Efficient (Non-recombining) Trees

David Heath and Stefano Herzel
Carnegie Mellon and University of Perugia

Introduction

- Often numerical computation of prices under the Black-Scholes model is done by some variant of the CRR "lattice model".
- It's known (Leisen and Reimer, 1996) that for any fixed strike price the errors go to 0 at least as fast as 1/N (N=depth of lattice).
- For "vanilla CRR" the work grows at the rate N^2. Avellaneda and Laurence (1999) show how to "prune" the lattice to reduce the work to rate $N^{3/2}$.

Non-recombining models

- General HJM models require a “tree” structure for computations
  - Why? It follows from Ho and Lee that under certain assumptions the only “up-down = down-up” model is the Ho and Lee model
- A tree of depth N requires 2^N-1 nodes, compared to CRR’s N(N+1)/2 nodes

In this talk we'll

- Give a method for the construction of "good" trees
- Show that the processes and results of these trees converge to the correct values
- Show that for European options in the CRR model, the error decreases no faster than 1/N
- Show that for European options, errors of "good" trees the decrease at least as fast as 1/(number of terminal nodes)

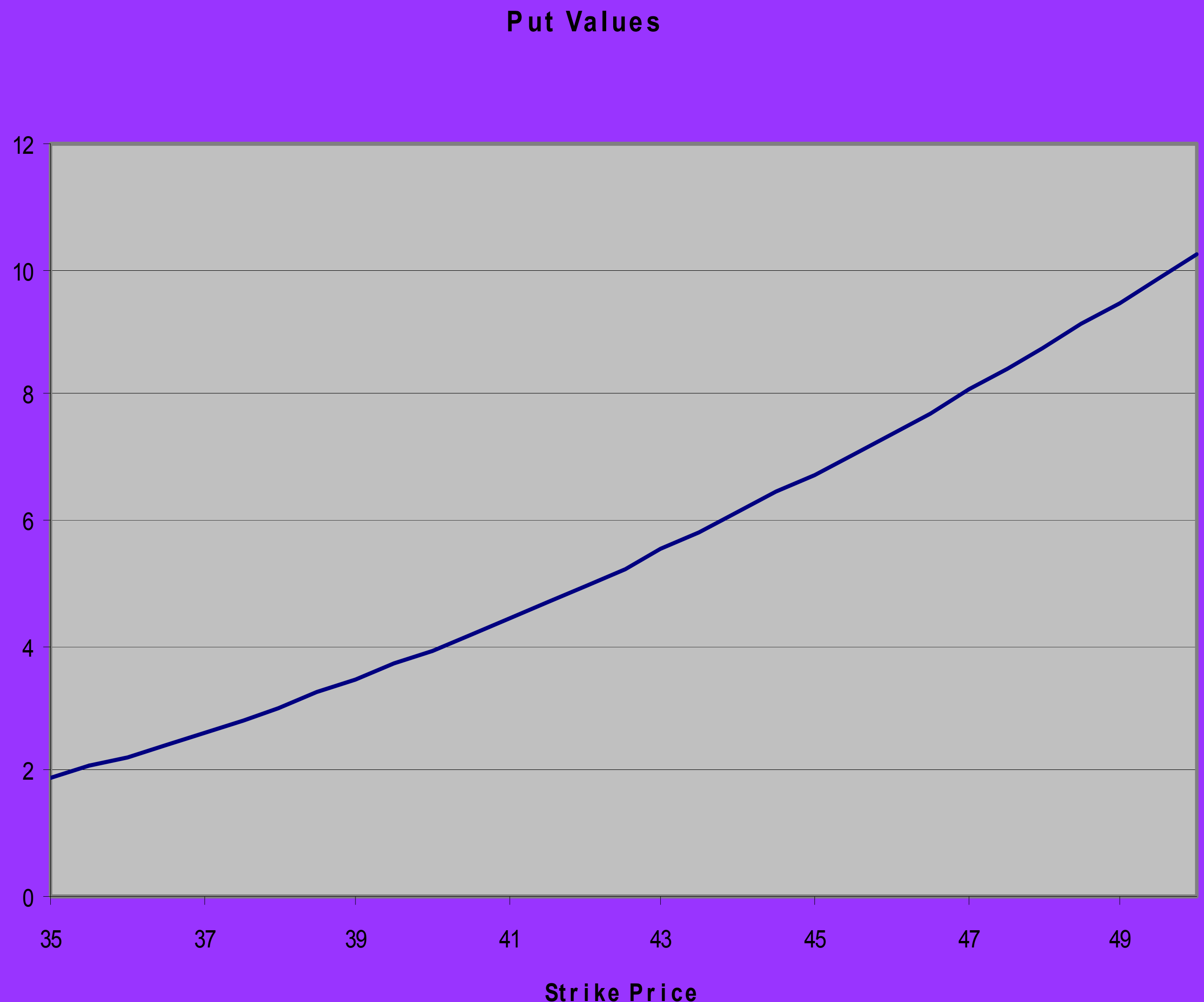
Understanding the errors in CRR computations

Think of valuing European puts at all strikes $K \in [K_1, K_2]$ with $0 < K_1 < K_2$, and a single maturity.

- Let $V_{lattice}(K)$ = Value of option with strike K as computed by a CRR
- Let $V_{B\text{-}S}(K)$ be value obtained from Black-Scholes
- Let $e(K) = V_{lattice}(K) - V_{B\text{-}S}(K)$ = the error

- What can we say about errors for an n-deep lattice?

$S_0=40, t=1, \sigma=.3, r=.04$



Properties of the CRR lattice computation

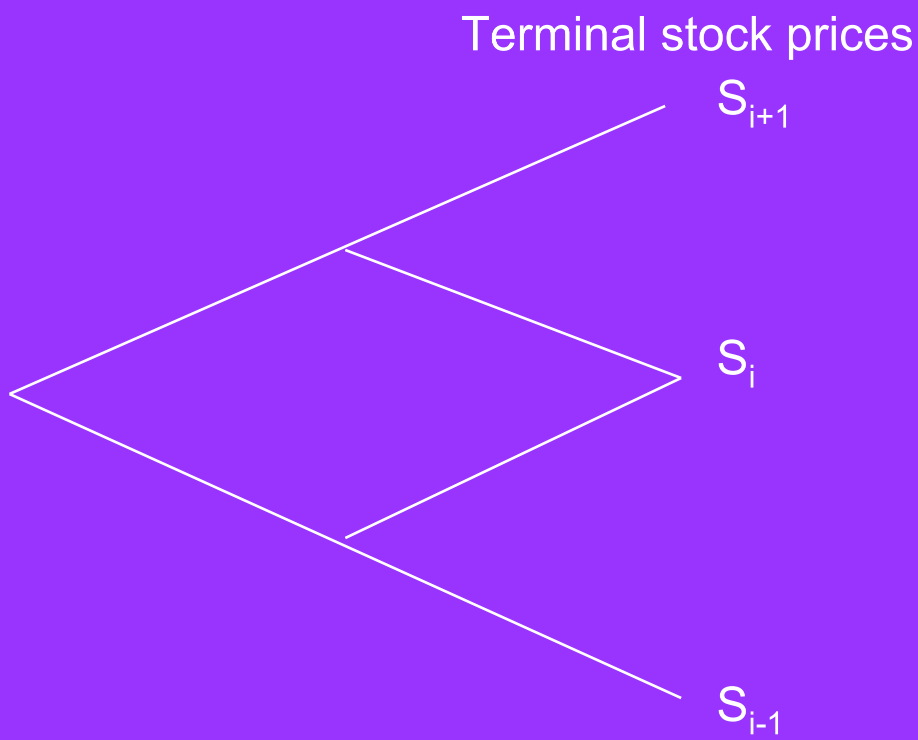
Let $S_i = S_i^{(n)}$ be the i^{th} stock price at depth N

- The differences between the values (i.e., S_{i+1}-S_i) are not constant (I call these "spaces")
- Look at spaces in some fixed interval of strike prices (as N grows)
  - Spacing gets smaller like $N^{-0.5}$
- (Value at terminal node i) = MAX(K-S_i,0) is piecewise linear as a function of K, breaks at S_i's
- This property is preserved by the "backwards induction" through the lattice
- Hence: $V_{lattice}$(K) is piecewise linear with breakpoints at the values of K=S_i for each i

Getting a lower bound for the worst-case error

- Whatever $V_{lattice}$ looks like, it's piecewise linear between values S_i
- The S_i are spaced about (const)*$N^{-0.5}$ apart
- The value V_{B-S} for a put (as a function of K) had a positive second derivative
- Hence on $[S_i, S_{i+1}]$ the error function $e(K) = V_{lattice}(K) - V_{B-S}(K)$ has a negative second derivative, always $\leq -\varepsilon$ for some $\varepsilon > 0$

At the final depth



Put value here =

$(K-S_{i+1})^+$

Put value here =

$(K-S_i)^+$

Put value here =
$(K-S_{i-1})+$

A simple fact

- Suppose that a function f, defined on an interval I = [a,b], has two continuous derivatives and satisfies $f'' \leq -\varepsilon$ on I, then

$$\max_{a \leq x \leq b}(|f(x)|) \geq \varepsilon\, (b-a)^2/16.$$

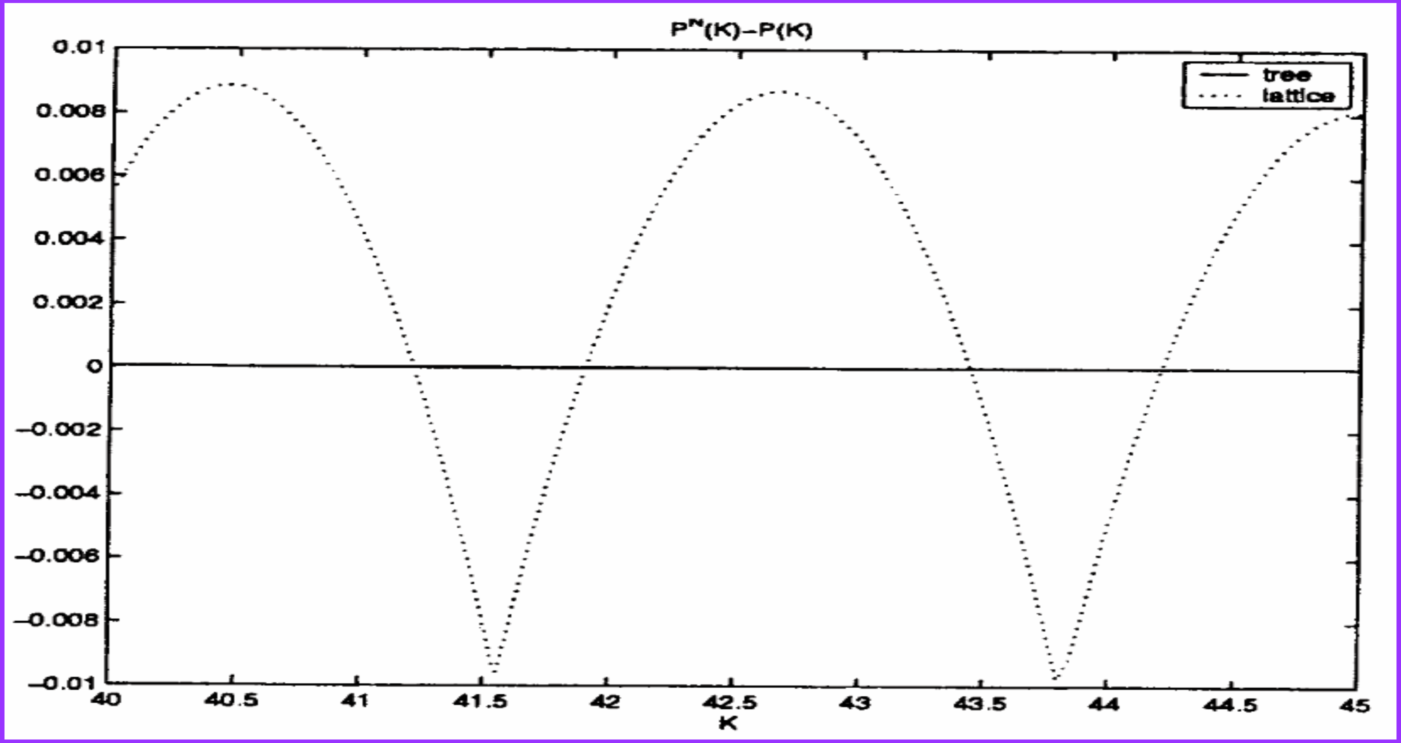
- Applying this result to the error function e on intervals of length exceeding some constant times $n^{-0.5}$ we see that the maximum error size exceeds

$$\varepsilon\,(\text{constant})^2/(16N)$$

where ε is the minimum of the second derivative of $V_{B\text{-}S}$ on the interval.

- Error decreases no faster than 1/N
- Overall effort grows at rate N^2
- Error decreases no faster than $(\text{work})^{-.5}$ so getting one more significant digit requires 100* the work
- Error shape resembles a parabola on each interval (opening downward).

Graph of errors in lattice computations



Errors for S_0=40, σ=0.3, r =.03;

Lattice was 7 deep with 8256 nodes; tree was 12 deep with 8191 nodes

(Tree method is about to be described.)

Properties of lattices and trees

- A lattice of depth n has (n)(n+1)/2 nodes
- A tree of depth n has 2^n-1 nodes
- Comon belief: lattices are better than trees (probably because of above numbers)
- Important question is tradeoff between computation speed and size of errors
- The simplest tree to construct uses the same "steps", but the branches don't recombine

Improving accuracy of trees

- This simplest tree is inefficient because there are only N+1 "really different" terminal stock prices used in a tree of depth N; tree could have had 2^N instead (with same amount of work)
- With only N+1 terminal prices we have the same rate of convergence (vs depth) as with lattices
- But tree of given depth takes much more work than a lattice of that depth.

The trick to making better trees

- To improve things, make the tree “a little irregular” - getting more final stock prices
- Recall: The sizes of the up-and-down motion of the of stock price involved 1/the “square root of the tree depth”.
- Replace this, at each node by a value computed from the following technique:

How to build a good tree

- For European options we’re just trying to compute an integral
- We’ll use a simple method
- We assume that the terminal value of the underlying is bounded and varies in a monotone way as a function of the terminal value of the underlier. (Requiring bounded variation instead would also work with minor changes.)

Some simple analysis

- We want:
  - To model the terminal distribution of the underlier's price(s) well
  - To model the evolution of those prices well (so we can price American options)
- For European options under Black-Scholes we're just doing integration:
  - $V(K) = \exp(-r\ T) * E(\ (K-S_T)^+\ )$

$$E((K-S_T)^+) = \int_{-\infty}^{\infty} (K - S_0 \exp(\sigma T^{.5}\ x - \sigma^2 T/2))^+ \frac{\exp(-x^2/2)}{\sqrt{2\pi}}\, dx$$

Approximating the integral

- Change variables:

$$\text{Set } x = \Phi^{-1}(u) \quad \text{so } dx = \frac{1}{\varphi(\Phi^{-1}(u))} du = \frac{1}{\varphi(x)} du$$

  - Integral becomes

$$\int_0^1 (K - S_0 \exp(\sigma T^{.5}\, \Phi^{-1}(u) - \sigma^2 T/2))^+ \, du$$

  - Notice that the integrand is a monotone (non-increasing) function of u. It's = K for u = 0, and it's = 0 for u = 1.

- Consider more general problem:
- Approximate $\int_0^1 f(u)du$ (assuming f is monotone).

  by a finite average: $\frac{1}{n}\left(\sum_{i=1}^{n} f(u_i)\right)$

- It's easy to see that if we divide [0,1] into n equal size pieces and take one u_i from each piece then,

$$\left|\int_0^1 f(u)du - \frac{1}{\mathrm{n}}\sum_{i=1}^{n} f(u_i)\right| \le \frac{f(0)-f(1)}{n}$$

- Let $x_i = \Phi^{-1}(u_i)$
- These x_i have an approximately normal distribution. (Approximating W(1)).

Summary

- If we choose (x_i) as above for i=1 to 2^d, insert these where W(1) should be in the valuation formula, and take the average (and discount) we'll find:
- Error does not exceed $e^{-rt} K / 2^d$

Approximating Brownian motion on the interval [0,1]:

- We'd like a stochastic process (defined on the experiment: toss a coin n times) whose terminal values are these x_i's which
  - is a martingale
  - is "close to" W (i.e., converges to W in C[0,1])
- We know that the usual random walk on this space converges to W as desired
- Let's make our process close to random walk

- Build a random walk on the space of n coin tosses by counting :
  - Z_k = (# of heads - # of tails in 1st k tosses)/$n^{.5}$
- Note that Z_n takes on values (2i-n)/$n^{.5}$ where i=0, 1, …, n corresponds to the number of heads experienced in the n tosses.

$$\text{Exactly} \binom{n}{i} \text{ paths end up at } \frac{2\mathrm{i} - \mathrm{n}}{\sqrt{\mathrm{n}}}$$

Example

- Arrange the desired x's in increasing order
- For 5 tosses, the terminal values and numbers of paths are:

Value	# of paths	x-values to match
-5	1	x_1
-3	5	$x_2, x_3, \ldots, x_6$
-1	10	$x_7, x_8, \ldots, x_{16}$
1	10	$x_{17}, x_{18}, \ldots, x_{26}$
3	5	$x_{27}, x_{28}, \ldots, x_{31}$
5	1	x_{32}

In other words

- Build a random variable on coin-toss space:
  - x_1 is the value for the "all-tails" sequence
  - $x_2, \ldots, x_6$ are the values for the 5 sequences which consist of 4-tails and 1-head
    - assign these 5 values arbitrarily to these 5 sequences
  - etcetera. Call the resulting random variable X.
- This builds a random variable taking on values $x_1, \ldots, x_{32}$ which is "as close as possible" to the random variable Z_5.

Getting the tree process

- Now we construct X_k by:
  - $X_k = E(X \mid \text{first k coin toss outcomes})$
- Clearly, $X_n = X$.
- Now since X was "close to" Z_5, we know
  - Difference between X_5 and Z_5 is small
  - X_k-Z_k is a martingale, so
    - $X_k - Z_k = E(X_5 - Z_5 \mid \text{first k tosses})$ is small
- (This follows from Doob's maximal inequality for martingales.) Hence the approximation is good for all t

Weak convergence

- If we interpolate between "tree dates" by keeping the process constant between these dates, we can get weak convergence of the tree processes to Brownian motion (in the Skorohod topology).
- This means that the prices of all bounded continuous claims converge to the correct values.

Summary

- For a "usual" lattice computation, the errors decrease no faster than 1/depth, while work grows at rate n^2.
- For good trees the errors decrease like 2^{-n}, while work grows like 2^n.
- The tree models converge weakly to Brownian motion (so the prices of all bounded continuous claims converge to the correct value).

The published version is:

- Efficient option valuation using trees (2002), David Heath and Stefano Herzel, Applied Mathematical Finance 9, 163-178.