/a \\ &= L_1^{n+1}, \end{aligned}$$

$$\begin{aligned} L_k^N &= \max(Y_k - 1, [L_{k-1}^{N+1} up + L_{k+1}^{N+1} d(1-p)]/a) \\ &= \max(Y_k - 1, [Y_{k-1} up + Y_{k+1} d(1-p)]/a - 1) \\ &= \max(u^{k-1} - 1, [u^{k-2} up + u^k d(1-p)]/a - 1) \\ &= \max(u^{k-1} - 1, u^{k-2}[up + u^2 d(1-p)]/a - 1) \\ &= u^{k-1} - 1 \\ &= L_k^{N+1} \ when\ k > 1, \end{aligned}$$

because $u^{k-2}[up + u^2 d(1-p)]/a < u^{k-2} u = u^{k-1}$ by proposition 3.6.1 (3)

Using induction by assuming $L_k^{n+1} \geq L_k^{n+2}$ for $\forall k$,

$$\begin{aligned} L_k^n &= \max(Y_k - 1, [L_{k-1}^{n+1} up + L_{k+1}^{n+1} d(1-p)]/a) \\ &\geq \max(Y_k - 1, [L_{k-1}^{n+2} up + L_{k+1}^{n+2} d(1-p)]/a) \\ &= L_k^{n+1} \ when\ k > 1. \end{aligned}$$

□

3.4 Proof of Proposition 2.2.5

Already included in the proof of Lemma 2.2.4 □

3.5 Proof of Theorem 2.2.6

Using induction, suppose the theorem is valid for n+1, $Y_k - 1 = L_k^n \geq L_k^{n+1} \geq Y_k - 1$, then $L_k^{n+1} = Y_k - 1$ and for $\forall j > k$, $L_j^{n+1} = Y_j - 1$,

$$\begin{aligned} L_j^n &= \max(Y_j - 1, [L_{j-1}^{n+1} up + L_{j+1}^{n+1} d(1-p)]/a) \\ &= \max(Y_j - 1, [Y_{j-1} up + Y_{j+1} d(1-p)]/a - 1) \\ &= Y_j - 1, \end{aligned}$$

and $L_j^{N+1} = Y_j - 1$ is always true. □

3.6 Proof of Corollary 2.2.7

By Theorem 2.2.6, $L_{k-1}^n > Y_{k-1} - 1$; and by Lemma 2.2.4, $L_{k-1}^n \geq L_{k-1}^{n+1}$ and $L_{k+1}^n \geq L_{k+1}^{n+1}$. Suppose in contrary to the conclusion we have $L_k^{n-1} = L_k^n$, then $0 = L_k^{n-1} - L_k^n = [(L_{k-1}^n - L_{k-1}^{n+1})up + (L_{k+1}^n - L_{k+1}^{n+1})d(1-p)]/a$, which implies $L_{k-1}^n - L_{k-1}^{n+1} = 0$ or $L_{k-1}^n = L_{k-1}^{n+1} > Y_{k-1} - 1$. Recursively we will get $L_{k-m}^{n-1+m} = L_{k-m}^{n+m} > Y_{k-m} - 1$; let $m = k-1$ we get $L_1^{n+k-2} = L_1^{n+k-1}$. This is possible by proposition 3.6.1 (5): $L_{N+2-n}^n = Y_{N+2-n} - 1$, so $L_k^n > Y_k - 1$ implies $k < N+2-n$ or $n+k-1 < N+1$ by Theorem 2.2.6 again. But this is contradictory to proposition 2.2.5. For general j the conclusion is arrived recursively. □

Redefine $up \equiv up/a, dn \equiv d(1-p)/a$ in the following of this section, using similar algebraic calculation method we can prove

Proposition 3.6.1.

1. *$up + dn = 1$*
2. *$up > dn$*
3. *$up + dn\ u^2 < u$ always true*
4. *$L_2^N = Y_2 - 1$*
5. *$L_{N+2-n}^n = Y_{N+2-n} - 1$*

Proof. 1. already given in the proof of Corollary 2.1.4

2. $up > dn$ iff $u(a-d)/(u-d) > d(1-p)$ iff $u(a-d) > d(u-a)$ iff $ua - 1 > 1 - da$ iff $ua + da > 2$ iff $u + 1/u > 2/a$; but $a > 1$ and $u + 1/u \geq 2\sqrt{u * 1/u} = 2 > 2/a$

3. $1 = up + dn > dn + dn = 2dn, 1 - 2dn > 0$; $up + dn\ u^2 < u$ iff $dn\ u^2 - u + 1 - dn < 0$; as $\Delta = 1 - 4dn(1-dn) = (1-2dn)^2$, the solution of $dn\ u^2 - u + 1 - dn = 0$ is $\frac{1 \mp \sqrt{(1-2dn)^2}}{2dn} = \frac{1 \mp (1-2dn)}{2dn} = (1, \frac{1}{dn} - 1)$; but $0 < dn < 1/2, 1/dn > 2, 1/dn - 1 > 1$, $up + dn\ u^2 < u$ iff $1 < u < \frac{1}{dn} - 1$; $dn = \frac{d(1-p)}{a} = d(1 - \frac{a-d}{u-d})\frac{1}{a} = \frac{du - da}{(u-d)a} = \frac{\frac{1}{a} - d}{u - d}$; $\frac{1}{dn} - 1 = \frac{u-d}{\frac{1}{a} - d} - 1 = \frac{u - \frac{1}{a}}{\frac{1}{a} - d}$; $u < \frac{1}{dn} - 1$ iff $\frac{u}{a} - ud < u - \frac{1}{a}$ iff $\frac{u+1}{a} < u + 1$; but this is always true as $a > 1$ (where we assume $0 < p < 1$, which will be always true if N is sufficient large, then $1 - p > 0, u - a > 0$, and $\frac{1}{a} - \frac{1}{u} > 0$)

4. in the proof of Lemma 2.2.4 we get $L_2^N = L_2^{N+1} = Y_2 - 1$

5. from the last equation and Theorem 2.2.6 we have $L_3^N = Y_3 - 1 = L_3^{N+1}$ and $L_4^N = Y_4 - 1 = L_4^{N+1}$; in our new notation $L_3^{N-1} = max(Y_3 - 1, L_2^N up + L_4^N dn) = max(Y_3 - 1, L_2^{N+1} up + L_4^{N+1} dn) = L_3^N = Y_3 - 1$. The general case can be arrived by induction. □

This is another proof of Corollary 2.1.5.

Property (3) means $C_2^N < E_2^N$, then will ever $L_2^n > Y_2 - 1$? and if so how many iterations are needed to value over this "exercise value"? This question is a special case of the computing complexity of our revised algorithm. The solution turns out to depend on the parameter combinations.

Proposition 3.6.2. $L_2^{N-1} > Y_2 - 1$ *iff*

$$(x^2 - x)y^3 + (x^2 - 2x)y^2 + (x+1)y - x^2 < 0, \tag{3.6.1}$$

where $x = a, y = u$.

Proof. $L_1^{N+1} = 0, L_2^{N+1} = u - 1, L_1^N = dn(u-1), L_2^N = u - 1, L_3^N = u^2 - 1, L_2^{N-1} = dn(u-1)up + (u^2 - 1)dn = (u-1)dn(up + u + 1)$. $L_2^{N-1} > Y_2 - 1$ iff $(u-1)dn(up+u+1) > u - 1$ iff $dn(1 + u + 1 - dn) > 1$ iff $u > dn + \frac{1}{dn} - 2$. $dn = d(1-p)/a = \frac{1}{au}(1 - \frac{a - \frac{1}{u}}{u - \frac{1}{u}}) = \frac{u-a}{au^2 - a}$.

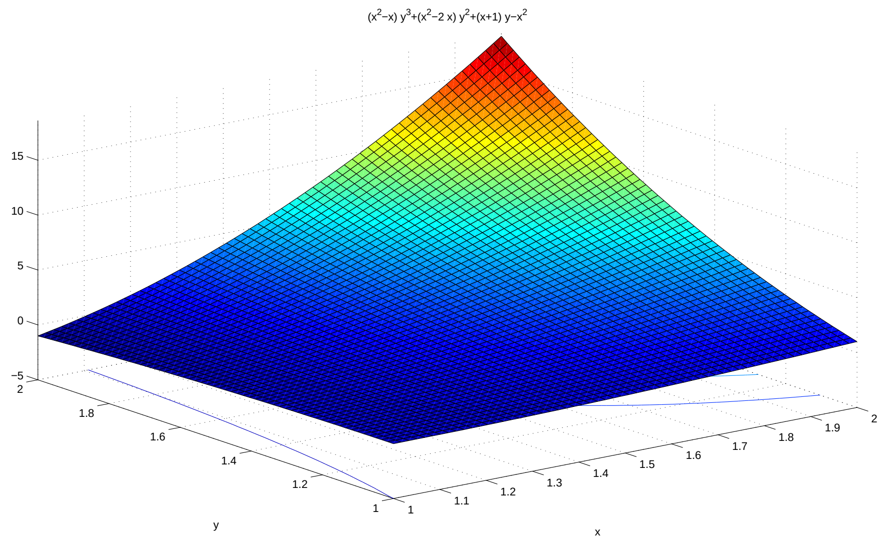

Figure 3.1: Continue region of L_2^{N-1}: $x = a, y = u, f(x,y) < 0$

$u > dn + \frac{1}{dn} - 2$ iff $u > \frac{u-a}{au^2-a} + \frac{au^2-a}{u-a} + 2$ iff $a^2 > (a+1)u + (a^2-2a)u^2 + (a^2-a)u^3 \; if \; u > a$. We assume $0 < p < 1$ and hence $u > a$. $\square$

The solution to this inequality can be seen from the 3-D picture 3.1 and 2-D picture 3.2. For very large N, u and a will be both very close to 1 with u be relatively much bigger than $a, (a,u)$ will naturally belong to the continue region. A closer look at the graph 3.3 reveals that for sufficient large N the continue region is approximately

$$\frac{\sigma}{r}\sqrt{\frac{N}{T}} > 5. \tag{3.6.2}$$

On the other hand if $L_2^{N-1} = Y_2 - 1$, we can prove that for sufficient large N and sufficient small n we will finally have $L_2^n > Y_2 - 1$. Same thing can be said of L_k^n. But a general analysis for estimation of maximum k remains open or a theoretical analysis regarding the complexity of our algorithm is still lacking. Also our algorithm cannot be extended to other kind of lookback options, the following chapter will briefly describe the negative results.

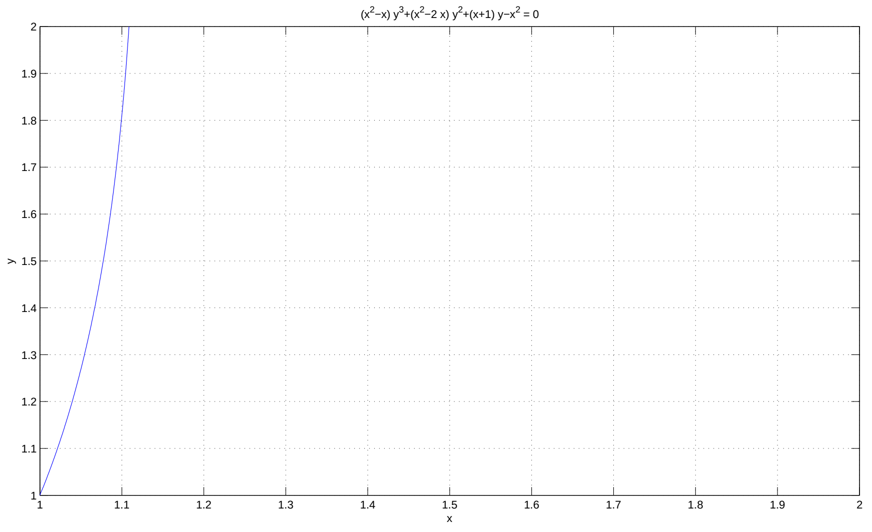

Figure 3.2: Continue region of L_2^{N-1}: $x = a, y = u, y > f(x)$

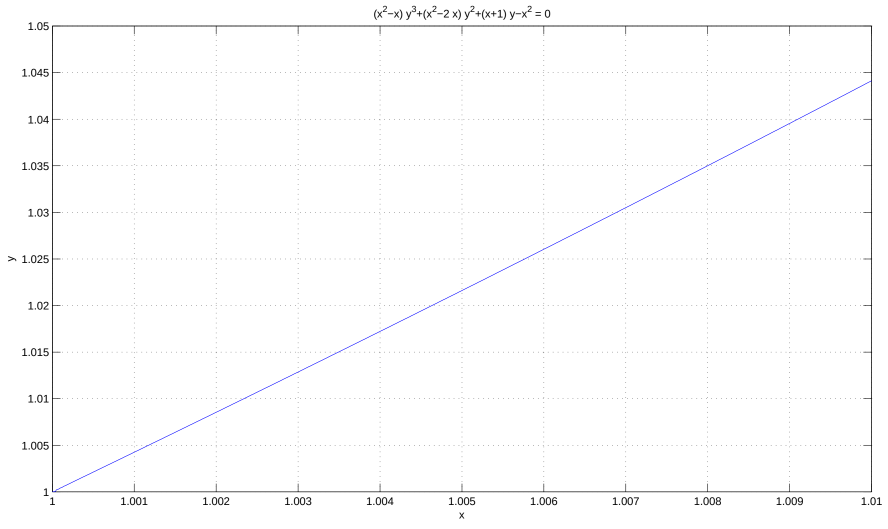

Figure 3.3: Continue region of L_2^{N-1} for sufficient large N

Chapter 4

Other Types of Lookback Options

In this chapter, we discuss the applicability of our proposed algorithm to other types of lookback options including the floating strike lookback call options, fixed strike lookback call option, fixed strike lookback put options.

4.1 Floating strike lookback call options

For American floating strike lookback call option with payoff at time t given by

$$S_t - \min_{k=0,\ldots,t} S_k,$$

we can derive parallel results with only minor modifications of the previous arguments, so only one of the proofs will be given:

Lemma 4.1.1. *Except for the case of the path state index equals its maximum value and the movement is down, and the underlying value is equal to or lower than S_0, it will not change when time index increase one; in the exceptional case the path state index decrease one.*

Corollary 4.1.2. *The optimum strategy is always continuing.*

Proof Let's denote A the node (n,k), P_A the underlying stock price and m_A a path (minimum value) state at A. At time $n + 1A$ going up to C and going down to B. The permissible state is the symmetry image of the slant triangle with respect to the left top vertex in the triangle as exemplified by Table 2.1.

By Lemma 4.1.1 we know in the special case of maximum path state value, the lower edge of the triangle,

$$m_C = m_A = P_A, m_B = m_A d = P_B = P_A d, P_C = P_A u. \tag{4.1.1}$$

We will use O to denote option value, E to denote exercise value and C to denote continue value, then

$$\begin{aligned} E_A &= P_A - m_A \\ &= 0, \\ C_A &= [O_C p + O_B(1-p)]/a \\ &\geq [E_C p + E_B(1-p)]/a \\ &= [(P_C - m_C)p + (P_B - m_B)(1-p)]/a \\ &= (P_A u - P_A)p/a \\ &= P_A(u-1)p/a \\ &> 0, \\ O_A &= C_A \\ &> E_A, \end{aligned}$$

because in general $u > 1$ and $0 < p < 1$. So in this case, the optimum strategy is continuing.

In the other cases,

$$m_C = m_B = m_A, P_C = P_A u, P_B = P_A d. \tag{4.1.2}$$

The continue value is

$$\begin{aligned} C_A &\geq [(P_C - m_C)p + (P_B - m_B)(1-p)]/a \\ &= P_A 1 - m_A/a \\ &> P_A - m_A, \\ O_A &= C_A \\ &> E_A, \end{aligned}$$

the exercise value, as $a > 1$. So the optimum strategy is still continuing. □

Theorem 4.1.3. *L_A and Y_A are constant along every 45° parallel lines of the state triangle and have the following recursion relationship:*

$$\begin{aligned} L_1^n &= [L_1^{n+1}d(1-p) + L_2^{n+1}up]/a, & (4.1.3)\\ L_k^n &= [L_{k-1}^{n+1}d(1-p) + L_{k+1}^{n+1}up]/a) \text{ when } k > 1, & (4.1.4)\end{aligned}$$

where k is the index of the line and n is time index.

Proposition 4.1.4. *L_k^n is strictly increasing with respect to k and strictly decreasing with respect to n.*

This corollary means the value of a floating strike lookback American call option is the same as a floating strike lookback European call option value. While the "ratio uniformity" and "ratio monotonicity" are still hold, but without the other two properties('exercise propagation" and "exercise barrier"), a similar revised algorithm do not exit for this call option. We can still use the available analytical form of European lookback option value to confirm our conjecture regarding two point and three point prediction. Table 4.1 gives the computing time (in seconds) of our revised algorithm for American lookback put option and that of the original RBCV algorithm for American lookback call option in various time steps. It is done in a computer with Intel Pentium(D) 3.40GHz CPU and 2.00GB RAM. It shows a roughly magnitude of the amount of reductions our algorithm made against the original algorithm (This calculation is done in C but the time reduction magnitude is comparable to that using Matlab, compare Table 2.6). The exact value of the European lookback call and put option: C and P are given by formula 1.2.1 and 1.2.2. The two or three point predictions are calculated using formula 2.3.3 or 2.3.4. We may expect that these predictions will have similar accuracy in the American lookback put case.

4.2 Fixed strike lookback call options

For American fixed strike lookback call option with payoff at time t given by

$$(\max_{k=0,\ldots,t} S_k - K)_+,$$

Table 4.1: Computing time (C=20.5521826180488,P=18.7232860368255)

x	250000	500000	1000000	2000000	4000000	8000000
alp time	0	1	2	7	20	57
alc time	157	738	2835	12830	48609	194986
alp value	19.59173395	19.60047554	19.60666040	19.61103556	19.61413017	19.61631885
alc value	20.53233428	20.53814490	20.54225504	20.54516205	20.54721798	20.54867191
$\frac{f(4x)-f(2x)}{f(2x)-f(x)}$	0.707349523	0.707278377	0.707228001	0.707192594		
2-p pred	20.55217298	20.55217780	20.55218021	20.55218141	20.55218201	
3-p pred	20.55218944	20.55218602	20.55218432	20.55218347		

where K is the fixed strike price, we can derive similar results:

Lemma 4.2.1. *Except for the case of the path state index equals its minimum value and the movement is up, and the underlying value is equal to or higher than S_0, it will not change when time index increase one; in the exceptional case the path state index increase one.*

Corollary 4.2.2. *If the optimum strategy at time $n+1$ is exercises regardless of up or down movement when coming from time n, then the optimum strategy at time n is also exercises, except for the special case of path state index equals its minimum value and the underlying value is equal to or higher than S_0; in that case, the optimum strategy is always continue.*

Proof Let's denote A the node (n,k), P_A the underlying stock price and M_A a path state at A. At time $n+1A$ going up to C and going down to B. By Lemma 4.2.1 we know in the general case,

$$M_C = M_B = M_A, P_C = P_A u, P_B = P_A d. \tag{4.2.1}$$

The continue value is

$$\begin{aligned}[(M_C - K)_+ p + (M_B - K)_+(1-p)]/a &= [(M_A - K)_+ p + (M_A - K)_+(1-p)]/a \\ &= (M_A - K)_+/a \\ &\leq (M_A - K)_+,\end{aligned}$$

the exercise value, as $a > 1$: it is optimal to exercise.

In the special case of minimum path state value,

$$M_B = M_A = P_A, M_C = P_C = P_A u, P_B = P_A d. \tag{4.2.2}$$

We will use O to denote option value, E to denote exercise value and C to denote continue value, then

$$\begin{aligned}E_A &= (M_A - K)_+, \\ C_A &= [O_C p + O_B(1-p)]/a \\ &\geq [E_C p + E_B(1-p)]/a \\ &= [(M_C - K)_+ p + (M_B - K)_+(1-p)]/a \\ &= [(M_A u - K)_+ p + (M_A - K)_+(1-p)]/a \\ &= [(M_A - K/u)_+ up + (M_A - K)_+(1-p)]/a \\ &\geq [(M_A - K/u)_+ up + (M_A - K)_+ d(1-p)]/a \\ &\geq [(M_A - K)_+ up + (M_A - K)_+ d(1-p)]/a \\ &= (M_A - K)_+, \\ O_A &= C_A \\ &\geq E_A,\end{aligned}$$

as in general $u > 1, d < 1, 0 < p < 1$ and $(uX)_+ = uX_+, (X - K_1)_+ \geq (X - K_2)_+,\ if\ u > 0, 0 \leq K_1 \leq K_2$. So in this case, the optimum strategy is continuing. $\square$

Even though at time $N+1$, the payoff of an American fixed strike lookback call option only depends on the path state (*i.e.*, the column number of the state matrix), due to the above corollary, the option value will broke the uniformity property during iteration and cannot have a similar algorithm to that of American floating strike lookback put option.

4.3 Fixed strike lookback put options

For American fixed strike lookback put option with payoff at time t given by

$$(K - \min_{k=0,\ldots,t} S_k)_+,$$

where K is the fixed strike price, we can derive similar results:

Lemma 4.3.1. *Except for the case of the path state index equals its maximum value and the movement is down, and the underlying value is equal to or lower than S_0, it will not change when time index increase one; in the exceptional case the path state index decrease one.*

Corollary 4.3.2. *If the optimum strategy at time n+1 is exercises regardless of up or down movement when coming from time n, then the optimum strategy at time n is also exercises, except for the special case of path state index equals its maximum value and the underlying value is equal to or less than S_0.*

Proof Let's denote A the node (n, k), P_A the underlying stock price and m_A a path state at A. At time $n + 1A$ going up to C and going down to B. By Lemma 4.3.1 we know in the general case,

$$m_C = m_B = m_A, P_C = P_A u, P_B = P_A d. \tag{4.3.1}$$

The continue value is

$$\begin{aligned}[(K - m_C)_+ p + (K - m_B)_+(1 - p)]/a &= [(K - m_A)_+ p + (K - m_A)_+(1 - p)]/a \\ &= (K - m_A)_+/a \\ &\leq (K - m_A)_+,\end{aligned}$$

the exercise value, as $a > 1$: it is optimal to exercise. □

No conclusive result can be arrived for the maximum path state case. It seems that all the other cases of lookback options are not amenable to our revised algorithm: it is unique to floating strike American lookback put options. While algorithms utilizing properties of this chapter for floating strike lookback options will have quadratic upper bound calculating complexity with respect to time steps, algorithms for fixed strike lookback will only have cubic upper bound calculating complexity.

Chapter 5

Conclusion

We devised algorithm that is in the original binomial lattice framework and thus simple to understand but yet is efficient enough to be of practical meaning: it can give option value to up to 4 or 5 decimal accuracy in real time. Our elementary approach eliminated the using of more abstruse martingale theory and the concept of absorbing barrier or reflecting barrier, making the well-known RBCV method more accessible to wider audience. On the other hand, our discovery working as stop criteria to the RBCV method makes it more perfect by cast off the vast unneeded calculations, and thousands of times faster than without our addition make our amendment to RBCV a not unnecessary part of it (As for Dai's method, perhaps our amendment to his is unnecessary, because his method is already so efficient. But why not, our addition is not a free lunch but is a free cookie after the lunch. Just a joke).

Other than this concrete Lookback option result, our approach also revealed some general principles or guidelines. In the lattice context we name it "uniformity", "exercise barrier", "monotonicity" and "exercise propagation" (our numerical experiment with ordinary American put and Asian option examples showed their wider applicability). More widely, these principles have their counterpart concepts in other methods. If we understand these principles from their underlying very primitive intention of reducing calculations and storages, we can comprehend still wider methods that cannot be categorized as one of the three major methods: PDE, MC and lattice.

As more examples of applications of these guidelines, we outline two other possible research topics. First, it worth a try to use the trinomial lattice for American put to adjust the parameter so that the strike price is just located at some node and eliminate the different position relation cases analysis. Another possible is to use trinomial lattice in the payoff function form recursion to adjust tree parameter so that the “turn point” of payoffs of up and down node can recombine or overlap somehow and reduce the increase speed of memory usage in each recursion.

Bibliography

[1] A. N. Avramidis, P. L'Ecuyer (2006) *Efficient Monte Carlo and Quasi-Monte Carlo Option Pricing Under the Variance-Gamma Model.* Management Science 52(12):1930-1944

[2] A. D. Andricopoulos, M. Widdicks, P. W. Duck, D. P. Newton (2003) *Universal option valuation using quadrature methods.* Journal of Financial Economics 67(3):447-471.

[3] A. D. Andricopoulos, M. Widdicks, D. P. Newton, P. W. Duck (2007) *Extending quadrature methods to value multi-asset and complex path dependent options.* Journal Of Financial Economics 83(2):471-499.

[4] S. Babbs (2000) *Binomial valuation of lookback options.* Journal of Economic Dynamics and Control 24:1499-1525.

[5] P. P. Boyle (1998) *A Lattice Framework for Option Pricing with Two State Variables.* Journal Of Financial And Quantitative Analysis 23(1):1-12.

[6] M. Broadie, J. B. Detemple (2004) *Option Pricing: Valuation Models and Applications.* Management Science 50(9):1145-1177.

[7] M. Broadie, P. Glasserman, S. G. Kou (1999) *Connecting discrete and continuous path-dependent options.* Finance Stochast 3:55-82.

[8] P. P. Boyle, A. Kolkiewicz, K. S. Tan (2002) *Pricing American derivatives using simulation: a biased low approach.* K.-T. Fang, F.J. Hickernell, H. Niederreiter (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 2000, Springer, Berlin, 181-200.

[9] P. P. Boyle, A. Kolkiewicz, K. S. Tan (2003) *An improved simulation method for pricing high-dimensional American derivatives.* Mathematics and Computers in Simulation 62:315-322.

[10] P. P. Boyle, Y. Lai, K. S. Tan (2005) *Pricing Options Using Lattice Rules* . North American Actuarial Journal 9(3):50-76

[11] J. Barraquand, T. Pudet(1994) *Pricing of American Path-Dependent Contingent Claims.* Journal of Mathematical Finance 6(1):17-51.

[12] P. P. Boyle, Y. Tian(1999) *Pricing lookback and barrier options under the CEV process.* Journal of Financial and Quantitative Analysis 34:241-264.

[13] P. P. Boyle, Y. S. Tian, J. Imai (1999) *Lookback options under the CEV process: a correction.* Journal of Financial and Quantitative Analysis Unpublished Appendixes, Notes, Comments, and Corrections `http://depts.washington.edu/jfqa/hold/342BoyleCrx1.pdf`

[14] R. H. Chan, Y. Chen, K. M. Yeung (2004) *A memory reduction method in pricing American options.* Journal of Statistical Computation and Simulation 74(7):501-511.

[15] A. Conze, R. Viswanathan (1991) *Path dependent options: the case of lookback options.* Journal of Finance 46:1893-1907.

[16] T. Cheuk, T. Vorst(1997) *Currency Lookback Options and Observation Frequency: A Binomial Approach.* Journal of International Money and Finance 16(2):173-187.

[17] R. H. Chan, C. Y. Wong, K. M. Yeung (2006) *Pricing multi-asset American-style options by memory reduction Monte Carlo methods.* Applied Mathematics And Computation 179(2):535-544.

[18] M. Dai (2000) *A Modified Binomial Tree Method for Currency Lookback Options.* Acta Mathematica Sinica, English Series 16(3):445-454.

[19] M. Dai, Y. K. Kwok (2005) *American Options with Lookback Payoff.* SIAM Journal of Applied Mathematics 66(1):206-227.

[20] M. Dai, Y. K. Kwok (2006) *Characterization of optimal stopping regions of American path dependent options.* Mathematical Finance 16(1):63-82.

[21] T. S. Dai, Y. D. Lyuu (2002) *Efficient, exact algorithms for asian options with multiresolution lattices.* Review of Derivatives Research 5(2):181-203.

[22] F. Dubois, T. Lelievre(2004/05) *Efficient Pricing of Asian Options by the PDE Approach.* Journal of Computational Finance 8(2):55-64.

[23] T. S. Dai, Y. D. Lyuu (2007) *An exact subexponential-time lattice algorithm for Asian options.* Acta Informatica 44(1):23-39.

[24] G. Foufas, M. G. Larson (2004) *Valuing European, Barrier, and Lookback Options using the Finite Element Method and Duality Techniques.* `http://www.math.chalmers.se/~foufas/paper1.pdf`

[25] P. Forsyth, K. Vetzal. (2002) *Quadratic convergence for valuing American options using a penalty method.* SIAM Journal on Scientific Computing 23(6):2095-2122.

[26] P. A. Forsyth , K. R. Vetzal, R. Zvan (2002) *Convergence of Numerical Methods for Valuing Path-Dependent Options Using Interpolation.* Review of Derivatives Research 5(3):273-314.

[27] P. A. Forsyth, K. R. Vetzal, R. Zvan (1998) *Convergence Of Lattice And PDE Methods For Pricing Asian Options.* Technical Report, University of Waterloo, Canada `http://www.cs.uwaterloo.ca/research/tr/1998/30/CS-98-30.pdf`

[28] P. A. Forsyth, K. R. Vetzal, R. Zvan. (1999) *A Finite Element Approach to the Pricing of Discrete Look backs with Stochastic Volatility.* Applied Mathematical Finance 6:87-106.

[29] J. Gatheral, Y. Epelbaum, J. Han, K. Laud, O. Lubovitsky, E. Kant, C. Randall (1999) *Implementing option-pricing models using software synthesis.* IEEE Computational Science and Engineering 1(6):54-64.

[30] M. B. Goldman, H. B. Sosin, M. A. Gatto (1979) *Path dependent options: buy at the low, sell at the high.* Journal of Finance 34:1111-1127.

[31] J. C. Hull. (2006) *Options, Futures, and Other Derivatives, Sixth Edition.* Prentice-Hall, Upper Sadder River, New Jersey.

[32] T. R. Klassen (2001) *Simple, fast, and flexible pricing of Asian options.* Journal of Computational Finance 4(30):89-124

[33] T. L. Lai, T. W. Lim(2004) *Exercise regions and efficient valuation of American lookback options.* Mathematical Finance 14:249-269.

[34] T. L. Lai, T. W. Lim (2004) *Efficient valuation of American floating-strike lookback options using a decomposition technique.* working paper of Stanford University. `http://www.stat.nus.edu.sg/~stalimtw/PDF/lb-float.pdf`

[35] F. A. Longstaff, E. S. Schwartz (2001) *Valuing American options by simulation: a simple least-squares approach.* The Review of Financial Studies 14:113-147.

[36] B. Papatheodorou (2005) *Enhanced Monte Carlo Methods for Pricing and Hedging Exotic Options.* `http://web.comlab.ox.ac.uk/oucl/work/mike.giles/psfiles/basileios.pdf`

[37] G. Petrella, S. G. Kou (2004) *Numerical pricing of discrete barrier and lookback options via Laplace transforms.* Journal of Computational Finance 8:1-37.

[38] L. Stentoft (2004) *Assessing the Least Squares Monte-Carlo Approach to American Option Valuation.* Review of Derivatives Research 7:129-168.

[39] M. E. Silva, T. Barbe (2005) *Quasi-Monte Carlo in finance: extending for problems of high effective dimension.* Economia Aplicada 9(4):577-594.

[40] S. K. Dutt, G. M. Welke (2005) *Just-In-Time Monte Carlo for Path Dependent American Options.* `http://faculty.haas.berkeley.edu/welke/images/RMC.pdf`

[41] K. S. Tan, P. P. Boyle (2000) *Applications of randomized low discrepancy sequences to the valuation of complex securities.* Journal of Economic Dynamics and Control 24:1747-1782.

[42] R. Wu, M.C. Fu (2003) *Optimal Exercise Policies and Simulation-Based Valuation for American-Asian Options.* Operations Research 51(1):52-66.

[43] Y. Yamamoto (2005) *Double-Exponential Fast Gauss Transform Algorithms for Pricing Discrete Lookback Options.* Publications of the Research Institute for Mathematical Sciences 41(4):989-1006.

[44] L. F. Yan (2005) *Pricing of Lookback Options.* `http://ww1.math.nus.edu.sg/ProjectArchive/Honours/archive_files/sci10007.pdf`

[45] R. Zvan, P. Forsyth, K. Vetzal. (1998). *Robust Numerical Methods for PDE Models of Asian Options.* Journal of Computational Finance 1:39-78.