TRUNCATION AND ACCELERATION OF THE TIAN TREE FOR THE PRICING OF AMERICAN PUT OPTIONS

TING CHEN
MARK JOSHI[1]

We present a new method for truncating binomial trees based on using a tolerance to control truncation errors and apply it to the Tian tree together with acceleration techniques of smoothing and Richardson extrapolation. For both the current (based on standard deviations) and the new (based on tolerance) truncation methods, we test different truncation criteria, levels and replacement values to obtain the best combination for each required level of accuracy. We also provide numerical results demonstrating that the new method can be 50% faster than previously presented methods when pricing American put options in the Black--Scholes model.

[1] Centre for Actuarial Studies, Department of Economics, University of Melbourne, VIC3010, Australia

INTRODUCTION

Valuation of European options within the Black-Scholes model is a long-solved problem with closed form formulas in terms of the cumulative normal distribution. However, the efficient and accurate valuation of American put prices in the Black-Scholes model is still a challenging problem.

Zhu (2006) was able to develop an analytical formula for the value of such an option. Whilst appealing, the formula involves two infinite sums of infinite double integrals, which may be useful for specialised needs such as arbitrary precision arithmetic, but for pricing, they take an unfeasible amount of time to evaluate compared to existing numerical methods.

The difficulties associated with obtaining analytical results have made numerical methods the natural approach to pricing American put options. Whilst advantageous for high-dimensional problems, Monte Carlo simulation is of less use for pricing American puts as it moves forward in time, making the assessment of exercise strategies difficult. In addition, for one-dimensional problems, such as the one being considered, the rate of convergence for Monte Carlo simulation is relatively slow.

However, these two major shortcomings of Monte Carlo simulations do not exist with tree pricing methods as they move backwards in time, and as a result assessing the exercise decision is trivial The binomial tree (CRR) method was introduced by Cox *et al* (1979) as a numerical method for pricing American put options. It was proven by Amin and Khanna (1994) that the CRR tree converges to the correct price. The rate of convergence was later studied by Lamberton (1998), and Leisen (1998) who showed that the CRR tree converges with order 1. Various methodologies have been proposed to accelerate the convergence and these generally focus on reducing the constant in front of the order, rather than on improving the order.

In addition to tree pricing methods, various other approximations are also competitive. Staunton (2005) benchmarked several of these approximations to price American put options, including: two analytic approximations due to Ju (1998) and Ju and Zhong (1999); the explicit finite difference method; the implicit finite difference method; and the Leisen and Reimer (1996) [C] binomial tree with and without truncation. Where applicable these techniques were tested with and without Richardson extrapolation. Considering time versus error, Staunton concluded that the Leisen—Reimer tree with Richardson extrapolation and truncation was the most effective method, in preference to other competing methods, including those based on numerical integration.

Joshi (2009) conducted further analysis by examining 11 different types of trees with 20 combinations of acceleration techniques, considering a total of 220 trees. From this comprehensive study, some of the best overall results were obtained with the Tian tree, together with truncation, smoothing and Richardson extrapolation. In particular, this new combination proved to be better than the truncated Leisen–Reimer tree with Richardson extrapolation, the best case in Staunton (2005).

However, Joshi chose not to examine the level of truncation in consideration of the speed / accuracy trade-off, but instead elected to use a conservative level of truncation to preserve accuracy. Here we address the issue of how to truncate. In particular, we look at the questions of where to truncate and what to do at truncated nodes.

Trinomial trees, such as the Tian (1993) fourth-order moment-matching tree will not be considered since Chan, Joshi, Tang and Yang (2009) have demonstrated that they are less efficient than the Tian binomial tree.

We present in this article a new method for determining where to truncate binomial trees and demonstrate numerically that this new method, when applied to the Tian tree with smoothing and Richardson extrapolation leads to substantial speed-ups for pricing American put options in the Black–Scholes model. By tuning the truncation method, level and replacement value, we find combinations of acceleration techniques more efficient than the best case in Joshi (2009), which was shown to be better than the best case in Staunton (2005). Therefore, we present the most efficient known numerical method to value American put options in the Black-Scholes model at the important accuracy levels of 10^{-3} to 10^{-4}.

We note that whilst Staunton tested against a wide range of methodologies, he did not test against the most recent approaches involving numerical integration, see Prekopa and Szantai (2010) for discussion such techniques and their relationships to binomial trees. We note, however, that these techniques appear to give different prices in the limit.

One advantage of tree methodologies is that as well as estimating the price, it is possible to read the exercise boundary off from the tree simply by observing the points of exercise. However, truncation will often result in the exercise boundary being in the area beyond the edge of the tree and so these techniques will not be beneficial to the user whose principal aim is to find the boundary.

We describe and define the Tian tree and its parameters in the **Tree implementation** section. In the **Truncation** section we describe the two different truncation methodologies considered: the common *standard deviation* method, and our new *tolerance* method. The **Acceleration techniques** section then outlines the other acceleration techniques used in conjunction with truncation followed by the **Benchmark framework** section with details of how numerical results are calculated. Finally, we present the **Numerical Results** section and conclude in **Conclusion** that we have found a method up to 50% more efficient than the best method in Joshi (2009). For completeness, some detailed tables are available in the **Appendix**.

TREE IMPLEMENTATION

The particular binomial tree implementation that we focus on is the Tian tree, also known as the Tian third-order moment-matching binomial tree proposed by Tian (1993). This tree matches the first three moments of the stock to the underlying Black-Scholes model.

Consider the following notation:
Let S_t denote the value of the underlying asset at time t,
K denote the strike of the option,
σ denote the volatility of the underlying,
r denote the risk free rate,
T denote Time to maturity or the option, and
n denote the number of steps/layers in the tree.

The Tian tree is a risk-neutral binomial tree with the following parameters:
Let $v_n = \exp(\sigma^2 \frac{T}{n})$

Then the upward move in an *n* step tree is given by:

$$u_n = \tfrac{1}{2}\exp(r\frac{T}{n})v_n(v_n + 1 + \sqrt{v_n^2 + 2v_n - 3})$$

The down move in an *n* step tree is given by:

$$d_n = \tfrac{1}{2}\exp(r\frac{T}{n})v_n(v_n + 1 - \sqrt{v_n^2 + 2v_n - 3})$$

Since the tree is risk neutral, the probability of an upward move is given by:

$$p_n = \frac{\exp(r\frac{T}{n}) - d_n}{u_n - d_n}$$

TRUNCATION

Truncation is based on the idea that a binomial tree should be pruned so that time is not wasted on computing nodes that are too far from the area of interest. Two classes of truncation methodologies will be examined.

The first follows Andicropoulos, Widdicks, Duck and Newton (2004). This method selects nodes in the tree to truncate based on standard deviation in log space from the present value of the strike, or standard deviations from the future value of the current stock price, or both. The rationale being that if the number of standard deviations is large enough, the value of the node makes little difference to the value of the option. We will refer to this method as the *standard deviation* method.

The second method looks at the option values in the nodes less their intrinsic value; this represents the time value component of the option value. A decision is made to truncate the remaining nodes in the layer once the time value is less than a tolerance level. This method relies on the property of American put options where the time value is greatest around the strike and is monotonically decreasing when deeply in or out of the money. As far as we are aware, this is a completely new truncation technique. We will refer to this method as the *tolerance* method.

All truncations methods have the feature that for a given set of truncation parameters there is a maximum level of accuracy that can be achieved. Thus once we have chosen our tolerance level, it is unlikely that we will achieve price substantially more accurate than the chosen fixed tolerance level. However, one could make the truncation tolerance a function of the number of steps, for example, setting it to a constant divided by the number of steps. This will then lead to a convergent method. We do not explore this idea here, since our focus is principally on finding a fast pricer at the levels of accuracy commonly used.

The replacement value

Both truncation methodologies require a rule for determining the replacement value for truncated nodes. If a parent node requires a child node that has been truncated, then the child node's value must be estimated.

We use four candidates for replacement values:

(1) ***Intrinsic*: Replace with the intrinsic value**. The benefit of this method is that it is very fast since evaluation of the intrinsic value is simple. The shortcoming is that it does not take into account the time value of holding onto the option to exercise in the future. As such, this method will understate the value of the option.

(2) ***Black-Scholes*: Replace with the max of the intrinsic value and the Black-Scholes value of the corresponding European option.** The benefit is that this should be more accurate as it takes into account the time value as a European option yet still allows for exercise immediately. However, it still does not consider the value of early exercise in the future. For example, at t = 5/200, early exercise at t=5/200 is considered, but not the value of any possible early exercise at t=6/200 to 199/200 (when T=1). This method would also understate the value of the option.

(3) ***Adjacent value*****: Replace with the max of intrinsic and the value of the adjacent node.** This technique is fast and aims to address the shortcomings of (1) and (2): for American puts, the benefit of early exercise builds slowly as we propagate through the tree. However, neither the intrinsic nor the European option value takes this into account, so when using (1) or (2) in the middle of the tree, we lose some benefits of early exercise. By utilising the value of the adjacent node, we replicate the propagated values. An issue could be the lack of adjustment for the change in intrinsic value from the adjacent node.

(4) ***Adjacent time value*****: Replace with the sum of the intrinsic value and the time value from the adjacent node.** This technique aims to improve (3) by adjusting for the changed intrinsic value while still preserving the time value built up through the layers. However, a limitation of this method is the lack of reduction in time value when deeply in or out of the money.

The *standard deviation* truncation method

Nodes are truncated if they are too far, in terms of standard deviations in log space, from the spot or the strike. Three ways to set the truncation boundary are defined in Andicropoulos, Widdicks, Duck and Newton (2004): ξ standard deviations to the future value of the spot $(S_0 e^{rt})$ or the present value of the strike $(Ke^{-r(T-t)})$ or both.

When truncation is based on *strike*, the upper and lower bounds are given by:
Upper bound: $S_{\max} = Ke^{-r(T-t)+\xi\sigma\sqrt{T-t}}$
Lower bound: $S_{\min} = Ke^{-r(T-t)-\xi\sigma\sqrt{T-t}}$

When truncation is based on the *spot*, the upper and lower bounds are given by:
Upper bound: $S_{\max} = S_0 e^{rt+\xi\sigma\sqrt{t}}$
Lower bound: $S_{\min} = S_0 e^{rt-\xi\sigma\sqrt{t}}$

Note that the truncation around *Spot* follows Andicropoulos, Widdicks, Duck and Newton (2004) where the future value has drift r in log space.[2]

When truncation is based on *both* strike and spot, the upper and lower bounds are:
Upper bound: $S_{\max} = \min(S_0 e^{rt+\xi\sigma\sqrt{t}}, Ke^{-r(T-t)+\xi\sigma\sqrt{T-t}})$
Lower bound: $S_{\min} = \max(S_0 e^{rt-\xi\sigma\sqrt{t}}, Ke^{-r(T-t)-\xi\sigma\sqrt{T-t}})$

[2] This is slightly different to Staunton (2005) where the mean of the stock in log space (drift $r - \frac{1}{2}\sigma^2$) is used. Both methods were tested and found to be very similar.

The *tolerance* truncation method

Truncations under this method occur based on whether the time value is less than a specified tolerance. For each layer, start at K and move up the layer until the time value is less than ξ and then start at $Ke^{-r(T-t)}$ and repeat the process down the layer.

On average, with a tolerance set to ξ, the final value given by the truncated tree should be roughly within ξ of the value given by the un-truncated tree. The reason is that suppose a whole layer is truncated, each node would contribute an error of no more than ξ, and the sum of the probabilities of obtaining all the nodes in a layer is 1. Extending this idea to the general case when truncations do not all occur in the same layer, the final option value is the probability-weighted sum of all boundary nodes, that is the final nodes and the nodes at the truncation boundary. There is no truncation error for final nodes, and the error in each of the truncated nodes should be roughly within ξ from the un-truncated tree, hence the probability-weighted sum should have an error generally within ξ of the un-truncated tree.

Of course, the error for the truncated tree could deviate outside ξ, since the truncation introduces errors in child nodes and hence affects the truncation decisions of parent nodes. As an example, consider a hypothetical tree where most nodes have time value 1.01ξ. Truncation can have a cascading effect reducing the time value on each parent node slightly, bringing time value to under ξ and hence further truncation, potentially introducing an error greater than ξ. A theoretical bound for the error introduced is $n\xi$, however this would be extremely rare and should not be a problem for practical applications.

Abbreviations

The following abbreviations may be used for results:
Trunc refers to the truncation criteria (see below).
Param refers to the truncation parameter, that is, the number of standard deviations or the tolerance.
Repl refers to the replacement value (see below).

For the truncation criteria:
K – Truncate with strike using the *standard deviation* method.
S_0 – Truncate with spot using the *standard deviation* method.
Both – Truncate with the both using the *standard deviation* method.
Tol – Truncate with the *tolerance* method.

For replacement values:
Intrinsic – The *Intrinsic* replacement value.
BS – The *Black-Scholes* replacement value.
Adj – The *Adjacent value* replacement value.
AdjTm – The *Adjacent time value* replacement value.

ACCELERATION TECHNIQUES

This section provides a brief outline of the acceleration techniques that are applied. The combination of smoothing and Richardson extrapolation as an acceleration technique for binomial trees was introduced by Broadie and Detemple (1996). We briefly review their methodology.

Richardson extrapolation

Richardson extrapolation is a useful numerical technique that aims to eliminate the first order error term to improve the error behaviour of numerical calculations.

Suppose the estimated price after *n* steps is

$$p_n = TruePrice + \tfrac{E}{n} + o(\tfrac{1}{n}),$$

where E is a constant. (This is not generally true for American put options in binomial trees but it is true enough for the technique to be useful.)

The Richardson extrapolated value is constructed as a weighted sum of the form

$$p_n^{RE} = TruePrice + o(\tfrac{1}{n}) \text{ and } p_n^{RE} = wp_n + (1-w)p_{\lfloor \frac{n}{2} \rfloor}.$$

We wish to cancel the error E, hence:

$$0 = w\frac{E}{n} + (1-w)\frac{E}{\lfloor \frac{n}{2} \rfloor}$$

$$\Leftrightarrow w = (1 - \frac{\lfloor \frac{n}{2} \rfloor}{n})^{-1} = \frac{n}{n - \lfloor \frac{n}{2} \rfloor} = \frac{n}{\lceil \frac{n}{2} \rceil} = \begin{cases} 2 & , n \text{ even} \\ \frac{2n}{n+1}, & n \; odd \end{cases}$$

When *n* is even, *w* = 2, and the extrapolated value is conveniently $p_n^{RE} = 2p_n - p_{\frac{n}{2}}$

Smoothing

Smoothing consists of replacing the values on the second last layer of the tree with values for European puts as calculated by the Black-Scholes formula. As there is no exercise opportunity between the second last and final step, for nodes in layer *n*-1, we effectively value a series of European puts with a single-step tree. The Black--Scholes value is much more accurate than a single-step tree and will vary smoothly for different nodes in the layer.

In addition, smoothing interacts well with Richardson extrapolation, while only increasing the calculation time marginally. This is in line with findings in Joshi (2009). The reason is that smoothing reduces the oscillations and noise in the error of the tree and this improves the performance of Richardson extrapolation.

For example, when truncating by *Strike* at *4* standard deviations with *Black-Scholes* as replacement value and *800* steps, we obtain:

(RMS error, time)	**RE = no**	**RE = yes**
Smooth = no	(4.06E-03, 0.00577)	(7.20E-03, 0.00803)
Smooth = yes	(3.13E-03, 0.00575)	(7.84E-05, 0.00805)

The pattern above repeats for other step numbers and truncation methods. We see that the two techniques individually achieve little but in combination are powerful.

Joshi (2009) showed that the control variate technique due to Hull and White (1988) has been superseded. It is no longer an effective methodology when Richardson extrapolation and smoothing are used.

BENCHMARK FRAMEWORK

Error measures

Three error measures are used as in Joshi (2009).

Let O(i) = Observed tree value. A(i) = Actual value

RMS: Root-mean-squared absolute error

$$\sqrt{\frac{\sum_{i=1}^{n}(O(i)-A(i))^2}{n}}$$

B&D: Root-mean-squared modified relative error due to Broadie and Detemple (1996):

$$\sqrt{\frac{\sum_{i\in M}\left(\frac{O(i)-A(i)}{A(i)}\right)^2}{\|M\|}}, \quad M=\{k : A(k)\geq 0.5\}$$

where $\|M\|$ is the number of elements in the set M, as not all observations are included. (In particular those with actual value less than 0.5 are excluded.) This is done to avoid distortions due to a small error on a small value.

Joshi: Root-mean-squared modified relative error due to Joshi (2009):
(for American put options)

$$\sqrt{\frac{\sum_{i=1}^{n}\left(\frac{O(i)-A(i)}{0.5+A(i)-Max(K-S_0,0)}\right)^2}{n}}$$

For brevity only errors in the Joshi measure are shown in the Numerical Results section. Results for other error measures are shown in the Appendix and are consistent with those in the Joshi measure.

Option parameters

Our methods are independent of option parameters and to demonstrate we will follow frameworks established by previous authors. Option parameters for the American puts are picked from random distributions as outlined in Joshi (2009), which is the exact distribution as used in Leisen (1998) which was based on Broadie and Detemple (1996). This distribution is "a reasonable reflection of options that are of interest to academics and practitioners" (Broadie and Detemple, 1996). A summary of the distribution is as follows:

Let:
the strike be set at 100;
the volatility vary uniformly from 0.1 to 0.6;
the time to maturity be with ¾ probability: uniform between 0.1 and 1,
and with ¼ probability uniform between 1 to 5;
the spot be uniform between 70 and 130;
the risk free rate be zero with 0.2 probability, and otherwise uniform from 0 to 0.10.

We give a pseudo-code description:

```
Let $U_i$ = uniform [0, 1)
K = 100, $\sigma$ = 0.1+ 0.5 $U_1$
T = IF $U_2$ < 0.75 THEN 0.1 + 0.9 $U_3$ ELSE 1 + 4 $U_3$
S0 = 70 + 60 $U_4$
r = IF 0.10 – 0.125 $U_5$ < 0 THEN 0 ELSE 0.10 – 0.125 $U_5$
```

Steps

For the purpose of evaluating truncation and other acceleration techniques, the steps of $n \in \{10, 25, 50, 75, 100, 200, 400, 800, 1200, 1600\}$ are used to obtain samples of error and time for the different truncation techniques.

The un-truncated Tian tree with n=16000 together with smoothing and Richardson extrapolation is used as the true value.

Target Accuracy

We first obtain a set of results in the form of {error, time taken} for different number of steps for each combination of methods The observations are from the pricing of 2000 options randomly chosen as above. Errors are computed using the three methods as described and time is the average running time.

Piecewise linear interpolation is then applied in log-log space to find the number of steps and time required to achieve any given accuracy/error. In cases where extra steps increase error, interpolation stops as the extra steps are not justified (the truncation error dominates the tree error).

We then find the time taken under each combination of methods to achieve:

- RMS errors of {0.001, 0.0005, 0.0001},
- B&D errors of {0.0001, 0.0005, 0.00001}, and
- Joshi errors of {0.001, 0.0005, 0.0001}.

These errors are of particular interest to us as they are at reasonable accuracy levels for practical use and require a moderate number of steps.

As the average option value is 13.52 and the average time value 6.39, the error sizes under each measure are roughly comparable.

Truncation parameters

For the *standard deviation* method, after some preliminary results, we will use {1, 2, 3, 3.5, 4, 4.5, 5, 6} standards deviations as the truncation parameter.

We will see that 1-2 standard deviations is too aggressive with unacceptable errors, but that 5-6 standard deviations is unnecessarily conservative.

For the *tolerance* method, following the three accuracy levels of {0.001, 0.005, 0.0001}, the tolerance parameters of {0.01, 0.005, 0.001, 0.005, 0.0001, 0.0005, 0.00001} will be used to test the effects of different tolerance levels.

NUMERICAL RESULTS

Comparable cases

The ***un-truncated tree*** is equivalent to the ***Tian 12*** tree in Joshi (2009), i.e. the Tian tree with smoothing and Richardson extrapolation but no truncation. The ***Tian 13*** tree, one of the equal best trees in Joshi (2009) is roughly equivalent to truncation by *Strike* to *6* standard deviations with the *Black-Scholes* replacement value. These two cases will be important benchmarks for us to judge the improvements the truncation methods can bring, against no truncation (*Tian 12*) and conservative truncation (*Tian 13*). We can then see that at the target accuracy levels, our best methods allow around 50% more evaluations per second than *Tian 13*, the previously best method.

The standard deviation method

In the tables that follow, the best three cases in each targeted accuracy level will be presented along with the *Tian 12* (or un-truncated tree) and *Tian 13* cases. The columns in order are: the target accuracy, the reason the case is selected, the truncation parameter, the truncation criteria, the replacement value, the number of interpolated steps required to achieve this level of accuracy, the average time required to value one option in milliseconds, the number of options valued per second, followed by the percentage improvement over the *Tian 12* (un-truncated) case and the percentage improvement over the *Tian 13* case.

Joshi error *(Tables for the other error measures are available in the appendix.)*

Target	Type	Param	Trunc	Repl	steps	Time(ms)	PerSecond	Improve T12	Improve T13
0.001	Best	3.5	*Strike*	*Intrinsic*	226.0	1.637	610.8	41%	30%
0.001	Second	4.0	*Strike*	*Intrinsic*	224.8	1.710	584.7	35%	25%
0.001	Third	4.0	*Strike*	*Adj*	226.0	1.714	583.3	35%	25%
0.001	Tian 13	6.0	*Strike*	*BS*	224.8	2.135	468.3	8%	
0.001	Un-truncated		none	none	224.8	2.309	433.1		
0.0005	Best	3.5	*Strike*	*Intrinsic*	431.9	3.892	256.9	97%	31%
0.0005	Second	4.0	*Strike*	*Intrinsic*	418.8	3.967	252.1	93%	29%
0.0005	Third	3.5	*Strike*	*BS*	419.1	4.031	248.1	90%	27%
0.0005	Tian 13	6.0	*Strike*	*BS*	418.7	5.118	195.4	50%	
0.0005	Un-truncated		none	none	418.7	7.671	130.4		
0.0001	Best	4.0	*Strike*	*Intrinsic*	1583.7	25.074	39.9	318%	34%
0.0001	Second	4.5	*Both*	*Adj*	1576.8	25.590	39.1	310%	31%
0.0001	Third	4.0	*Strike*	*BS*	1568.4	25.757	38.8	307%	30%
0.0001	Tian 13	6.0	*Strike*	*BS*	1568.1	33.542	29.8	213%	
0.0001	Un-truncated		none	none	1568.1	104.837	9.5		

Table 1: Best cases for Joshi error under standard deviation truncation.

Hence the choice of truncation by *Strike* to *3.5* standard deviations with the *Intrinsic* replacement value is very competitive for moderate accuracies (0.001, 0.0005) while of truncation by *Strike* to *4* standard deviations with the *Intrinsic* replacement value is the best for higher accuracy.

Improvements of 41%, 97% and 318% over the un-truncated tree can be achieved at medium (0.001), moderate (0.0005) and high (0.0001) accuracies. Furthermore improvements of around 30% can be achieved over the *Tian 13* tree, one of the best trees in Joshi (2009).

Figure 1 on the next page shows the closeness of the best standard deviation methods.

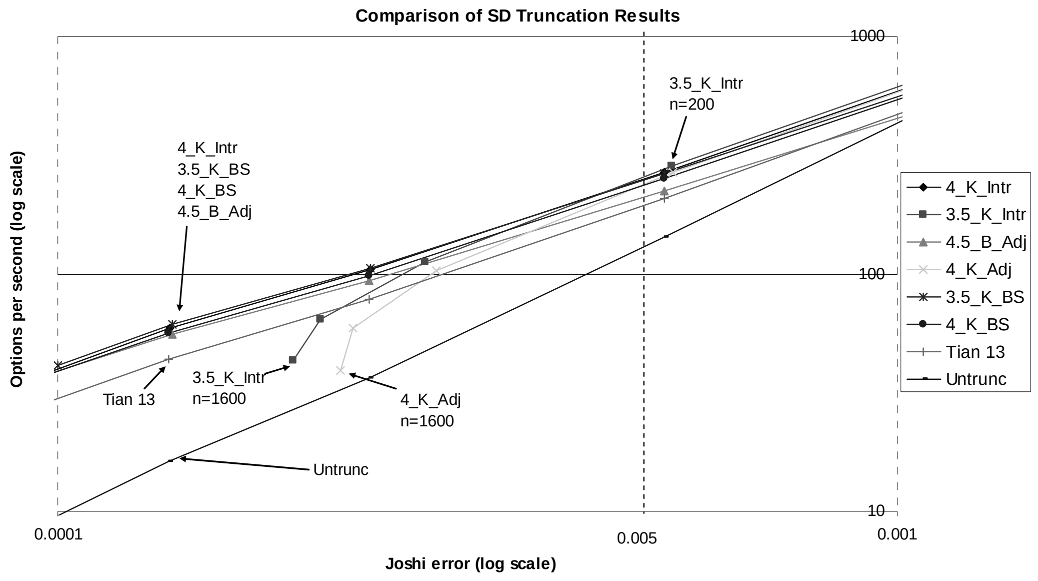

Figure 1: Comparison of Standard Deviation Truncation results.

The tolerance method

Joshi error *(Tables for other error measures are available in the appendix)*

Target	Type	Param	*Trunc*	*Repl*	steps	Time(ms)	PerSecond	Improve T12	Improve T13
0.001	Best	1E-04	*Tol*	*Intrinsic*	224.9	1.458	685.9	57%	46%
0.001	Second	5E-05	*Tol*	*Intrinsic*	224.8	1.476	677.3	55%	45%
0.001	Third	5E-04	*Tol*	*Intrinsic*	232.0	1.495	668.8	53%	43%
0.001	Tian 13	6.0	*Strike*	*BS*	224.8	2.135	468.3	7%	
0.001	Un-truncated		none	none	224.8	2.309	437.9		
0.0005	Best	1E-04	*Tol*	*Intrinsic*	421.0	3.321	301.1	131%	54%
0.0005	Second	5E-05	*Tol*	*Intrinsic*	419.1	3.359	297.7	128%	52%
0.0005	Third	1E-05	*Tol*	*Intrinsic*	418.7	3.419	292.5	124%	50%
0.0005	Tian 13	6.0	*Strike*	*BS*	418.7	5.118	195.4	50%	
0.0005	Un-truncated		none	none	418.7	7.671	130.4		
0.0001	Best	1E-05	*Tol*	*Intrinsic*	1572.9	20.873	47.9	402%	61%
0.0001	Second	5E-04	*Tol*	*BS*	1606.0	20.879	47.9	402%	61%
0.0001	Third	1E-04	*Tol*	*BS*	1568.5	21.101	47.4	397%	59%
0.0001	Tian 13	6.0	*Strike*	*BS*	1568.1	33.542	29.8	213%	
0.0001	Un-truncated		none	none	1568.1	104.837	9.5		

Table 2: Best cases for Joshi error under tolerance truncation.

Improvements of 57%, 131% and 402% over the un-truncated tree can be achieved at medium (0.001), moderate (0.0005) and high (0.0001) accuracies. This dominates the improvements from the *standard deviation* truncation method. Furthermore, the results are more efficient than the *Tian 13* tree, the best tree in Joshi (2009), at a rate of around 50% more evaluations per second.

Figure 2 and Figure 3 on the next page illustrates the performance of the *tolerance* method. It is a powerful method provided that the required tolerance is known, which is perfect for numerical applications with a target accuracy.

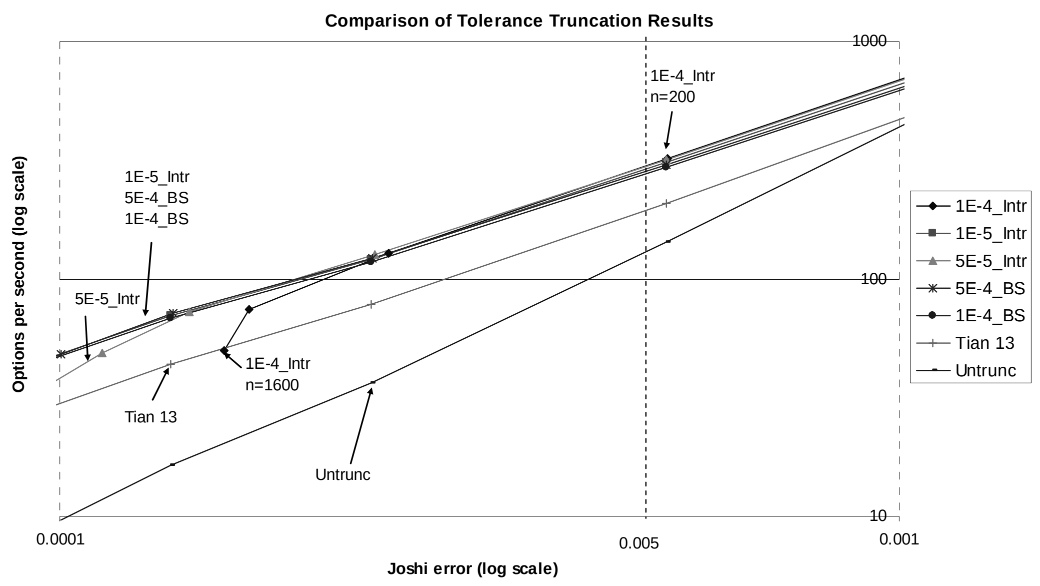

Figure 2: Comparison of Tolerance Truncation results.

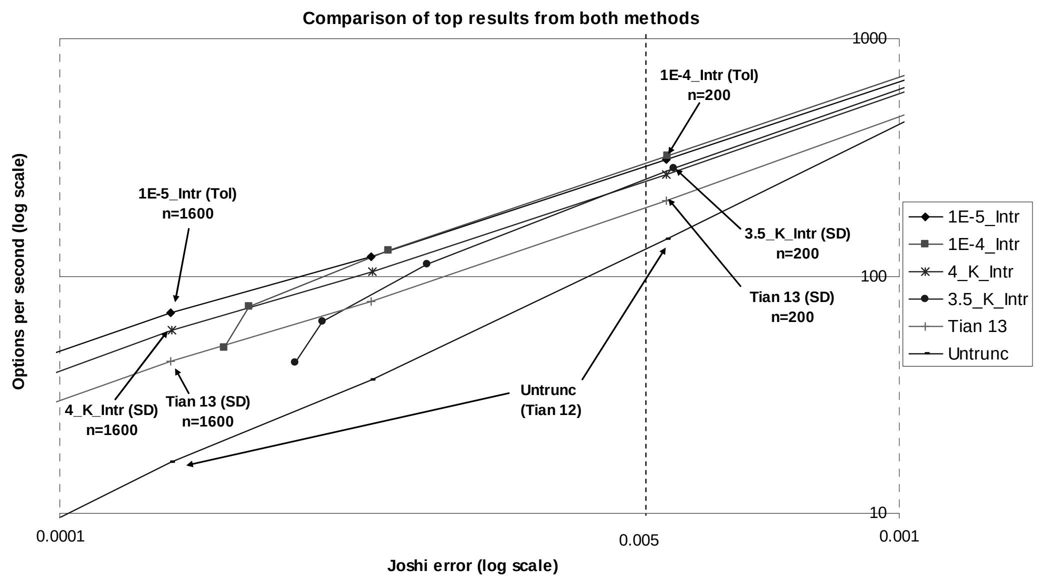

Figure 3: Comparison of the best results from both methods

Error behaviour

Truncation error

We define *truncation error* to be the additional error of the truncated tree compared to the non-truncated tree. By this definition, if the truncated tree has a smaller error than the un-truncated tree then there is no *truncation error*. This can happen when the replacement value is more accurate than the truncated node, e.g. the *Black-Scholes* replacement value on an aggressively truncated *10*-step tree.

In *Table 3* below, the truncation error is shown for best cases under Joshi error. "Closest available steps" is the closest matching step number that is actually ran in the benchmark. Aggregate statistics of mean, standard deviation and maximum are shown for the truncation error as defined above.

Target accuracy	Param	Trunc	Repl	Interp Steps	Closest available steps	Options Per Second	Truncation error		
							Mean	StDev	Max
1E-03	1E-04	*Tol*	*Intrinsic*	224.9	200	800.0	3.2E-06	1.1E-05	1.1E-04
1E-03	3.50	*Strike*	*Intrinsic*	226.0	200	718.9	2.0E-05	4.5E-05	6.8E-04
5E-04	1E-04	*Tol*	*Intrinsic*	421.0	400	322.4	8.2E-06	2.2E-05	1.8E-04
5E-04	3.50	*Strike*	*Intrinsic*	431.9	400	285.1	4.5E-05	7.7E-05	9.3E-04
1E-04	1E-05	*Tol*	*Intrinsic*	1572.9	1600	46.8	1.9E-06	4.5E-06	3.5E-05
1E-04	4.00	*Strike*	*Intrinsic*	1583.7	1600	39.3	7.5E-06	1.2E-05	1.0E-04

Table 3: Truncation error statistics for the best cases under Joshi error

Hence not only is the best case with the *tolerance* truncation method faster, it is superior in all three statistics of truncation error.

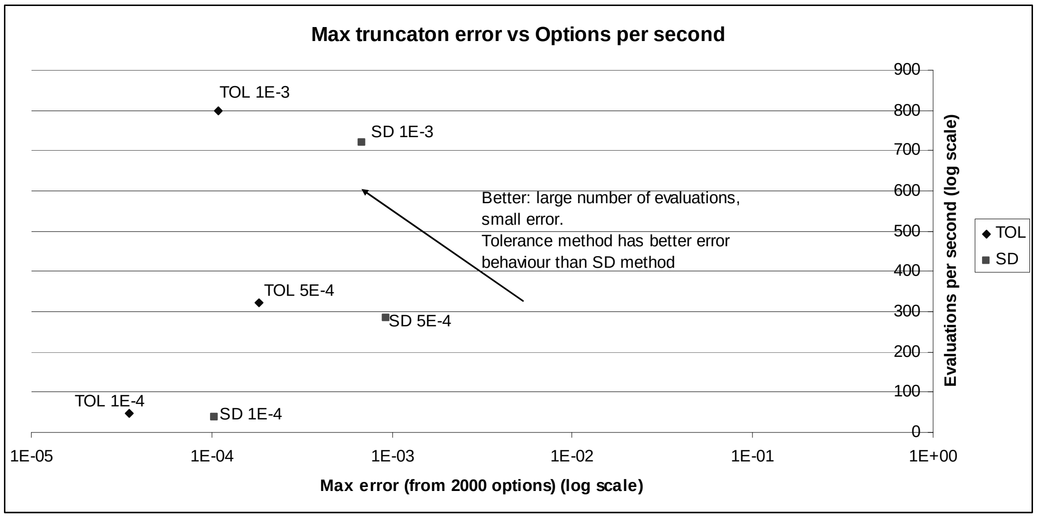

Figure 4: Max truncaton error vs Options per second.

Error and convergence

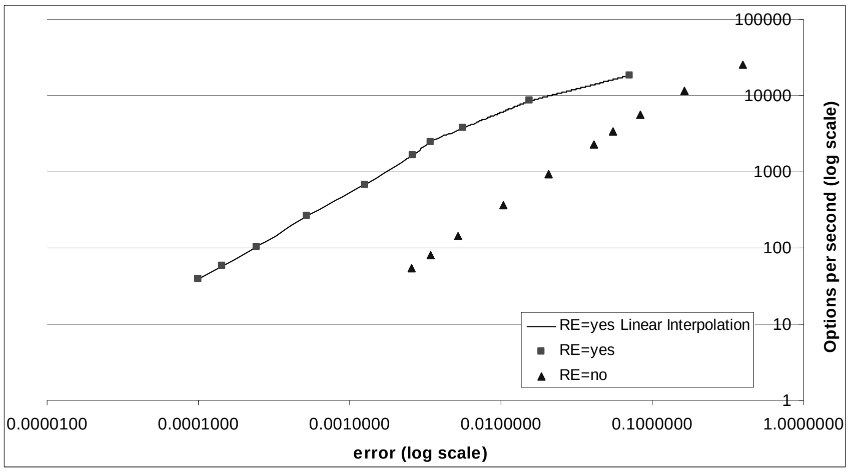

Figure 5: The effect of Richardson extrapolation on error – with convergence.

Figure 5 shows the RMS error versus time in log-log scale for truncation by *strike* to 4 standard deviations, with the *Intrinsic* replacement value and the smoothing acceleration technique. The series denoted by squares are option values with Richardson extrapolation while the triangles are option values without performing Richardson extrapolation. The line is the piecewise linear interpolation of the option values which had Richardson extrapolation applied. Clearly, Richardson extrapolation helps, providing 10 times better error for the same running time, or about 10 times faster for the same error.

We will also consider the case when asymptotic convergence does not occur, i.e. when the lead error term is no longer $\frac{E}{n}$. Richardson extrapolation is still worthwhile if you consider the figure below:

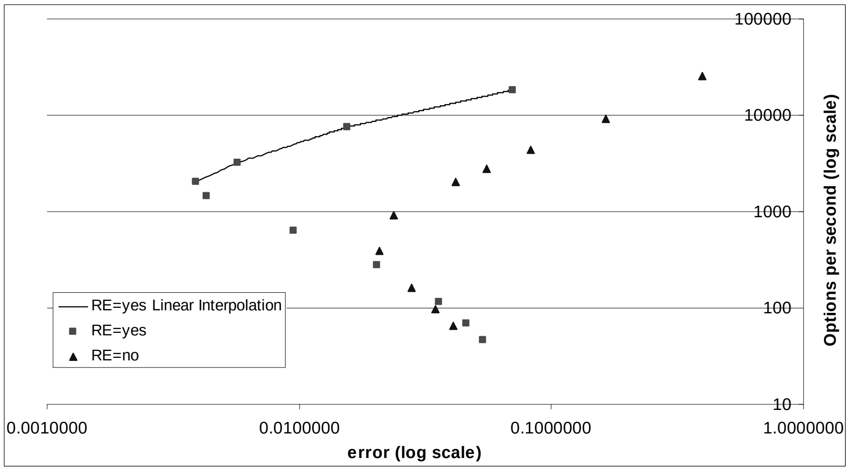

Figure 6: Richardson extrapolation when not converging asymptotically

Figure 6 shows the RMS error versus time in log-log scale for truncation by *Spot* to *3* standard deviations, with the *Black-Scholes* replacement value and smoothing. The values do not asymptotically converge as truncation is too aggressive, increasing the number of steps causes more error to be introduced. Only values in the upper half of the curve are of interest as we want more evaluations per second for the same error. We can see from *Figure 6* that even when errors are not convergent, Richardson extrapolation still can improve error for the same speed or allow faster evaluations at the same error.

Hence Richardson extrapolation is beneficial for both cases (with and without convergence) as we are able to achieve smaller errors and faster time.

It is not surprising that under aggressive truncation, the errors do not asymptotically tend towards zero as the number of steps increase, as truncation can introduce a bias in the tree that causes it not to converge. Increasing the number of steps can even exacerbate the truncation error.

Take the *tolerance* method for example with a tolerance of 10^{-4}, we would not expect convergence as the errors cannot improve beyond 10^{-4}. In fact, the reader may remember that a theoretical error bound for the *tolerance* method is $n\xi$. Hence needlessly increasing *n* actually increases this theoretical bound. Empirical results also support that once the desired accuracy of approximately ξ is reached, increasing *n* further can actually increase errors.

Hence a limitation of using the *tolerance* method is the need to come up with a suitable step size given a target accuracy and tolerance. Through our data it would seem 200/400/1600 steps is about right for accuracy levels of 0.001/0.0005/0.0001. We will not seek to generalise the optimal step number here but leave the problem for future work.

Extension of the tolerance parameter to encourage convergence

Various extensions can also be made to how the tolerance can be specified. One simple extension is to set tolerance to decrease as *n* increases, i.e. use

tol = k/n where *k* is a constant and *n* is the step count.

This would enable the *tolerance* method to converge as n increases. This will not be explored as we are primarily concerned about the *tolerance* method as an intelligent way to truncate to achieve maximum speed at the target accuracy of 10^{-3} to $10^{-4.}$ Hence fixed tolerance parameters are more suitable for this purpose. While the convergence property is desirable, there are many ways to specify tolerance as a function of step count and thus this is a complex topic that deserves further exploration on its own. Furthermore, the *tolerance* method is demonstrating great performance improvements at high accuracy levels, which could be explored with convergence to high precision in a future analysis.

Regardless of how the tolerance parameter is specified, one major benefit of the *tolerance* method is the greater control on the error of truncation, as the method intuitively attempts to limit truncation errors to some specified level close to the un-truncated tree.

CONCLUSION

The best numerical results are achieved with the following combination:

- The *tolerance* truncation method
  - with tolerance *parameter* of about one fifth of the targeted accuracy
  - with the *Intrinsic* replacement value
  - 200/400/1600 steps for accuracy levels of 0.001/0.0005/0.0001.
- in combination with the acceleration techniques of
  - smoothing and
  - Richardson extrapolation.

We were able to achieve results 50% faster than the previous known best tree.

One major benefit of the *tolerance* method is the greater control on the error of truncation, as the method intuitively attempts to limit truncation errors to some specified level close to the un-truncated tree. Errors are better controlled: branches are pruned aggressively when time value is small and hence replacement values are good approximations (deeply in or out of the money).

Improvements of up to 350% can be achieved with the *tolerance* truncation when compared to the un-truncated tree with smoothing and Richardson extrapolation, or *Tian 12*. In addition, an improvement of 50% can be gained over *Tian 13*.

If the *standard deviation* method must be used, then in combination with smoothing and Richardson extrapolation, truncation by *Strike* should be used with a parameter of 3.5 to 4.0 standard deviations and *Intrinsic* replacement value. This is also an improvement over Tian 12 and 13, but is smaller than those from the *tolerance* method.

Staunton (2005) found that LR trees with smoothing and truncation was the best numerical method, while Joshi (2009) found that the *Tian 13* tree was superior to the LR tree. We have further extended that and achieved an improvement of 50% on top of the *Tian 13*. Hence, the combination of acceleration techniques of *tolerance* truncation with Richardson extrapolation and smoothing is the most efficient known numerical technique for valuing American options at the accuracy levels considered.

APPENDIX

The standard deviation method – timing results

Trunc:	K				S_0				Both			
StDevs:	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time
6.0	8.10E-5	7.35E-6	9.77E-5	0.0335	8.10E-5	7.35E-6	9.77E-5	0.0372	8.10E-5	7.35E-6	9.77E-5	0.0315
5.0	8.10E-5	7.34E-6	9.77E-5	0.0296	8.02E-5	6.99E-6	9.79E-5	0.0323	8.01E-5	6.99E-6	9.79E-5	0.0280
4.5	8.07E-5	7.14E-6	9.77E-5	0.0276	1.20E-4	3.42E-5	1.14E-4	0.0298	1.21E-4	3.47E-5	1.14E-4	0.0261
4.0	7.93E-5	8.66E-6	9.89E-5	0.0254	1.37E-3	3.88E-4	5.73E-4	0.0274	1.39E-3	3.95E-4	5.82E-4	0.0243
3.5	2.31E-4	8.64E-5	1.91E-4	0.0233	1.33E-2	3.07E-3	4.09E-3	0.0248	1.35E-2	3.15E-3	4.18E-3	0.0224
3.0	2.05E-3	7.56E-4	1.37E-3	0.0211	9.42E-2	1.75E-2	2.19E-2	0.0224	9.58E-2	1.81E-2	2.25E-2	0.0206
2.0	8.44E-2	3.12E-2	3.79E-2	0.0166	1.82E+0	2.26E-1	2.49E-1	0.0173	1.86E+0	2.43E-1	2.61E-1	0.0167
1.0	1.74E+0	3.55E-1	3.18E-1	0.0119	8.47E+0	7.17E-1	7.67E-1	0.0122	8.50E+0	7.19E-1	7.70E-1	0.0129

*Table 4: Standard deviation, accuracy and time with **Intrinsic** as replacement value and n=1600.*

Trunc:	K				S_0				Both			
StDevs:	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time
6.0	8.10E-5	7.35E-6	9.77E-5	0.0345	8.10E-5	7.35E-6	9.77E-5	0.0383	8.10E-5	7.35E-6	9.77E-5	0.0326
5.0	8.10E-5	7.35E-6	9.77E-5	0.0306	8.08E-5	7.35E-6	9.77E-5	0.0333	8.08E-5	7.35E-6	9.77E-5	0.0290
4.5	8.10E-5	7.38E-6	9.77E-5	0.0286	7.84E-5	7.38E-6	9.77E-5	0.0308	7.84E-5	7.38E-6	9.77E-5	0.0271
4.0	8.15E-5	7.56E-6	9.78E-5	0.0265	1.41E-4	7.56E-6	9.78E-5	0.0284	1.40E-4	7.56E-6	9.78E-5	0.0253
3.5	8.55E-5	9.02E-6	1.00E-4	0.0243	1.41E-3	9.02E-6	1.00E-4	0.0259	1.40E-3	9.02E-6	1.00E-4	0.0235
3.0	2.08E-4	4.84E-5	3.42E-4	0.0222	1.03E-2	4.84E-5	3.42E-4	0.0234	1.03E-2	4.84E-5	3.42E-4	0.0216
2.0	9.39E-3	7.19E-3	7.45E-3	0.0178	2.01E-1	7.19E-3	7.45E-3	0.0183	2.02E-1	7.19E-3	7.45E-3	0.0178
1.0	2.05E-1	3.68E-2	6.25E-2	0.0131	9.43E-1	3.68E-2	6.25E-2	0.0133	9.47E-1	3.68E-2	6.25E-2	0.0138

*Table 5: Standard deviation, accuracy and time with **BS** as replacement value and n=1600.*

Trunc:	K				S_0				Both			
StDevs:	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time
6.0	8.10E-5	7.35E-6	9.77E-5	0.0335	8.10E-5	7.35E-6	9.77E-5	0.0372	8.10E-5	7.35E-6	9.77E-5	0.0315
5.0	8.12E-5	7.57E-6	9.78E-5	0.0296	8.09E-5	7.57E-6	9.78E-5	0.0323	8.11E-5	7.57E-6	9.78E-5	0.0279
4.5	8.65E-5	1.33E-5	9.95E-5	0.0275	8.30E-5	1.33E-5	9.95E-5	0.0298	8.48E-5	1.33E-5	9.95E-5	0.0261
4.0	2.48E-4	1.04E-4	2.17E-4	0.0255	2.92E-4	1.04E-4	2.17E-4	0.0273	2.25E-4	1.04E-4	2.17E-4	0.0242
3.5	2.02E-3	9.07E-4	1.63E-3	0.0233	2.44E-3	9.07E-4	1.63E-3	0.0248	1.66E-3	9.07E-4	1.63E-3	0.0224
3.0	1.40E-2	6.17E-3	1.08E-2	0.0212	1.55E-2	6.17E-3	1.08E-2	0.0223	1.01E-2	6.17E-3	1.08E-2	0.0205
2.0	3.14E-1	1.37E-1	2.22E-1	0.0166	2.87E-1	1.37E-1	2.22E-1	0.0173	1.77E-1	1.37E-1	2.22E-1	0.0168
1.0	2.66E+0	1.20E+0	1.54E+0	0.0120	2.23E+0	1.20E+0	1.54E+0	0.0122	9.43E-1	1.20E+0	1.54E+0	0.0129

*Table 6: Standard deviation, accuracy and time with **Adj** as replacement value and n=1600.*

Trunc:	K				S_0				Both			
StDevs:	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time
6.0	8.10E-5	7.35E-6	9.77E-5	0.0335	8.10E-5	7.35E-6	9.77E-5	0.0373	8.10E-5	7.35E-6	9.77E-5	0.0316
5.0	8.12E-5	7.57E-6	9.78E-5	0.0296	8.12E-5	7.57E-6	9.78E-5	0.0324	8.14E-5	7.57E-6	9.78E-5	0.0279
4.5	8.71E-5	1.33E-5	9.97E-5	0.0276	8.61E-5	1.33E-5	9.97E-5	0.0298	9.45E-5	1.33E-5	9.97E-5	0.0262
4.0	2.83E-4	1.05E-4	2.32E-4	0.0255	2.44E-4	1.05E-4	2.32E-4	0.0273	4.65E-4	1.05E-4	2.32E-4	0.0243
3.5	2.41E-3	9.14E-4	1.78E-3	0.0233	2.11E-3	9.14E-4	1.78E-3	0.0249	4.26E-3	9.14E-4	1.78E-3	0.0225
3.0	1.71E-2	6.22E-3	1.18E-2	0.0211	1.55E-2	6.22E-3	1.18E-2	0.0224	3.08E-2	6.22E-3	1.18E-2	0.0206
2.0	4.28E-1	1.40E-1	2.49E-1	0.0167	3.72E-1	1.40E-1	2.49E-1	0.0173	7.30E-1	1.40E-1	2.49E-1	0.0168
1.0	5.50E+0	1.39E+0	2.26E+0	0.0120	4.00E+0	1.39E+0	2.26E+0	0.0122	7.96E+0	1.39E+0	2.26E+0	0.0129

*Table 7: Standard deviation, accuracy and time with **AdjTm** as replacement value and n=1600.*

*Tables 4-7*only represent a small subset of the results table compiled, which is too large to be included. However for illustration purposes, the n=1600 case gives an indication to the level of accuracy that can be achieved with the combination of acceleration techniques.

The number of standard deviations.

It can be observed that for 5-6 standard deviations, the error is virtually identical to that of the un-truncated tree. The time taken is about a third of the un-truncated tree. This is consistent with Joshi (2009).

At 4.5 standard deviations, the error starts to change but is largely the same. The time taken is reduced to 27% of the un-truncated tree. When truncating to 3-4 standard deviations, the errors are greater but remain reasonable. Time taken is reduced to 20%. When too aggresive at 1-2 standard deviations, the errors become significant. Time taken is reduced to as low as 10%, but there is little point as the values are useless.

It would seem that 3-5 standard deviations are good candidates for truncation, depending on the level of accuracy required.

The truncation criteria

Truncation by *Strike* generally performs better than truncation by *Spot*, either being faster or having a smaller error. Therefore, it must be more effective to truncate from the end of the tree rather than from the root, although performance is largely similar.

Generally, truncation by *Strike* outperforms truncation by *Both*. When an accurate replacement value such as the *Black-Scholes* value is utilised, the error when truncating by *Spot* is comparable with truncating by *Strike*, but time taken is greater, due to the Black-Scholes value being a more time consuming but accurate replacement value. When a fast replacement value such as the *Intrinsic* is used, the error for truncation by *Spot* is greater than truncation by *Strike*, but taking a similar amount of time. This suggests that truncation by *Spot* may be truncating nodes that cannot be easily replaced or not worth truncating.

The replacement values

Adjacent and *Adjacent time value* were designed to be improvements over *Intrinsic*. They do perform well when truncating by *Spot* or *Both*, but perform poorly when truncating by *Strike*, suggesting there is interaction between truncation criteria and replacement value.

Surprisingly *Adjacent value* performs generally better than *Adjacent time-value*. One possible explanation is that the rapid disappearance of time-value when away from the money means that using the time value is ineffective. This motivated the *tolerance* truncation method.

Replacing with the *Black-Scholes* value is only marginally slower but more reliable. This could be the method of choice if reliability is a consideration.

The standard deviation method – Target accuracy

RMS error

Target	Type	Param	*Trunc*	*Repl*	steps	Time(ms)	PerSecond	Improve T12	Improve T13
0.001	Best	3.5	*Strike*	*Intrinsic*	216.1	1.542	648.5	39%	31%
0.001	Second	4.0	*Strike*	*Intrinsic*	216.5	1.626	615.0	32%	25%
0.001	Third	4.0	*Strike*	*Adj*	218.1	1.634	612.0	32%	24%
0.001	Tian 13	6.0	*Strike*	*BS*	216.6	2.027	493.4	6%	
0.001	Un-truncated		none	none	216.6	2.150	465.1		
0.0005	Best	3.5	*Strike*	*Intrinsic*	376.3	3.233	309.3	92%	36%
0.0005	Second	4.0	*Strike*	*Intrinsic*	375.6	3.422	292.2	82%	29%
0.0005	Third	3.5	*Strike*	*BS*	379.4	3.537	282.7	76%	24%
0.0005	Tian 13	6.0	*Strike*	*BS*	376.1	4.399	227.3	41%	
0.0005	Un-truncated		none	none	376.1	6.223	160.7		
0.0001	Best	4.0	*Strike*	*Intrinsic*	1312.1	19.255	51.9	297%	38%
0.0001	Second	3.5	*Strike*	*BS*	1394.7	20.126	49.7	280%	32%
0.0001	Third	4.0	*Strike*	*BS*	1344.5	20.781	48.1	268%	28%
0.0001	Tian 13	6.0	*Strike*	*BS*	1336.9	26.642	37.5	187%	
0.0001	Un-truncated		none	none	1336.9	76.402	13.1		

Table 8: Best cases for RMS error under standard deviation truncation.

B&D error

Target	Type	Param	*Trunc*	*Repl*	steps	Time(ms)	PerSecond	Improve T12	Improve T13
0.0001	Best	3.5	*Strike*	*Intrinsic*	184.9	1.257	795.6	31%	33%
0.0001	Second	4.0	*Strike*	*Intrinsic*	187.5	1.341	745.7	23%	24%
0.0001	Third	4.5	*Strike*	*AdjTm*	188.3	1.394	717.3	18%	19%
0.0001	Tian 13	6.0	*Strike*	*BS*	188.0	1.666	600.3	-1%	
0.0001	Un-truncated		none	none	188.0	1.646	607.6		
0.00005	Best	4.0	*Strike*	*Intrinsic*	351.0	3.123	320.2	81%	31%
0.00005	Second	4.5	*Strike*	*Intrinsic*	357.5	3.354	298.2	69%	22%
0.00005	Third	4.5	*Strike*	*AdjTm*	364.4	3.466	288.5	63%	18%
0.00005	Tian 13	6.0	*Strike*	*BS*	357.9	4.104	243.7	38%	
0.00005	Un-truncated		none	none	357.9	5.658	176.8		
0.00001	Best	4.0	*Strike*	*Intrinsic*	1235.1	17.687	56.5	284%	38%
0.00001	Second	4.5	*Both*	*BS*	1164.2	18.048	55.4	277%	36%
0.00001	Third	4.5	*Strike*	*Intrinsic*	1240.7	19.143	52.2	255%	28%
0.00001	Tian 13	6.0	*Strike*	*BS*	1260.5	24.476	40.9	178%	
0.00001	Un-truncated		none	none	1260.5	67.996	14.7		

Table 9: Best cases for B&D error under standard deviation truncation.

The table for the Joshi error is available in the Numerical Results section as *Table 1*.

The tolerance method – timing results

Repl:	Intrinsic				BS			
Tol:	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time
1E-5	8.07E-5	6.87E-6	9.81E-5	0.0214	8.11E-5	7.44E-6	9.77E-5	0.0229
5E-5	8.65E-5	1.63E-5	1.12E-4	0.0204	8.15E-5	7.66E-6	9.78E-5	0.0221
1E-4	1.12E-4	3.68E-5	1.57E-4	0.0200	8.19E-5	7.88E-6	9.78E-5	0.0217
5E-4	5.35E-4	2.53E-4	7.51E-4	0.0190	8.60E-5	9.31E-6	1.00E-4	0.0208
1E-3	1.20E-3	5.83E-4	1.64E-3	0.0185	1.02E-4	1.12E-5	1.33E-4	0.0203
5E-3	8.08E-3	4.21E-3	1.00E-2	0.0171	7.68E-4	7.94E-5	9.56E-4	0.0191
1E-2	1.87E-2	1.02E-2	2.16E-2	0.0164	2.58E-3	2.54E-4	3.13E-3	0.0185
Repl:	Adj				AdjTm			
Tol:	RMS	B&D	Joshi	time	RMS	B&D	Joshi	time
1E-5	8.21E-5	8.74E-6	9.80E-5	0.0256	8.21E-5	8.77E-6	9.79E-5	0.0279
5E-5	9.02E-5	1.75E-5	1.03E-4	0.0250	9.11E-5	1.79E-5	1.01E-4	0.0274
1E-4	1.07E-4	3.02E-5	1.17E-4	0.0248	1.09E-4	3.12E-5	1.10E-4	0.0272
5E-4	3.31E-4	1.36E-4	3.62E-4	0.0252	3.35E-4	1.43E-4	2.66E-4	0.0278
1E-3	6.65E-4	2.68E-4	7.31E-4	0.0249	6.62E-4	2.87E-4	5.11E-4	0.0275
5E-3	3.92E-3	1.32E-3	4.43E-3	0.0229	3.74E-3	1.51E-3	2.79E-3	0.0254
1E-2	8.82E-3	2.62E-3	9.79E-3	0.0223	8.17E-3	3.11E-3	5.97E-3	0.0251

Table 10:Tolerance method, accuracy and time with n=1600.

The tolerance parameter

As expected, the tolerance controls the accuracy of the option values. For the *Intrinsic* and the *Black-Scholes* replacement values, when the tolerance is set to a value lower than the RMS error of the un-truncated tree, the truncated tree has roughly the same RMS error as the un-truncated tree.

In other words, the RMS error of the truncated tree = RMS error of un-truncated tree + truncation error. As under this truncation method the error is more controlled, we can pick a suitable tolerance depending on the target accuracy and stay close to the un-truncated tree, and the truncation algorithm will ensure the nodes truncated will not adversely affect the accuracy.

The replacement value

The time taken with the *Adjacent value* and *Adjacent time value* replacement values is longer than even the BS replacement error (and less accurate too). This is caused by the truncation algorithm being dependent on the replacement value used, and the larger and less accurate values provided by *Adjacent value* and *Adjacent time value* are adversely affecting the efficiency of the truncation algorithm. Therefore, *Adjacent value* and *Adjacent time value* are not suitable replacement values under the *tolerance* method.

The *Black-Scholes* replacement value results in a more accurate option value, but at the expense of longer time. Therefore, we have a trade-off, for which we present the analysis in “The tolerance method – targeted accuracy” section. In general it would seem for the *Intrinsic* replacement value, setting the tolerance to a value smaller than the targeted accuracy (e.g. 0.0001 for 0.0005 accuracy) is best while for the *Black-Scholes* replacement value setting the tolerance to equal the targeted accuracy (e.g. 0.0005 for 0.0005 accuracy) is best.

The tolerance method – Target accuracy

RMS error

Target	Type	Param	Trunc	Repl	steps	Time(ms)	PerSecond	Improve T12	Improve T13
0.001	Best	5E-04	*Tol*	*Intrinsic*	217.9	1.379	725.3	54%	47%
0.001	Second	1E-04	*Tol*	*Intrinsic*	216.5	1.387	721.0	53%	46%
0.001	Third	5E-05	*Tol*	*Intrinsic*	216.5	1.405	711.6	51%	44%
0.001	Tian 13	6.0	*Strike*	*BS*	216.6	2.027	493.4	5%	
0.001	Un-truncated		none	none	216.6	2.150	470.6		
0.0005	Best	1E-04	*Tol*	*Intrinsic*	375.7	2.857	350.0	117%	54%
0.0005	Second	5E-05	*Tol*	*Intrinsic*	375.8	2.907	343.9	113%	51%
0.0005	Third	5E-04	*Tol*	*Intrinsic*	392.6	2.951	338.8	110%	49%
0.0005	Tian 13	6.0	*Strike*	*BS*	376.1	4.399	227.3	41%	
0.0005	Un-truncated		none	none	376.1	6.223	161.2		
0.0001	Best	1E-05	*Tol*	*Intrinsic*	1333.4	16.590	60.3	357%	61%
0.0001	Second	5E-05	*Tol*	*Intrinsic*	1382.4	16.722	59.8	353%	59%
0.0001	Third	1E-04	*Tol*	*BS*	1349.2	17.185	58.2	341%	55%
0.0001	Tian 13	6.0	*Strike*	*BS*	1336.9	26.642	37.5	184%	
0.0001	Un-truncated		none	none	1336.9	76.402	13.2		

Table 11: Best cases for RMS error under tolerance truncation.

B&D error

Target	Type	Param	*Trunc*	*Repl*	steps	Time(ms)	PerSecond	Improve T12	Improve T13
0.0001	Best	5E-04	*Tol*	*Intrinsic*	186.6	1.133	882.8	43%	47%
0.0001	Second	1E-04	*Tol*	*Intrinsic*	186.4	1.144	874.1	42%	46%
0.0001	Third	5E-05	*Tol*	*Intrinsic*	187.3	1.166	858.0	39%	43%
0.0001	Tian 13	6.0	*Strike*	*BS*	188.0	1.666	600.3	-2%	
0.0001	Un-truncated		none	none	188.0	1.646	615.5		
0.00005	Best	1E-04	*Tol*	*Intrinsic*	344.5	2.550	392.2	121%	61%
0.00005	Second	5E-05	*Tol*	*Intrinsic*	349.7	2.644	378.2	113%	55%
0.00005	Third	1E-05	*Tol*	*Intrinsic*	356.5	2.762	362.1	104%	49%
0.00005	Tian 13	6.0	*Strike*	*BS*	357.9	4.104	243.7	37%	
0.00005	Un-truncated		none	none	357.9	5.658	177.4		
0.00001	Best	1E-05	*Tol*	*Intrinsic*	1200.1	14.329	69.8	375%	71%
0.00001	Second	5E-05	*Tol*	*BS*	1297.1	16.567	60.4	310%	48%
0.00001	Third	1E-05	*Tol*	*BS*	1271.1	16.664	60.0	308%	47%
0.00001	Tian 13	6.0	*Strike*	*BS*	1260.5	24.476	40.9	178%	
0.00001	Un-truncated		none	none	1260.5	67.996	14.7		

Table 12: Best cases for B&D error under tolerance truncation.

Table for Joshi error is available in the Numerical Results section as *Table 2*.

The *tolerance* method intuitively goes well with RMS absolute error, as the tolerance controls the truncation error in each path that flows to the absolute error.

For the two relative error measures, we can in a way interpret them as the absolute error when the option value (for B&D error) is close to 1 or when time value (for Joshi error) is close to 0.5 (an additional weight of 0.5 is added in the Joshi error measure). As the average option value was 13.52 and average time value 6.39, we would expect both relative errors to be more lenient. In fact, the B&D error measure is definitely more lenient, due to errors not measured on options with actual value less than 0.5 and average option value being greater than 1. However, the Joshi error is only more lenient if the arithmetic average is used. Under the root-mean-squared average Joshi error is slightly more strict, due to large relative errors with deeply in the money options with very small time value.

REFERENCES

Amin, K., Khanna, A.,(1994), Convergence of American put options values from discrete- to continuous-time financial models, Mathematical Finance, Vol 4, No 4, 289--304

Andicropoulos, A. D., Widdicks, M., Duck, P. W. and Newton, D. P. (2004) Curtailing the Range for Lattice and Grid Methods. *Journal of Derivatives*, 11, 55–61.

Broadie, M., and Detemple, J. (1996) American Option Valuation: New Bounds, Approximations, and a Comparison of Existing Methods. *The Review of Financial Studies*, 9, 1211–1250.

Chan, J. H., Joshi, M. S., Tang, R. and Yang, C. (2009) Trinomial or Binomial: Accelerating American Put Option Price on Trees. *Journal of Futures Markets*, 29(9), 826-839.

Cox, J. C., Ross, S. A. and Rubinstein, M. (1979) Option Pricing: A Simplified Approach. *Journal of Financial Economics*, 7, 229-263.

Hull, J. and White, A. (1988) The use of the control variate technique in option pricing. *Journal of Financial and Quantitative Analysis*, 23, September, 237–25

Joshi, M. S. (2004) *C++ Design Patterns and Derivatives Pricing.* Cambridge University Press

Joshi, M. S. (2008) *The Concepts and Practice of Mathematical Finance.* Cambridge University Press, 2nd ed.

Joshi, M. S. (2009) The Convergence of Binomial Trees for Pricing the American Put. *Journal of Risk*, Vol 11, Number 4, Summer, 87-108

Ju. N. (1998) Pricing an American option by approximating its early exercise boundary as a multipiece exponential function. *Review of Financial Studies*, 11(3), 627-46.

Ju, N., and Zhong, R. (1999). An approximate formula for pricing American options. *Journal of Derivatives*, 7(2), 31-40.

Lamberton, D., (1998), Error estimates for the binomial approximation of American put options, Ann. Appl. Probab. Volume 8, Number 1 206--233.

Leisen, D. P. (1998) Pricing the American Put Option: A Detailed Convergence Analysis for Binomial Models. *Journal of Economic Dynamics and Control*, 22, 1419–1444.

Leisen, D. P. and Reimer, M. (1996) Binomial models for option valuation-examining and improving convergence. *Applied Mathematical Finance*, 3, 319–346.

Prekopa, A. and Szantai T., (2010), On the analytical—numerical valuation of the Bermudan and American put option, *Quantitative Finance*, Vol 10, No 1, 59 -- 74

Staunton, M. (2005) Efficient estimates for valuing American options. *The Best of Wilmott (Vol. 2)*, United States: Wiley.

Tian, Y. (1993) A modified lattice approach to option pricing. *Journal of Futures Markets*, 13(5), 563–577.

QuantLib (n.d.) QuantLib - A free/open-source library for quantitative finance. Retrieved 24 May 2009, from http://quantlib.org.

West, G. (2005) Better Approximations to Cumulative Normal Functions. *Wilmott Magazine*, May, 70-76.

Zhu, S. P. (2006) An exact and explicit solution for the valuation of American put options. *Quantitative Finance*, Vol 6(3), pages 229-242, June.