18

American Options under Jump-diffusions: an Introduction[1]

18.1 Introduction

Jump-diffusion models are becoming increasingly important in modern option pricing. In combination with stochastic volatility, they provide a well-motivated foundation on which to explain market pricing. After all, prices do jump and volatility is stochastic. Option prices are very sensitive to the details.

In this month's column, I provide an introduction to American-style option pricing under jump-diffusions. Perhaps you are a code developer charged with implementing some version of this for a trading group. Or a researcher just entering this topic who wants an overview. Then, this article is for you. I may also have something to interest the experts.

Here's the good news. If you understand the behavior of American-style options under GBM (geometric Brownian motion: the Black-Scholes model), then you won't be surprised. Jumps add some wrinkles but the overall picture is similar to the one you know. For the researchers, perhaps the good news is bad news, but for them there is still plenty to do – that's because accurate, controllable, and efficient approximation schemes are still needed. Especially if we accept that the "real world" consists of jump-diffusions + stochastic volatility: then, in that world, traders need fast, reliable pricing for general models. Much is still needed for that.

What is this basic picture? We have one because of a decade of work on this problem by the experts: Kaushik Amin [5], Jesper Andreasen and Barbara Gruenewald [9], Huyen Pham [206], Leif Andersen and Jesper Andreasen [6], Svetlana Boyarchenko and Sergei Levendorskii [35], and others. The ingredients of the picture are: (i) the small-time behavior, (ii) the large-time behavior, (iii) put/call duality or symmetry, and (iv) general-time behavior, seen via numerical algorithms. First, I'll review the mathematical set-up of our problem and then discuss each of these items in turn. More detail may be found in the references, with the exception of my remarks on put/call duality, which are perhaps new.

[1] First published in Wilmott Mag., (Mar., 2003), 46-51.

18.2 The problem

First, let's briefly review the model. Under a jump-diffusion, the (risk-adjusted) stock price S_t evolution follows a combination of geometric Brownian motion B_t plus an independent, Poisson-driven, compound jump process. The jumps arrive with intensity (mean arrival rate) λ and $N_t = 0, 1, 2, \ldots$ counts the cumulative jumps to time t. When the stock price jumps: $S \to S\mathrm{e}^{x_t}$. Here $x_t \in (-\infty, \infty)$ is a jump amplitude, randomly drawn from some (risk-adjusted) jump distribution: $x_t \sim q(x)$. For example, for Merton's 1976 jump-diffusion model [186], $q(x)$ is a normal distribution parameterized by a mean-variance pair (μ_J, σ_J^2). Merton's model is sometimes called *the* jump-diffusion model, but in this article, $q(x)$ is an arbitrary probability distribution, with discrete or continuous support.

In a useful but somewhat awkward notation, the stock price evolution is given by

$$S_T = S_0 \exp\{\omega T + \sigma B(T)\} \prod_{0<t\le T} \mathrm{e}^{x_t} 1_{\{N_t \neq N_{t^-}\}}, \qquad x_t \sim q(x). \tag{18.1}$$

The product terms are defined to be e^{x_t} when there is a jump and 1 otherwise. The constant ω is the drift and is chosen to make the discounted stock price process a martingale. Throughout, we'll assume that an owner of the stock receives a continuous dividend yield δ, making this a theory most appropriate for broad-based index options. In that case, the drift is given by $\omega = r - \delta - \sigma^2/2 - \lambda k$, where r is an interest rate, σ is the diffusion volatility and $k = \int (\mathrm{e}^x - 1) q(x) dx$. The American-style put option, for example, then has a fair value given by

$$P(S_t, t) = \max_{\theta \in [t,T]} \mathbb{E}_t[\mathrm{e}^{-r(\theta - t)}(K - S_\theta)^+]. \tag{18.2}$$

In (18.2), θ is the time at which it is optimal to exercise (the stopping rule time), T is the expiration time, K is the strike price, and $\mathbb{E}_t$ denotes the time t expectation. I also use throughout the notation $f^+ = \max(f, 0)$. This stopping time problem has the general solution: exercise at the first moment when $S_t \le S_p^*(\tau)$, where $S_p^*(\tau)$ is the critical exercise boundary (subscript indicates the put) and $\tau = T - t$. This rule divides the (t, S) space into two regions: (i) a stopping region $S_t \le S_p^*(\tau)$, where $P(S, t) = K - S$ and (ii) a continuation region where $P(S, t) > (K - S)^+$. In the continuation region (using subscripts for derivatives), $P(S, t)$ satisfies the equation

$$\begin{aligned} P_t - rP + (r - \delta - \lambda k) S P_S + \frac{1}{2}\sigma^2 S^2 P_{SS} \\ + \lambda \int [P(S\mathrm{e}^x, t) - P(S, t)]\, q(x) dx = 0. \end{aligned} \tag{18.3}$$

Smooth Fit and Other Principles. In the Black-Scholes case, it is well-known that $P(S, t)$ is both continuous and has a continuous S-derivative across the exercise boundary. This is the smooth-fit condition and it continues to hold for the jump-diffusion case. Pham [206] gives a short proof of this, and reports that the result was earlier obtained by Xiao Lan Zhang in his 1994 dissertation.[2] In addition, other results familiar from the GBM model continue to hold: $P_{SS} \ge 0$, $P_t \le 0$ ($\partial P / \partial \tau \ge 0$), and $dS_p^*(\tau)/d\tau \le 0$.

[2] See Zhang [252]. Boyarchenko and Levendorskii [35] show that the smooth fit principle will fail for some types of pure-jump Levy processes. However, as far as I am aware, this failure doesn't apply to any of the many continuous-time models that have so far been suggested by researchers for security prices.

18.3 Small-time behavior

Consider an American-style call option with strike K. Right at expiration, assuming you haven't previously exercised, of course you will exercise if $S_T > K$. So $S_c^*(0) = K$. In correspondence, Jesper Andreasen reminded me that sometimes for American-style options, $S^*(0^+) \neq K$; that is, the critical boundary can be discontinuous at expiration. When does this happen?

First, recall how this works for the Black-Scholes model. If you are a short time Δt from expiration, with the stock price S, the continuation value is $\mathrm{e}^{-r\Delta t}\mathbb{E}[(S_T - K)^+]$. Think of a binomial lattice model where the two possible values for S_T are both at least as large as K. Then, the continuation value is

$$\mathrm{e}^{-r\Delta t}\mathbb{E}[(S_T - K)^+] = S\mathrm{e}^{-\delta\Delta t} - K\mathrm{e}^{-r\Delta t} \approx (S - K) - \Delta t(\delta S - rK).$$

This is less than the exercise value $S - K$ whenever $\delta S > rK$, so the exercise boundary at Δt is $S_c^*(\Delta t) = \max(1, r/\delta)K$. This means that the critical boundary is discontinuous as $\Delta t \to 0$ whenever $r > \delta$, dropping from $(r/\delta)K$ to K.

Andreasen and Gruenewald [9] extend this argument to various jump processes. I'll just frame their result in a little more generality and leave the proof to the reader:[3]

Proposition 18.1 *Consider an American-style option with critical boundary* $S^*(0^+) = \lim_{\Delta t \downarrow 0} S^*(\Delta t)$ *in the jump-diffusion* (18.1) *where both* $(r, \delta) > 0$. *Then,* $S^*(0^+)$ *is given by the following:*

$$\text{Call:} \quad \frac{S_c^*(0^+)}{K} = \max(1, A), \quad \text{where} \qquad A\delta - r - \lambda \int 1 - A\mathrm{e}^x)^+ q(x)\,dx = 0.$$

$$\text{Put:} \quad \frac{S_p^*(0^+)}{K} = \min(1, A), \quad \text{where} \qquad A\delta - r + \lambda \int (A\mathrm{e}^x - 1)^+ q(x)\,dx = 0.$$

An Example. Consider two possible point jumps from my Feb 2003 column example:[4] an up-jump with amplitude $\mathrm{e}^{x_+} = 1.25$ with (conditional) probability q_+ and a down-jump with $\mathrm{e}^{x_-} = 0.50$ and probability $q_- = 1 - q_+$. The remaining parameters are $r = 0.08$, $\delta = 0$, $\sigma = 0.40$, and $\lambda = 1$. All numerical results with point jumps in this article will use this example. So, Proposition 18.1 says that, for the put, you need to solve $r = q_+(1.25A - 1)^+ + (1 - q_+)(0.5A - 1)^+$. There is a discontinuity in the exercise boundary only if there is a solution $A < 1$. The borderline case $A = 1$ occurs when $q_+ = 4r = 0.32$. For example, for $q_+ = \{0, 1/4, 1/2, 3/4, 1\}$ the proposition yields $S_p^*(0^+) = \{1, 1, 0.928, 0.88533, 0.864\}$ respectively.

As a corollary, we see that when $\delta = 0$ (no dividends), then the put option boundary is continuous at expiration if $r > \lambda \int (\mathrm{e}^x - 1)^+ q(x)\,dx$. Huyen Pham proved (in [206], Theorem 4.1) that if $r > \lambda \int (\mathrm{e}^x - 1)^+ q(x)\,dx$, then $S_p^*(\tau)$ is continuous for $\tau > 0$. Our corollary slightly sharpens his result to $\tau \geq 0$. In addition, we see that if there are *no positive jumps*, then the put option boundary is continuous for all $\tau \geq 0$ whenever $r \geq \delta$.

[3] As a hint for the interested reader, consider differentiating the integrals in the Proposition with respect to A. This shows they are monotone in A for all $A \geq 0$. That's why we know that the solutions are unique

[4] This is reprinted here as the chapter: 'Perpetual American Options Made Easy', which precedes this one.

18.4 Large-time behavior

This topic was the subject of the previous column, so I'll be very brief here. As with GBM, the idealized limit of a perpetual American-style option has an exact solution. Since the limit is time-homogeneous, the critical exercise boundary becomes a single number S^*, where I'll drop the τ dependence.

For any jump-diffusion, the characteristic function of the process has the form $\mathbb{E}[\exp(\mathrm{i}zX_T)] = \exp(-T\psi(z))$, where $\psi(z) = -\mathrm{i}z\omega + z^2\sigma^2/2 - \lambda\int(\mathrm{e}^x - 1)\,q(x)\,dx$. Then, the so-called resolvent (or Laplace transform of the characteristic function) may be factored $r/(r+\psi(z)) = \phi_r^+(z)\phi_r^-(z)$. Explicitly, one obtains the factors from a straightforward complex-plane integration. In Mathematica, for example, just compute:

$$\phi_r^-(\xi) = \exp\left\{\int_{(\Im\,\xi)^+ < \Im z < \sigma^+} \frac{dz}{-2\pi\mathrm{i}}\,\frac{\xi\log\left[r+\psi(z)\right]}{z(\xi - z)}\right\}. \tag{18.4}$$

In (18.4), you integrate parallel to the real z-axis, in a strip bounded below by $(\Im\,\xi)^+ := \max(\Im\xi, 0)$. The strip is bounded above by σ^+, where $\mathrm{i}\,\sigma^+$ is the first zero of $r+\psi(z)$ with positive imaginary part.[5]

Once you've coded (18.4), the critical boundary for the put option is $S_p^* = K\phi_r^-(-\mathrm{i})$. A related complex-plane (double) integration also gives the put option value. These results are due to Boyarchenko and Levendorskii [35]. Things are simpler when there are either no positive jumps: $S_p^* = -\mathrm{i}rK/\psi'(-\mathrm{i})$, or no negative jumps: $S_p^* = \sigma^+K/(\sigma^+ + 1)$. Our working example, with two point jumps, tests all 3 of these cases. For example at $q_+ = (0, 1/2, 1)$, then I computed previously $S_p^*/K = (0.2552, 0.3215, 0.4314)$. These results will be shown consistent with a simple lattice algorithm below.

18.5 Put/call duality

Write the solution to the American-style put option problem as $P(S,t) = g(x,t)$, where $x = \log S$. In the continuation region, (18.3) becomes

$$g_t - rg + (r - \delta - \frac{1}{2}\sigma^2 - \lambda k)g_x + \frac{1}{2}\sigma^2 g_{xx} + \lambda\int\left[g(x+z,t) - g(x,t)\right]q(z)dz = 0.$$

Now let $g(x,t) = \mathrm{e}^{-y}h(y,t)$, where $y = -x$. It's easy to show that $h(x,t)$ satisfies an equation of exactly the same form, namely

$$h_t - \hat{r}h + (\hat{r} - \hat{\delta} - \frac{1}{2}\sigma^2 - \hat{\lambda}\hat{k})h_x + \frac{1}{2}\sigma^2 h_{xx} + \hat{\lambda}\int\left[h(x+z,t) - h(x,t)\right]\hat{q}(z)dz = 0.$$

but where now

$$\hat{r} = \delta, \quad \hat{\delta} = r, \tag{18.5}$$

$$\hat{\lambda} = \lambda(k+1) = \lambda\int \mathrm{e}^x q(x)\,dx, \quad \text{and} \quad \hat{q}(x) = \frac{\mathrm{e}^{-x}q(-x)}{\int \mathrm{e}^{-x}q(-x)\,dx}. \tag{18.6}$$

[5] This zero $\mathrm{i}\sigma^+$ is assumed to occur before reaching any analyticity boundary of $\psi(z)$.

And of course $\hat{k} = \int (\mathrm{e}^x - 1)\hat{q}(x)\,dx$. Equations (18.5-18.6) generalize a well-known "put/call symmetry" from the Black-Scholes case, but I like the term *duality*. The dual quantities are the ones with the hats: $\hat{}$. This transformation can be interpreted as a change in numeraire from, say, dollars to shares, but we don't need that. For our purposes, duality simply takes a put option solution P into a call option solution C for a (different) jump-diffusion model. If you apply it twice, you end up back where you started. Specifically, if we display most of the parameter dependencies, it's easy to see that

$$P(S, K, t; \{r, \delta, \lambda, q(\cdot)\}) = C(K, S, t; \{\delta, r, \hat{\lambda}, \hat{q}(\cdot)\}). \tag{18.7}$$

Also, the critical exercise boundaries are related by:

$$\frac{S_p^*(\tau; \{r, \delta, \lambda, q(\cdot)\})}{K} \times \frac{S_c^*(\tau; \{\delta, r, \hat{\lambda}, \hat{q}(\cdot)\})}{K} = 1. \tag{18.8}$$

These relations are both interesting and useful. For example, for code developers, you don't need to develop for both puts and calls. You can just develop for puts and evaluate calls using (18.7). Alternatively, if you have coded for both, it's a good check to verify (18.7-18.8). Note that (18.7) applies both to the American-style and Euro-style solutions.

In general, the dual jump model is a different model; the jump distribution might not be from the same parameterized class. But consider the Merton jump-diffusion where $q(\cdot)$ is a normal density with parameters (μ_J, σ_J^2). In that case $\hat{q}(\cdot)$ is also normal but with the parameters $(\hat{\mu}_J, \sigma_J^2)$, where $\hat{\mu}_J = -(\mu_J + \sigma_J^2)$.[6]

Remember we are working here entirely with the "risk-adjusted" process. Recall from "Fear of Jumps" the risk adjustments from a power utility equilibrium model. Both power utility and duality multiply jump distributions by an exponential factor, among other things. So, various additional relations could be derived by combining the two transformations. I'll leave that to the interested reader.

18.6 Basic explicit lattice algorithms

Consider a simple explicit lattice model for the (non-jumping) GBM case. That is, with $x = \log S$, create a rectangular lattice (not a tree) with spacing $(\Delta t, \Delta x) = (\Delta t, \sigma(\Delta t)^{1/2})$. At each lattice point, if you advance in time Δt, then the log-stock price is allowed to move up or down Δx. The transition probabilities for this are $p_\pm = 1/2 \pm (\omega/\sigma)(\Delta t)^{1/2}$, where $\omega = r - \delta - \sigma^2/2$. This can be thought of as a "forward" recursion. The American-style put option solution is obtained from the "backwards" recursion

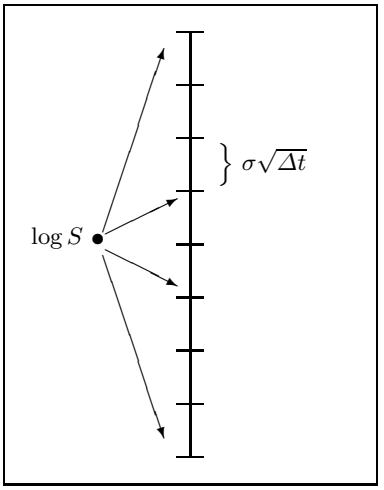
$$P(S_t, t) = \max\Big\{(K - S_t), \mathbb{E}_t[\mathrm{e}^{-r\Delta t} P(S_{t+\Delta t}, t + \Delta t)]\Big\}. \tag{18.9}$$

In the above, $S_t = S_{t(i)} = \exp(x_i)$ is the value of S at a lattice point and $S_{t+\Delta t} = \exp(x_{i\pm1})$ is the value of S at the two allowed (Brownian motion) transitions. The expectation only consists of these two terms with transition probabilities given above. You start the recursion at $t = T - \Delta t$ and work your way backward to the desired valuation time. Of course, the lattice is truncated so that $x_i \in (x_0 - x_{max}, x_0 + x_{max})$ and you have approximately $N_X = 2\,x_{max}/\Delta x$ values of x_i. There are $N_T = T/\Delta t$ time-steps and the original problem is solved as $\Delta t \to 0$, and $x_{max} \to \infty$. In practice, x_{max} can typically be taken to be a small integer.

Within this basic explicit lattice algorithm, you can add jumps. (See Figure 18.1). With our working example, consider again the stock price evolution looking forward in

[6] The Merton model can also be "self-dual". When the expected jump size is zero $(\mu_J = -\sigma_J^2/2)$ and $r = \delta$, then the dual parameters are identical to the original.

Fig. 18.1. Allowed Transitions for GBM + a Point Jump to Two Points



time from a lattice point x. Now, there can also be a jump to one of two possible points which are at a distance $x_\pm = \{\log(1.25), \log(0.50)\}$ from x. Of course, these two points will generally lie in between lattice nodes. For the moment, let's ignore that and just consider the 4 transition probabilities associated with Figure 18.1. First, we have the 2 Brownian motion transitions. These now have probabilities $(1 - \lambda\Delta t)p_\pm$ where the $p_\pm$ is given above but it uses $\omega = r - \delta - \sigma^2/2 - \lambda k$. Next, we have the 2 jump transition probabilities, which are given by $\lambda\Delta t q_\pm$. Pretty simple.

To make it work, we split up each of the jump transitions shown in the figure to the two surrounding nodes, with the probabilities weighted by the relative distances to the closest nodes. So, in the algorithm, for this example, we have a total of 6 allowed transitions. That's it! It's a simple modification of the binomial model, easily coded. Like its binomial cousin, it's not the most efficient routine in the world. Nevertheless, you would probably want to code this one before moving on to more advanced routines. We call it the "basic explicit lattice algorithm".

Some code. I coded the above routine for the working example model in a simple C/C++ console app.[7] There's not much to it beyond what we have discussed. It treats the put option. At a given time to expiration $\tau_n = n\Delta t$, I first iterate over all the nodes greater than the strike price. Of course, you would never exercise on those so you don't even check, and you can break at the first zero put price (since the price is monotonic in S). Next, you iterate from the strike price and below. If $r > 0$ (and x_{max} is large enough!) you will always hit a pair of adjacent stock prices (S_i, S_{i+1}) such that $P(S_i) = K - S_i$ and $P(S_{i+1}) > K - S_{i+1}$, suppressing the dependence on the level n. Again, you can break.

[7] The code, called `LevyLattice.cpp`, may be found at `http://www.optioncity.net/publications.htm`

Then, the put option should be exercised at a stock price $S_i \le S_p^*(\tau_n) \le S_{i+1}$, and I just use a simple linear interpolation to estimate it.[8]

Some results. Table 18.1 shows some results from my code. I show the critical exercise boundary and the at-the-money put values for three different times to expiration: $T = \{10^{-6}, 1, 100\}$ years. (At $T = 10^{-6}$, I only take one time step to show S^*). In addition, I give the theoretical $T = \infty$ results from my Mathematical implementation of the Boyarchenko/Levendorskii formulae (see (17.17) and discussion). After a little experimentation, I settled on x_{max} cutoff values of 1, 3, and 10 for the 3 times in the Table. I don't believe any greater values for x_{max} would change the results shown.

The algorithm works well at both large and small times, although to get the critical exercise boundary accurately, you need a lot of time steps. This is typical of the binomial model. The boundary discontinuity at $T = 0$ for $q_+ > 0.32$ is correctly reproduced. You don't have to do anything special to get it. At $T = 100$, the asymptotic perpetual results are well-approximated. If you just want "penny accurate" option values, then fewer time steps are needed (see below).

Some more code: log-normal jumps. It's straightforward to extend this algorithm to arbitrary jump distributions. I did this for the Merton's model; this extension is essentially the algorithm of [5]. Whatever your jump distribution, you truncate it at some high and low critical points and allocate the area under the distribution among the nodes. For example, for the normal distribution, I allowed jump transitions to each lattice point within the interval $\mu_J \pm 3\sigma_J$ plus the two surrounding points. This rule produces N_J jump transitions, where $N_J \approx 2 + [6\,\sigma_J/(\sigma\Delta t^{1/2})]_-$. It also *exactly* reproduces my point jump results (at $q_+ = 0$ or $q_+ = 1$) as $\sigma_J \to 0$. For Table 18.1, $N_J \le 4$, but clearly for the Merton model, N_J can be much larger.

For example, in "Fear of Jumps", I drew a smile chart for some one-month SPX options, and fitted it roughly to Merton's jump diffusion. Let's use those parameters again. They were: $r = 0.018$, $\delta = 0.017$, $\sigma = 0.254$, $\lambda = 0.30$, $\mu_J = -0.25$, and $\sigma_J = 0.15$. Let's round SPX to exactly 1000 and consider $T = 1$ year. Now that I want to look at different strikes, I changed the lattice variable slightly to $x = \log(S/K)$ so that $S = K$ is always the zero lattice node. The x-lattice was truncated at $x_{min,max} = \pm 3$.

Some results for various strikes may be found in Table 18.2. The exact solution of the Euro-style model is shown. The "Lattice" results are from the algorithm I have just described. The "Control Variate" entry is just the American-style Lattice value plus the Euro-style error added back. This control variate correction, often applied to the binomial model, seems very useful here.[9] Especially since fast computations are available for the exact Euro-style values for any jump diffusion (see Lewis [163], reprinted here). Using this correction, you may consider the Table 18.2 entries with 250 time steps accurate enough for trading applications. (I would). If so, we have a pretty fast method since you can see that this American-style result (ignoring the time to get the control variate) only required a run time of 0.3 seconds on my desktop machine.[10]

[8] Specifically, let the slope $m = (P(S_{i+1}) - P_c(S_i))/(S_{i+1} - S_i)$, where $P_c(S_i)$ is the (non-optimal) continuation value below the exercise boundary. Then, I estimate $S_p^* = (K - P_c(S_i) + mS_i)/(1 + m)$.

[9] Another well-known technique, extrapolation to the limit, performed poorly for me in this application. I tried to apply it by assuming that prices were converging with some power series in $1/n^{1/2}$.

[10] Compiler: MS Visual Studio.net C++ compiler. Machine: 2Ghz P4 Gateway desktop. To get the control variate, you have to run the same lattice routine for the Euro-style option. This costs another 0.5 sec, say. Then, you need the exact Euro result. I got it from some Mathematica code. A compiled routine will definitely cost you no more than

18.7 More efficient schemes

If you double the number of time steps, from whatever level, it's easy to see that the run time of the basic explicit algorithm increases by a factor of 4. That's because the run-time is proportional to $N_T \times N_X \times N_J$. Doubling the time steps increases (i) N_T by 2, and the other two factors by $\sqrt{2}$ each. If this run-time becomes too large for your application, perhaps because your jump distribution is relatively dispersed, then it's time to explore more efficient approaches. Here is a brief literature guide.

If the number of jump nodes becomes large, you may be able to use the Fast Fourier Transform (FFT) to more quickly evaluate the convolution-type sums. This idea (among others) is applied in Andersen and Andreasen [6]. The FFT algorithm itself is explained in Press et al. [213] and you can also use it to get the Euro-style exact results needed for the control variate method (see Carr and Madan [46]).

Implicit finite difference schemes for jump-diffusions are discussed in references already mentioned and also in Tavella and Randall [232]. If you're working in an environment with ready access to some good ODE routines, then you should also try Gunter Meyer's [190] *method of lines* approach to this problem. I found Meyer's method very easy to prototype in Mathematica using the built-in NDSolve function. Finally, if your application involves more general Levy processes, where the jump intensity parameter doesn't exist, check out the approach in Hirsa and Madan [129].

Special case schemes. Finally, recall from the SPX example, that the fitted parameters for the jump distribution were $\mu_J = -0.25$ and $\sigma_J = 0.15$. The mass of the jump distribution on the negative axis is $\Phi(-\mu_J/\sigma_J) \approx 0.952$ or more than 95%. This suggests that the SPX could just as easily be fit with a *truncated jump distribution*, truncated so that there are no positive jumps. If you have no positive jumps, then for put options you can employ a much more efficient algorithm, using the *early exercise representation*. The early exercise representation is developed in Pham [206], and its discretization is discussed in Andreasen and Gruenewald [9] and Gukhal [120]. This important method can be applied to the put problem with arbitrary (negative) jump distributions. I'll discuss it in an upcoming column.[11]

another small fraction of a sec for the Merton model. For any other jump model, it's at most a single complex-plane integration.

[11] See the article "American-style Options with One-sided Jumps" (Ch. 21).

Table 18.1. Basic Explicit Lattice Algorithm for Jump-diffusions. Two Point Jumps: American-style ($S_0 = K = 100$)

Jump Probability		$T = 10^{-6}$	$T = 1$			$T = 100$			$T = \infty$	
Up	Down	(S^*/K)	Tsteps	(S^*/K)	Put	Tsteps	(S^*/K)	Put	(S^*/K)	Put
0.0	1.00	0.9996	10^3	0.4703	24.687	10^4	0.2574	51.131		
			10^4	0.4693	24.678	10^5	0.2561	50.894		
			10^5	0.4688	24.677	10^6	0.2557	50.870	0.2552	50.868
0.25	0.75	0.9996	10^3	0.4976	22.588	10^4	0.2881	47.457		
			10^4	0.4958	22.583	10^5	0.2861	47.296		
			10^5	0.4954	22.582	10^6	0.2851	47.280	0.2847	47.278
0.50	0.50	0.9280	10^3	0.5252	20.198	10^4	0.3269	42.970		
			10^4	0.5234	20.197	10^5	0.3230	42.912		
			10^5	0.5228	20.197	10^6	0.3220	42.906	0.3215	42.906
0.75	0.25	0.8853	10^3	0.5534	17.548	10^4	0.3750	37.316		
			10^4	0.5515	17.552	10^5	0.3704	37.396		
			10^5	0.5510	17.552	10^6	0.3694	37.404	0.3688	37.405
1.00	0.00	0.8640	10^3	0.5826	14.757	10^4	0.4417	29.730		
			10^4	0.5807	14.767	10^5	0.4343	30.012		
			10^5	0.5799	14.768	10^6	0.4321	30.040	0.4314	30.044

Table 18.2. Basic Explicit Lattice Algorithm. Log-normal jumps ($S_0 = 1000$)

		Put Option Values			
		Euro-style		American-style	
Strike	N_{Tsteps}	Lattice	Exact	Lattice	+Control Variate
600	250	5.53	5.55	5.54	5.56
	1000	5.54		5.55	5.56
	4000	5.54		5.56	5.57
1000	250	110.82	110.89	111.38	111.45
	1000	110.87		111.42	111.44
	4000	110.88		111.44	111.46
1400	250	408.67	408.60	412.52	412.45
	1000	408.62		412.48	412.46
	4000	408.61		412.47	412.46

Table 18.3. Run-times for the Table 18.2 entries

N_T (Tsteps)	N_X (Xsteps)	N_J (Jumps)	Run-time(Amer-style)
250	381	58	0.3 sec.
1000	761	115	4.7 sec.
4000	1519	229	73 sec.