Computational Finance – Pricing The American Option

1 Introduction

Recall that the American option has strike K and maturity T and gives the holder the right to exercise at any time in $[0, T]$. The American option is not straightforward to price in the Monte Carlo framework that we have discussed. The reason is that the derivative cash flow function $f(S, t)$ is not well defined. The problem is that we cannot compute the derivative cash flow until we know how the American option is going to be exercised. If, on the other hand, we knew the optimal exercise strategy, then it would be a straightforward task, using Monte Carlo, to obtain the expected discounted cashflows, and hence the price.

Lets first define what an exercise strategy is. Denote an exercise strategy by $\pi(S, t)$, which is a binary valued function of two variables, the price and the time. The exercise strategy $\pi(S, t)$ specifies whether to exercise or not at the state (S, t),

$$\pi(S,t) = \begin{cases} 1 & \text{exercise in state } (S,t), \\ 0 & \text{do not exercise in state } (S,t). \end{cases}$$

2 Review of the Risk Neutral Stock Dynamics

Remember that all pricing occurs in the *risk-neutral* world, which is governed by the *Martingale measure*. Let's first recall the stock dynamics in the risk neutral world,

$$\begin{aligned} dS &= rSdt + \sigma S dW, \\ d\log S &= (r - \tfrac{1}{2}\sigma^2)dt + \sigma dW, \end{aligned}$$

where dW is a more formal way to write $\sqrt{dt}\epsilon(t)$ with $\epsilon(t)$ being a zero mean unit variance independent random variable, and σ is the real world volatility of the stock. We have alternatively written this random process as

$$S(t + \Delta t) = S(t)e^{\eta},$$

where $\eta \sim N((r - \frac{1}{2}\sigma^2)\Delta t, \sigma^2 \Delta t)$. Since $S = e^{\log S}$ and $\log S$ is a real valued random process, this means that S is a positive valued random process. Intuitively, S has a reflective barrier at zero. The expected move in S is an increase by a factor $e^{r\Delta t}$, a consequence of the risk neutral dynamics because all prices are martingales,

$$S(t) = E[e^{-r\Delta t} S(t + \Delta t)].$$

Intuitively, as S gets closer to 0, it will tend to move up by larger additive factors than down. When S is far from 0, this is stll the case, but the asymmetry about $Se^{r\Delta t}$ will not be as severe. This intuition has an important implication for the American call option, namely that it is never optimal to exercise early.

3 The American Call Option

If at any time the stock S is below the strike K, then there is no reason to exercise. If, on the other hand, $S > K$, there is a choice to be made. Should one exercise now, and obtain an instant profit of $S - K$, or wait in the hopes that S increases fast enough (to offset discounting). If S increases fast enough, we can exercise later and make more money.

The intuition above that the stock is more likely to increase faster than $e^{r\Delta t}$ than slower seems to suggest that it is better to wait and exercise later. This argument seems applicable to any time, thus it should be *always* better to wait. This intuition seems to lead to the bizzare conclusion that it is never optimal to exercise the American call option before expiry. In this case, the American call option is exactly a European call option, and so its price is also exactly the same as that of the European call. Is there something wrong with the intuition? While it seems plausible that S is more likely to increase than decrease, does it always increase at a fast enough rate? The answer is *yes*, and we are in fact led to the following theorem

Theorem 3.1 *The American call option and the European call option are equivalent.*

To prove this, we will simply show that it is never optimal to exercise. Consider time t where a decision to exercise appears for the first time, i.e., $S(t) - K > 0$. The cash flow from exercising is thus $S(t) - K$. Consider now the alternative strategy of waiting for a small time Δt and then exercising. Lets compute the expected discounted cash flow for this strategy (which is available to us, since we hold an American option). We want $E[e^{-r\Delta t}(S(t + \Delta t) - K)^+]$. Since $(S(t + \Delta t) - K)^+ \geq S(t + \Delta t) - K$, we have

$$\begin{aligned} E[e^{-r\Delta t}(S(t+\Delta t) - K)^+] &\geq E[e^{-r\Delta t}(S(t+\Delta t) - K)], \\ &= E[e^{-r\Delta t}S(t+\Delta t)] - e^{-r\Delta t}K, \\ &\overset{(a)}{=} S(t) - e^{-r\Delta t}K,, \qquad (1) \\ &> S(t) - K, \end{aligned}$$

where (a) follows because we are in the risk neutral world (Martingale world), which means that in this world, the price today of every instrument $(S(t))$ is the expected discounted price tomorrow $(E[e^{-r\Delta t}S(t + \Delta t)])$. Thus we see that by waiting a fixed period Δt, the expected discounted cash flow is larger in the risk neutral world. Since waiting for a fixed period Δt is only a *subset* of the options available to the holder of the American option, by waiting and *optimally* exercising later, we should be able to access even higher discounted cash flows. We conclude that for every time t where we have the choice to exercise, it is better to wait. This concludes the proof of Theorem 3.1. Note that the proof we have given is not specific to the GBM risk neutral process, and in fact applies to any risk neutral process.

Exercise 3.1

Explicitly verify equation (1) by explicitly computing $E[e^{-r\Delta t}S(t+\Delta t)]$ to obtain $S(t)$.

[Hint: Use the fact that $S(t+\Delta t) = S(t)e^{\eta}$. You thus need to compute $E[e^{\eta}]$, where $\eta \sim N((r-\frac{1}{2}\sigma^2)\Delta t, \sigma^2\Delta t)$.]

The unfortunate (or fortunate) outcome is that there is no more work to do for the American call option. We thus turn to the American put option.

4 The American Put Option

It seems that the same argument above should apply to the American put option. Consider a time t when $K - S(t) > 0$, i.e. the holder has a decision to make as to exercising or not. If we go through the same analysis as above for the *fixed* strategy of waiting to exercise at time Δt, we obtain

$$E[e^{-r\Delta t}(K - S(t+\Delta t))^+] \geq e^{-r\Delta t}K - S(t).$$

The $\geq$ arises from the fact that $(K - S(t+\Delta t))^+ \geq K - S(t+\Delta t)$. Now the RHS is strictly less that $K - S(t)$ and all we know is that the LHS is at least the RHS so we can not conclude anything about whether it is better to exercise now or wait a small time Δt and exercise. In fact one might argue that if $K - S(t) > 0$ and Δt is small enough, approaching 0, then this $\geq$ becomes close enough to equality – i.e., for small enough Δt, $e^{-r\Delta t}K - S(t) \approx E[e^{-r\Delta t}(K - S(t+\Delta t))^+] < K - S(t)$. Thus it looks like by waiting a fixed time Δt and then exercising, one obtains smaller cash flow. This is in fact true, and so it is always better to exercise now than wait a (small) *fixed* time and exercise later. However, remember that waiting and exercising after a small fixed time is only a *subset* of the options available to the holder of the American put option – if the holder decides to wait, then he will exercise *optimally* later. Optimal exercise in the future has a value at least that of waiting for a small fixed period and exercising, but if the additional value of optimal exercise over waiting a fixed time cannot overcome the discounting ($e^{-r\Delta t}$), then it may be optimal to exercise now than wait and optimally exercise later. Thus, we are not in the same boat as the American call option –

What can we say about the optimal exercise strategy? We can get some general properties of the optimal exercise strategy. In particular, the two properties that we would like to establish are that

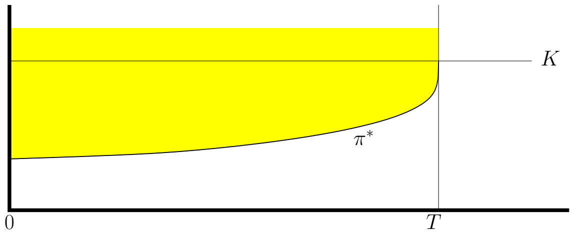
1. At every time t, there is an optimal exercise price point $\pi^*(t)$. Below the price $\pi^*(t)$, it is optimal to exercise at time t and above this price, it is optimal not to exercise, holding and optimally exercising later, over $(t, T]$. The function $\pi^*(t)$ defines an optimal exercise boundary.

2. The optimal exercise boundary $\pi^*(t)$ is non-decreasing, with $\pi^*(T) = K$.

The first property is true for certain price processes in the risk neutral world, and the second is true for any Markovian risk neutral dynamics. The upshot of all this discussion is that the optimal exercise strategy can be represented by an *optimal exercise threshold function* $\pi^*(t)$. The optimal exercise strategy is then given by

$$\pi^*(S,t) = \begin{cases} 1 & S < \pi^*(t), \\ 0 & S \geq \pi^*(t). \end{cases}$$

Further, the optimal exercise threshold function $\pi^*(t)$ is a monotonically increasing function of t. The situation is illustrated in the figure, where the shaded region indicates the states where it is not optimal to exercise.



We will now prove the second property for any Markovian risk neutral dynamics, in particular our GBM risk neutral dynamics. Suppose that it is optimal to exercise in state (S, t). This means that the cash flow from exercising, equal to $K - S$ is at least the expected discounted cashflow from optimal exercise over $(t, T]$ starting from price S at time t. In particular, for any exercise strategy,

$$K - S \geq E[\text{discounted cashflow for exercise according to any exercise strategy } \pi].$$

Now consider the state (S, τ) for any $\tau > t$. Immediate exercise gives the cash flow $K - S$, just as in the state (S, t). Suppose that it is not optimal to exercise the option. In this case, there must exist some exercise stratety π which yields an expected discounted cash flow greater than $K - S$, where we follow the exercise strategy π starting from state (S, τ). Now consider using this strategy π starting at time t with the stock at price S, i.e., starting from the state (S, t). Since the risk neutral dynamics is Markovian, the price dynamics starting from state (S, t) over a time period of length $T - \tau$ are exactly the same[1] as the dynamics starting from state (S, τ) over the time period of length $T - \tau$ (to maturity), because the future dynamics over a time period of length $T - \tau$ only depends on the current price which in both cases is S. Thus, starting from state (S, t) and following exercise strategy π one has the same expected discounted cashflows over the time interval $(t, t+T-\tau]$ as one would have from following strategy π from state (S, τ) to the maturity of the option. By assumption, this expected cash flow is greater than $K - S$, and so we have a stretegy π which starting

[1] in the sense that every price path has the same probability

from state (S,t) produces an expected discounted cash flow greater than $K-S$, which is a contradiction. Thus, we conclude that if it is optimal to exercise in state (S,t), then it is optimal to exercise in state (S,τ) for every $\tau \geq t$. This proves that $\pi^*(t)$ is non-decreasing in t.

Proving that there is a well defined exercise point $\pi^*(t)$ below which it is optimal to exercise and above which it is optimal to hold is a little tricky and depends on the particular risk neutral dynamics. The approach is to show that if $\pi^*(S,t)=1$ then $\pi^*(S',t)=1$ for all $S' \leq S$. We give an intuitive argument and leave a more formal argument to an exercise. Suppose that it is optimal to exercise at (S,t). Then, intuitively, it is better to take the money $K-S$ and run, than wait and optimally exercise later. Thus, intuitively, it should be optimal to exercise at (S',t) for all $S'<S$, since one is getting more money. This becomes particularly so since due to the reflecting barrier at 0, the stock is "more likely" to move up (relative to $e^{r\delta t}$) than down and hence it is even more imperative to take the money and run – since the stock price is even more likely to go up from S' than it was from S, one should definitely take the money $K-S'$ if it was already optimal to take $K-S$ and run since the asymmetry in the up versus down moves has gotten worse. The next exercise gives a slightly more formal discussion of this statement, which should probably be skipped on a first reading.

Exercise 4.1

Let π^* be the optimal exercise strategy. Show that if $\pi^*(S,t)=1$, then for all $S'<S$, $\pi^*(S',t)=1$.

The following sketch should guide you through the argument. Imagine starting two identical processes at (S,t) and (S',t), $S'<S$. Let π_S be the optimal exercise strategy starting from (S,t) and correspondingly $\pi_{S'}$ the optimal exercise strategy starting from (S',t). Consider now the process $\Delta\log S$ and $\Delta\log S'$. These are identical processes. We can thus define the optimal exercise strategies equivalently in terms of the processes $\Delta\log S$ and $\Delta\log S'$. Consider the paths defined with respect to $\Delta\log S'$. Let $\{p_\alpha\}$ be the paths on which $\pi_{S'}$ would exercise at time $\tau(p_\alpha)$ with $\Delta_\tau(p_\alpha)$ denoting $\Delta\log S'$ at this time of exercise on p_α. Since one is exercising, it must be that $K-S'e^{\Delta_\tau(p_\alpha)}>0$

Let $E_{p_\alpha}=\int_{\{p_\alpha\}} d\mu(p_\alpha)$, where $d\mu(p_\alpha)$ is the risk neutral measure. Then, since it is not optimal to exercise from (S',t), it must be that

$$K-S' < E_{p_\alpha}[e^{-r\tau(p_\alpha)}(K-S'e^{\Delta_\tau(p_\alpha)})]. \tag{2}$$

$\pi_{S'}$ also defines an exercise strategy starting from S (though it may be sub-optimal). Since $\Delta\log S$ and $\Delta\log S'$ are identical processes, the measure is the same. Further, since it is optimal to exercise at S, it must be that

$$K-S \geq E_{p_\alpha}[e^{-r\tau(p_\alpha)}(K-Se^{\Delta_\tau(p_\alpha)})]. \tag{3}$$

Compining (2) and (3),

$$S' - S > (S' - S)E_{p_\alpha}[e^{-r\tau(p_\alpha)}e^{\Delta_\tau(p_\alpha)}],$$

and since $S' - S < 0$,

$$1 < E_{p_\alpha}[e^{-r\tau(p_\alpha)}e^{\Delta_\tau(p_\alpha)}]. \tag{4}$$

Rearranging (2) and using (4), we obtain

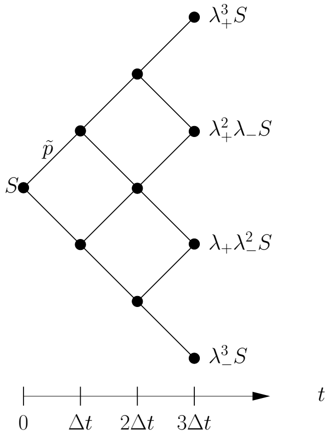
$$K(1 - E_{p_\alpha}[e^{-r\tau(p_\alpha)}]) \quad < \quad S'(1 - E_{p_\alpha}[e^{-r\tau(p_\alpha)}e^{\Delta_\tau(p_\alpha)}]) < 0,$$

which is clearly a contradiction because the leftmost term is at least 0 since $r > 0$ and $\tau(p_\alpha) \geq 0$.

We now consider several methods for obtaining the optimal exercise threshold function, and the price of the American put option.

5 Dynamic Programming on the Binomial Tree for Pricing the American Put

Our first algorithm will be based on a dynamic programming approach for the binomial tree risk neutral dynamics of a stock. We briefly recap the binomial tree model for the risk neutral dynamics. The binomial tree is illustrated in the figure below for three time steps.



in which the parameters $\lambda_\pm$ are given by

$$\begin{aligned}\lambda_+ &= e^{\mu\Delta t + \sigma\sqrt{\frac{1-p}{p}}\sqrt{\Delta t}},\\ \lambda_- &= e^{\mu\Delta t - \sigma\sqrt{\frac{p}{1-p}}\sqrt{\Delta t}}.\end{aligned}$$

Here μ, σ are the real world drift and volatility, and p is arbitrary. We have droped the subscript R for simplicity of notation. The risk neutral probability is given by

$$\tilde{p} = \frac{e^{r\Delta t} - \lambda_-}{\lambda_+ - \lambda_-}.$$

In the n step binomial tree, there are n discrete time steps, $\Delta t, 2\Delta t, \ldots, n\Delta t$, with $n\Delta t = T$ (T is the maturity of the option). At time step $i\Delta t$, there are $i+1$ possible stock values, $\lambda_+^k \lambda_-^{i-k} S$ for $k = 0, \ldots, i$. Thus we can use the pair of indices (i,k) to index a node on the binomal tree (it is actually a grid). Note that the total number of nodes on the grid is $1 + 2 + \cdots + n + (n+1) = \Theta(n^2)$.

Let's introduce two quantities, $V(i,k)$ and $\pi^*(i,k)$ to denote the *value* of holding the option and being at node (i,k) and the optimal exercise strategy (1 or 0) evaluated on (i,k). Remember that (i,k) stands for the time $i\Delta t$ at which the stock price is $\lambda_+^k \lambda_-^{i-k} S$. Clearly $V(0,0)$ is the price of the option. From $\pi^*(i,k)$ one can easily compute the optimal exercise threshold function $\pi^*(i)$. We will thus focus on evaluating the quantities $V(i,k)$ and $\pi^*(i,k)$ efficently.

Consider the last time step, n with the nodes (n,k), $k \in [0,n]$. Since at time T, the only option is to exercise if it is profitable to do so, it is clear that

$$\begin{aligned} \pi^*(n,k) &= \begin{cases} 1 & K - \lambda_+^k \lambda_-^{n-k} S \geq 0, \\ 0 & K - \lambda_+^k \lambda_-^{n-k} S < 0, \end{cases} \\ V(n,k) &= (K - \lambda_+^k \lambda_-^{n-k} S)^+, \end{aligned}$$

We now show how to compute $V(i-1,k)$ and $\pi^*(i-1,k)$ for all $k \in [0, i-1]$ given that we know $V(i,k')$ for all $k' \in [0,i]$. Once we have done this, the algorithm will be clear. We start with $\{V(n,k)\}_{k=0}^n$ and $\{\pi^*(n,k)\}_{k=0}^n$ which are known. We then compute $\{V(n-1,k)\}_{k=0}^{n-1}$ and $\{\pi^*(n-1,k)\}_{k=0}^{n-1}$ from $\{V(n,k)\}_{k=0}^{n-1}$, and so on, proceeding backwards to $V(0,0)$.

Consider $V(i-1,k)$. In state $(i-1,k)$ there are two options: exercise immediately if the option is in the money, in which case the cash flow is $(K - \lambda_+^k \lambda_-^{i-1-k} S)^+$; or, wait. If we wait, there are two possible scenarios: the stock goes up to $\lambda_+^{k+1} \lambda_-^{i-1-k} S$, i.e. to the node $(i, k+1)$; or, the stock goes down to $\lambda_+^k \lambda_-^{i-k} S$, i.e. to the node (i,k). The values $V(i,k+1), V(i,k)$ of being in both of these states are known (by assumption). Thus, the expected discounted value (under the risk neutral measure) of holding can be computed as

$$V_h(i-1,k) = e^{-r\Delta t}(\tilde{p} V(i,k+1) + (1-\tilde{p}) V(i,k)).$$

Similarily, the value of exercising is

$$V_{ex}(i-1,k) = (K - \lambda_+^k \lambda_-^{i-1-k} S)^+.$$

If the cash flow of exercising is at least the expected cash flow of waiting, then it is optimal to exercise, and vice versa. Thus, we conclude that

$$\pi^*(i-1,k) = \begin{cases} 1 & V_{ex}(i-1,k) \geq V_h(i-1,k), \\ 0 & V_{ex}(i-1,k) < V_h(i-1,k), \end{cases} \tag{5}$$

$$V(i-1,k) = \max\{V_{ex}(i-1,k), V_h(i-1,k)\}. \tag{6}$$

Equations (5) and (6) are the key steps in the dynamic programming algorithm. Notice that every node of the tree will be visited, and at each node a constant amount of work is done, so the running time of the algorithm is $\Theta(n^2)$. The quadratic running time is essentially unavoidable. If all the values $\pi^*(i,k)$ and $V(i,k)$ are stored, then the memory requirement is also $\Theta(n^2)$, which for n reasonably large is unmanageable. Luckily, it is possible to run the algorithm with only a memory requirement $\Theta(n)$, to store the exercise function $\pi^*(i)$ and obtain the price $V(0,0)$. The main idea is that once $V(i-1,k)$ is computed, $V(i,k')$ will never be needed again, and so that memory can be reused.

Exercise 5.1

Show more explicitly that only $\Theta(n)$ memory is needed for the dynamic programming algorithm. Try to minimize the memory requirement as much as you can.

We give the full algorithm in pseudo code below.

```
1:  **Algorithm:** Pricing American Put using Dynamic Programming
2:  Select $\Delta t$ and compute $\lambda_\pm$ and $\tilde{p}$.
3:  Initialize $\pi^*(n) = K$ and a vector $\mathbf{v}$ of size $n+1$ to $v_k = V(n,k)$ for $k \in [0,n]$.
4:  for $i = n-1$ to $0$ do
5:     for $k = 0$ to $i$ do
6:        $v_k \leftarrow \max\{(K - \lambda_+^k \lambda_-^{i-k} S)^+, e^{-r\Delta t}(\tilde{p} v_{k+1} + (1-\tilde{p}) v_k)\}$
7:        if $v_k \le (K - \lambda_+^k \lambda_-^{i-k} S)^+$ then
8:           $\pi^*(i) = \lambda_+^k \lambda_-^{i-k} S$
9:  $v_0$ is the option price.
10: $\pi^*(i)$ contains the optimal exercise threshold function.
```

The algorithm as stated above is perfectly fine for an infinite precision machine, however on a finite precision machine, a certain amount of care needs to be taken. In particular, lets consider the "simple" task of initializing $\mathbf{v}$ in step [3] of the algorithm. This involves computing the stock price s_k since $v_k = (K - s_k)^+$, so we consider the initialization of s_k, for $k = 0, \ldots, n$. A natural approach would be to first initialize $s_0 = \lambda_-^n S$ and then initialize the remaining s_k for $k \in [1,n]$ using the update $s_k = \frac{\lambda_+}{\lambda_-} s_{k-1}$. Let's consider the numerical value of $s_0 = \lambda_-^n S$. Note that for $\Delta t = \frac{T}{n}$ small enough, $\sqrt{\Delta t} \gg \Delta t$ and so $\lambda_- \approx e^{-\sigma\sqrt{\frac{p}{1-p}}\sqrt{\frac{T}{n}}}$. Thus,

$$\lambda_-^n \quad \approx \quad e^{-\sigma\sqrt{nT}\sqrt{\frac{p}{1-p}}}.$$

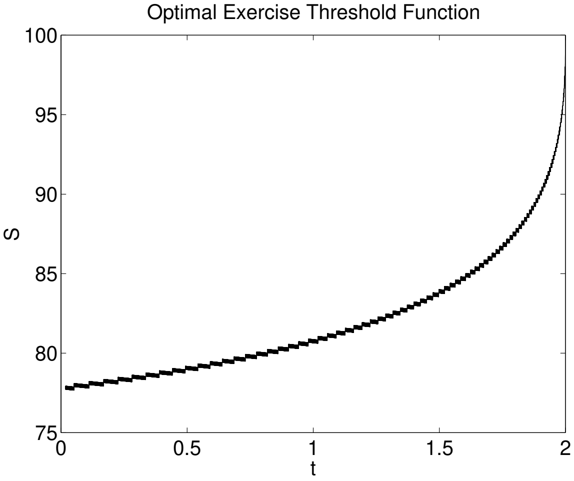
Thus, for large n, to within the numerical precision of most computers, s_0 will evaluate to 0, and if the update $s_k = \frac{\lambda_+}{\lambda_-} v_{k-1}$ is used with starting condition $s_0 = 0$, all the s_k will be zero, and the algorithm will be doomed from the begining.

Exercise 5.2

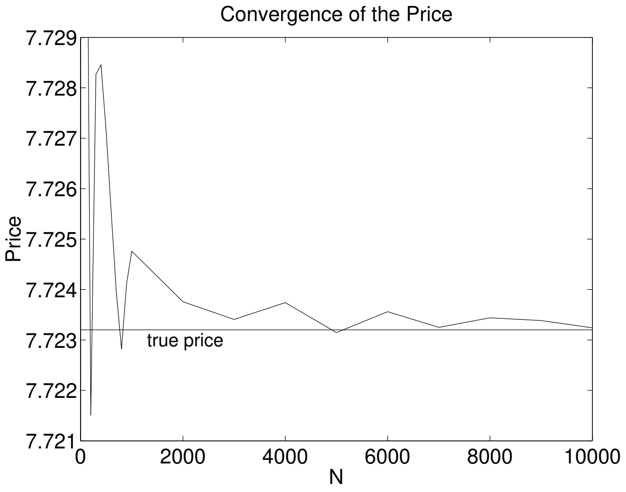
Give a better approach to innitializing the vector $\mathbf{v}$ than the one discussed above.

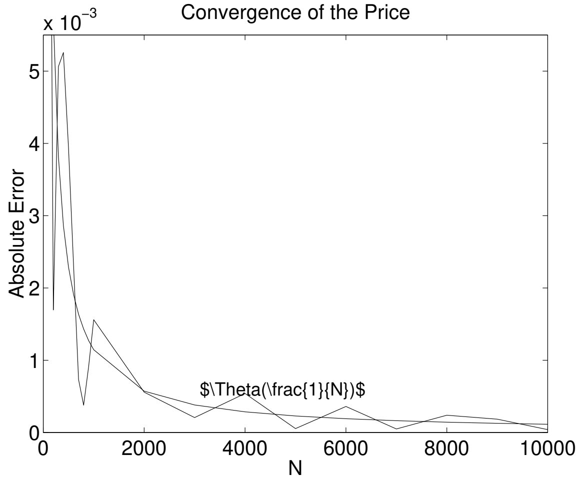
[Hint: Consider computing $\log s_k$. Show that $\log s_k = k \log \lambda_+ + (n - k) \log \lambda_- + \log S$.]

As an example, running our dynamic programming algorithm with $S = K = 100$, $T = 2$ years, $r = 0.05$ (annualized), $\sigma = 0.2$ (annualized) and $n = 200,000$, the American put option price was 7.723197. The optimal exercise threshold function is given in the figure below.



As one increases n, the price converges to the true price, hence this method is one of a class of methods known as convergent methods. However, the behavior of this convergence is interesting. It is not monotonic, so the price cannot be used to bound the true price in any systematic way. It is known that for the Binomial tree with n discretization time steps, the absolute error in the price converges at a rate $\Theta(\frac{1}{n})$. Some of these behaviors are illustrated in the figures below.





6 Monte Carlo Pricing of the American Put

As already mentioned, the straightforward Monte Carlo approach will not work, as the derivative cash flows are not well defined given a path. As a result, the typical approach to price the American put using Monte Carlo is a two stage process:

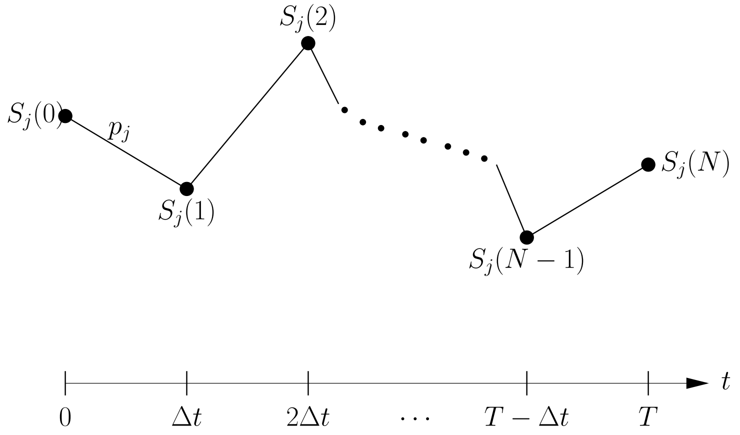
1. First obtain the optimal exercise threshold function.
2. Given the optimal exercise function, the derivative cash flows for any path are now well defined, hence the price can be obtained using the straight forward Monte Carlo approach discussed earlier.

Once again, the algorithm will be a dynamic programming one, however it will somehow be based on a set of Monte Carlo paths instead of on the Binomial tree.

So, suppose that M Monte Carlo paths $p_1, \ldots, p_M$ have been generated. We will be interested in computing $V(S,t)$, the value of holding the option at time t when stock price is S. Once again, assume that t is discretized into n time steps, $t = 0, \Delta t, \ldots, n\Delta t$ ($\Delta t = T/n$). If we use the continuous model for generating the stock, then S will be continuous.

Recap (generating paths for the risk neutral world in the continuous model). Let S_0 be the initial price. The dynamics of the process $\log S$ are $d \log S = (r - \frac{1}{2}\sigma^2)dt + \sigma dW$. To generate one path, generate n independent random variables $\Delta_i \sim N((r - \frac{1}{2}\sigma^2)\Delta t, \sigma^2 \Delta t)$. Set $\Delta \log S_i = \sum_{k=1}^{i} \Delta_i$ and $S_i = S(i\Delta t) = S_0 e^{\Delta \log S_i}$.

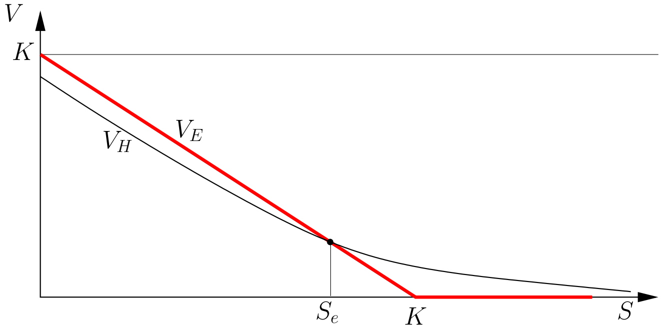
Let's examine in detail a single path, for example p_j, $j \in [1, M]$. We denote the $n+1$ stock prices along the path p_j at the times $i\Delta t$ for $i \in [0, n]$ by $S_j(i)$. The figure below illustrates.



As usual, we begin at time T, where $V(S,T)$ is known, $V(S,T) = (K-S)^+$, and $\pi^*(T) = K$. Now consider $V(S, T - \Delta T)$. The value of exercising, $V_{ex}(S, T - \Delta T) = (K - S)^+$. If we could compute the value of holding and optimally exercising after time $T - \Delta t$ which we will denote $V_h(S, T - \Delta t)$, then we would immediately have

$$\begin{aligned} V(S, T - \Delta T) &= \max\{V_{ex}(S, T - \Delta T), V_h(S, T - \Delta t)\}, \\ &= \max\{(K - S)^+, V_h(S, T - \Delta t)\}, \end{aligned}$$

where $V_h(S, T - \Delta t)$ is the expected discounted cash flows from waiting and optimally exercising later, after $T - \Delta t$. For the moment, we will focus on $V_h(S, T - \Delta t)$ and so for notational simplicity, we will drop the dependence on $T - \Delta t$ and write $V_h(S)$. Lets first try to get a handle on what $V_h(S)$ should look like as a function of S. First, since the next available time step is Δt in the future, and the cash flow from exercise can never be more than K, we see that $V_h(S) \leq Ke^{-r\Delta t}$. Certainly $V_h(S) > 0$ for all S, so we immediately have two trivial bounds. Further, it is reasonable to assume that $V_h(S)$ is monotonically decreasing in S (this was more formally argued in the previous section when we discussed optimal exercise). A property of $V_h(S)$ that we will not prove here is that it is convex (and hence continuous). The figure below summarizes all this information about $V_h(S)$. Also shown is $V_{ex}(S) = (K - S)^+$.



Since $V_h(0) < V_{ex}(0)$ and $V_h(K) > V_{ex}(K) = 0$, there is a point of intersection of these two curves for some $S_e \in [0, K]$. By the convexity of the two functions, this point of intersection is unique, and is exactly the value of the stock below which it is optimal to exercise and above which it is optimal to hold. According to the notation in the figure, $\pi^*(T-\Delta t) = S_e$. Thus, we see that if we had V_h, then solving the equation $V_h(S) = V_{ex}(S)$ for S_e gives us the optimal exercise threshold. To be completely explicit, lets completely define what $V_h(S)$ is.

> $V_h(S)$ *is the expected discounted cash flows from holding and optimally exercising after time* $T - \Delta t$.

6.1 The Least Squares Method (LSM)

We now come back to the path p_j. Remember that $S_j(n-1)$ is a realized stock price at time $T-\Delta t$. Corresponding to the realized price $S_j(n-1)$ is the realized price $S_j(n)$, also on the path p_j. Thus, we have the *pair* of prices $(S_j(n-1), S_j(n))$. We know the value $V(S_j(n)) = (K - S_j(n))^+$. Thus, *for this particular path* p_j, the discounted value of holding and *optimally* exercising later must equal the discounted value $e^{-r\Delta t}V(S_j(n))$. The expected value of this quantity $E[e^{-r\Delta t}V(S_j(n))]$ over all paths p_j passing through $S_j(n-1)$ is exactly $V(S_j(n-1))$,

$$V_h(S_j(n-1)) = E_{p_j}[e^{-r\Delta t}V(S_j(n))|p_j \text{ passes through } S_j(n-1)].$$

We could thus replace this expectation with an average over all the paths that pass through $S_j(n-1)$, however it is very unlikely that there is more than one such path. No fear, we will use the above equation to write

$$V_h(S_j(n-1)) = e^{-r\Delta t}V(S_j(n)) + \epsilon_j,$$

were $E[\epsilon_j|S_j(n-1)] = 0$. Or equivalently,

$$e^{-r\Delta t}V(S_j(n)) = V_h(S_j(n-1)) + \epsilon_j.$$

In words, $e^{-r\Delta t}V(S_j(n))$ is a (noisy) unbiased estimate of $V_h(S_j(n-1))$. To summarize this discussion, let $x_j = S_j(n-1)$ and $y_j = e^{-r\Delta t}V(S_j(n))$. Then

$$y_j = V_h(x_j) + \epsilon_j.$$

In words, we wish to infer the *function* $V_h(x)$, and we have a set of examples

$$\{(x_1, y_1), \ldots, (x_M, y_M)\},$$

which are noisy *examples* (one from each path p_j) of the function $V_h(x)$. In this form, we have a standard *learning* or *regression* problem. We can proceed by trying to learn the function $V_h(x)$ by regressing the $\{y_j\}$ on the $\{x_j\}$. In the original paper which introduced this approach, Longstaff and Schwarz used a quadratic form for $V_h(x)$,

$$V_h(x) = a_0 + a_1 x + a_2 x^2,$$

and used a least squares approach to "fitting" a_0, a_1, a_2 to the data $\{(x_j, y_j)\}_{j=1}^M$.

Exercise 6.1

Suppose that one assumes an order d polynomial form for $V_h(x)$,

$$V_h(x) = \sum_{k=0}^{d} a_k x^k.$$

Let the vector $\mathbf{a}$ be given by $\mathbf{a}^T = [a_0 a_1 \cdots a_d]$, and define the matrix $\mathbf{X}$ and vector $\mathbf{y}$ by

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^d \\ 1 & x_2 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_M & x_M^2 & \cdots & x_M^d \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}.$$

The sum of squared errors is then given by

$$E = (\mathbf{Xa} - \mathbf{y})^T (\mathbf{Xa} - \mathbf{y}).$$

Minimize this error with respect to $\mathbf{a}$ to obtain the least squares solution

$$\mathbf{a}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

Exercise 6.2

Compute exactly the value function $V_h(S, T - \Delta t)$, without making any use of the paths.

Answer: $V_h(S, T - \Delta t) = Ke^{-r\Delta t}\Phi(-d_-) - S\Phi(-d_+)$, where

$$d_\pm = \frac{\log \frac{S}{K} + (r \pm \frac{1}{2}\sigma^2)\Delta t}{\sigma\sqrt{\Delta t}}, \text{ and } \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} ds\ e^{-\frac{1}{2}s^2}$$

is the standard normal distribution function.

[Hint: Argue first that holding the option at time $T - \Delta t$ is like holding a European option. What is the strike, what is the time to expiry and what is the initial stock price?]

Given the result of the last exercise, it seems a little overkill to generate paths, and do a fitting exercise just to obtain $V_h(S, T - \Delta t)$, when we can compute it exactly. We persevere with this approach because it is not only $V_h(S, T - \Delta t)$ that we need, but we need V_h for all times. Since we know that entire function $V_h(S, T - \Delta t)$, then why don't we fit this function itself? The reason is that the simple (say) quadratic model will not be able to fit this entire function well. Thus we would rather fit it in the *important* region. How do we know what the important region is? This is exactly how the the paths help us. They serve to (probabilistically) identify the important region as the points that are likely to occur on a typical path. Thus in a typical path, the stock price will "tend to pass" through one of the $S_j(n-1)$, thus it is important to optimally exercise in these regions. Sub-optimally exercising on very unlikely paths will not affect our pricing of the put option. An alternative question is then whether it should be better to use the exact price to obtain $V_h(S_j(n-1))$ rather than the approximate price furnished by y_j. It is true that this will improve the fit function a little. However, we will not be able to easily extract the exact function V_h for earlier times, so this improvement will only come at time $T - \Delta t$ and will not affect the result significantly. However, when n is small, then this additional improvement could be useful.

We will use the notation $\hat{V}_h(S)$ to denote our estimate of the *entire function* $V_h(S)$. When we explicitly want to show that $\hat{V}_h$ is specified by the set of parameters $\mathbf{a}$ inferred from the examples $\{x_j(n-1), y_j(n-1)\}_{j=1}^{M}$, we will write $\hat{V}_h(S; \mathbf{a}_{n-1})$ – note that the subscript $n-1$ on $\mathbf{a}$ indicates the time step.

Recap:

$$\begin{aligned}
x_j(n-1) &= S_j(n-1), \\
y_j(n-1) &= e^{-r\Delta t} V(S_j(n)) = e^{-r\Delta t}(K - S_j(n))^{+} \\
&= V_h(S_j(n-1)) + \epsilon_j, \\
\mathbf{a}_{n-1} &= \text{parameters } \mathbf{a} \text{ which optimally fit the examples } \{x_j(n-1), y_j(n-1)\}_{j=1}^{M}, \\
\hat{V}_h(S; \mathbf{a}_{n-1}) &= \text{our estimate for the function } V_h(S, (n-1)\Delta t). \hat{\pi}^*(n-1) = S : \hat{V}_h(S; \mathbf{a}_{n-1}) = V_{ex}(S);
\end{aligned}$$

As $M \to \infty$, there are many results in learning theory that justify the statement $\hat{V}_h(S; \mathbf{a}_{n-1}) \to \hat{V}_h(S; a^*_{n-1})$ which is the "best fit" to $V_h(S)$ within the parameterized class of functions (parameterized by $\mathbf{a}$). However, the best fit may not be such a good approximation unless the parameterized class of functions is carefully chosen so that it can implement the true function $V_h(S)$. Another further consideration is that it should be possible to find the best fit $\mathbf{a}$ efficiently.

Note that to compute $\hat{V}(S; \mathbf{a}_{n-1})$, all we need are x_j, y_j. Once we have $\hat{V}(S; \mathbf{a}_{n-1})$, we can obtain the optimal exercise threshold by solving the equation $\hat{V}_h = V_{ex}$. To obtain the y_j, we only need to be able to compute $V(S_j(n))$. This is the basis of the dynamic programming algorithm because we know $V(S; T)$.

Once we have $\hat{V}_h(S; \mathbf{a}_{n-1})$, there are actually two ways in which to proceed (remember we need $V(S, n-1)$). We will describe both

1. We can compute $V(S_j)$ using the relationship

$$V(S, n-1) = \max\{V_{ex}(S), \hat{V}_h(S; \mathbf{a}_{n-1})\}.$$

2. Remember that we only need to compute V on the particular S_j of a particular time step. Having computed $\pi^*(n-1)$ as the solution to $V_{ex}(S) = \hat{V}_h(S; \mathbf{a}_{n-1})$, for all $S_j \le \pi^*(n-1)$, we can set $V(S_j) = K - S_j$. For all $S_j > \pi^*(n-1)$, we set $V(S_j) = y_j$.

   Note that one does not actually have to compute $\pi^*(n-1)$. One can simply compare $V_{ex}(S_j)$ with $\hat{V}_h(S; \mathbf{a}_{n-1})$ and update accordiing to

$$V(S_j) = \begin{cases} K - S_j & V_{ex}(S_j) \ge \hat{V}_h(S; \mathbf{a}_{n-1}), \\ y_j & V_{ex}(S_j) < \hat{V}_h(S; \mathbf{a}_{n-1}). \end{cases}$$

Exercise 6.3

What is the difference between these two approaches?

The advantage of the first approach is that the values are less noisy as they resulted from a fit which "averages" information from all the data. However, they may be wrong depending on how good the function $\hat{V}_h$ is and the capabilities of the parameterized class used to approximate V_h. As a result, this error may propagate down the chain. The advantage of the second approach is that (modulo the determination of the threshold), the y_j above the threshold are *unbiased* estimates of the $V_h(S_j)$. This is important, because it means that the error is only in the determination of the theshold. Near the viscinity of the threshold, the values may be wrong, and should be $V_{ex}(S_j)$, but by definition, near the threshold, V_{ex} and V_h will be close, and so should the unbiased estimates. Thus the second approach only suffers from whatever small error there is in the optimal exercise threshold, and will propagate backward far less. Thus, we will continue with this second approach. It is instructive, however, to compare the prices that one would obtain from both approaches.

Once we have $V(S_j, n-1)$ and $\pi^*(n-1)$, we can use these values to obtain $V_h(S_j, n-2)$ and $\pi^*(n-2)$, and so on down to time step 0. Thus we construct the optimal exercise threshold function π^*. What about $V(S_j, 0)$? Since all the S_j are equal to S at time 0, there will be a problem with the regression. However, we do not need to do a regression at this time, we can simply compute the average value of holding by taking the average of $V(S_j, \Delta t)$. Then,

$$V(S, 0) = \max\{(K - S)^+, \frac{1}{M} \sum_{j=1}^{M} V(S_j, 1)\}.$$

Unfortunately, depending on the size of M, this final price will be optimistically biased (i.e., the price will be too high). Of course, this bias will decrease as M increases, however, it will still be there. To see why, lets consider a particular degenerate case of three paths and we use the quadratic polynomial form for $\hat{V}$. In this case, at every time step, we will be able to exactly fit the three examples. Let's now consider time step $n-1$. Since $\hat{V}_h$ will exactly fit the y_j, the fitting process is not averaging out any noise. It is simply memorizing the values of holding for these particular paths. As a result, we will hold and exercise later exactly for the paths where we should hold, and we will exercise those paths which we should not hold. It is as if we had the benefit of foresight and knew exactly what was going to happen in the future. Thus the $V(S, n-1)$ that we obtain will be positively biased. This positive bias will continue to time step $n-2$, i.e., we will only exercise the options we know will not make money. The end result is that $V(S,0)$ will have a serious positive bias.

Exercise 6.4

Consider the following algorithm for pricing the American put option using the M Monte Carlo paths p_j. Compute $V(S_j, n) = (K - S_j)^+$. Now for time step i, assume that $V(S_j, i+1)$ has been computed. We compute $V(S_j, i)$ by comparing the value of exercising with the value of holding on this path,

$$V(S_j, i) = \max\{(K - S_j)^+, e^{-r\Delta t} V(S_j, i+1)\}.$$

(a) Show that this algorithm computes the cashflow for each path p_j as

$$\max_t e^{-rt}(K - S_j(t))^+.$$

[Hint: Use induction.]

(b) What is wrong with this approach, for example will the price be too high or too low? Why?

The previous exercise is an extreme case of looking forward and optimally exercising on each path. The price of the resulting option will be extremely biased, similar to a lookback option with payoff $K - min_t S(t)$. The bias in the algorithm of the previous exercise is chronic in the sense that even as M increases, this bias will persist.

Fortunately, the LSM method will not suffer from this chronic bias. The resulting optimal threshold function from the LSM method may or may not be good, but we can say for sure, that the expected discounted cash flows resulting from *exercising with respect to this threshold function* must be at most the price of the put option. On the other hand, $V(S,0)$ as constructed by our LSM algorithm will be a serious over estimate of the expected cash flows that would result from exercising with respect to this exercise threshold function on another set of paths. Computing $V(S,0)$ using the same set of paths as were used to compute this optimal threshold function results in a positive bias because this exercise threshold function in some sense used some forward looking. The upshot of this discussion is that after obtaining the optimal exercise threshold function, one must generate a new set of Monte Carlo paths and re-price with respect to this exercise function. This second Monte Carlo is well defined since now the exercise function is known, and so the derivative cash flows are well defined. We can now summarize the entire algorithm. We show the version where the exercise threshold is obtained by solving at each time step for $V_{ex}(S) = \hat{V}_h(S)$.

```
 1: **Algorithm:** Pricing the American Put using LSM.
 2: Select $\Delta t$ and generate $M$ paths $\{p_j\}_{j=1}^{M}$.
 3: Initialize $\pi^*(n) = K$ and a vector $\mathbf{v}$ of size $M$ to $v_j = (K - S_j(n))^+$ for $j \in [1, M]$.
 4: **for** $i = n - 1$ to $0$ **do**
 5:     Construct the examples $\{x_j, y_j\}_{j=1}^{M}$: $x_j = S_j(i)$; $y_j = e^{-r\Delta t} v_j$.
 6:     Obtain the function $\hat{V}_h(S; \mathbf{a}_i)$, where $\mathbf{a}_i$ are the best fit parameters to the examples.
 7:     Set $\pi^*(i)$ to the solution of $\hat{V}_h(S; \mathbf{a}_i) = V_{ex}(S)$.        $//\pi^*(i) < K$
 8:     **for** $j = 0$ to $M$ **do**
 9:         **if** $S_j(i) \le \pi^*(i)$ **then**
10:             $v_j \leftarrow K - S_j$
11:         **else**
12:             $v_j \leftarrow y_j$                    //An alternative is: $v_j \leftarrow \hat{V}_h(S_j; \mathbf{a}_i)$.
13: Generate a new set of $L$ paths.
14: Compute the expected discounted cash flows w.r.t. $\pi^*$ and these new paths.
```

Note that in step 16, the new set of paths need not be computed with the same discretization as the first M paths. One approach is to use a few discretization steps to obtain the optimal exercise thresholds at these steps. A continuous exercise threshold function can be obtained using interpolation. The price can now be obtained using this exercise function and the new paths computed on a much finer discretization.

We now analyze the computational complexity of the algorithm. Let $\tau(M, \mathbf{a})$ be the computational complexity of obtaining the best $\mathbf{a}$ on M examples. This process needs to be run n times. Additionally after each fit, we need to compute the exercise threshold, which involves the solution of a (non-linear) equation. This can typically be done very efficiently, in some constant number of steps κ. In this case, the overall complexity is $O(n(\tau(M, \mathbf{a}) + \kappa))$. For d dimensional linear regression, $\tau(M, \mathbf{a}) = O(Md^2)$ where d is the dimension of the regression.

As the algorithm is written, it seems that we need to store M paths of size n each for a total memory requirement of nM. If we closely examine the algorithm, we see that at any time, the algorithm only processes two consecutive time steps. For example consider the first stage in the algorithm, processing the time steps $S_j(n-1)$ and $S_j(n)$. We therefore do not need the entire paths at any time, so suppose we first generate $S_j(n-1)$ and $S_j(n)$. After processing this stage, we now need $S_j(n-2)$, but we will never again need $S_j(n)$, so this memory can be reused to store $S_j(n-2)$ by reusing memory in this way, we see that all we need is $O(M)$ memory to store the paths at the time steps we are processing, and $O(n)$ memory to store the optimal exercise threshold function, for a total of $O(M + n)$.

The simplest thing to do, therefore, is to randomly generate the end points of the paths, $S_j(n)$, and then also randomly generate $S_j(n-1)$. Unfortunately, this is *wrong*. A key to the algorithm is the sequence on the path. $S_j(n)$ must have followed $S_j(n-1)$, i.e. the pairs $S_j(n-1), S_j(n)$ are not independent, they are correlated. We know how to generate paths forward in time. If we could somehow generate paths backward in time in such a way that reproduces the exact distribution of paths as if they were generated forward in time, then we would be in much better shape. As we first generate the end points, $S_j(n)$, and then for each end point, we can generate backward in time to get $S_j(n-1)$. When we are done with this stage, we generate backward from $S_j(n-1)$ to obtain $S_j(n-2)$, reusing the memory used for $S_j(n)$ to store $S_j(n-2)$, and so on. We now discuss backward path generation.

6.1.1 Backward Path Generation

It suffices to consider how to generate $\Delta\log S_j(i)$ for $i \in [1,n]$, from which we obtain $S_j(i) = S_j(0)e^{\Delta\log S_j(i)}$.

Recap: The log process. The process $\Delta\log S$ starts at 0 and follows a Brownian motion given by the dynamics

$$\Delta\log S = (r - \tfrac{1}{2}\sigma^2)dt + \sigma dW.$$

Thus the distribution of $\Delta\log S(t)$ has a distribution given by

$$\Delta\log S(t) \sim N((r - \tfrac{1}{2}\sigma^2)t, \sigma^2 t).$$

The Normal density function with mean μ and variance σ^2 is given by $N(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{1}{2\sigma^2}(x-\mu)^2)$

Thus, we can generate $\Delta\log S_j(n)$ for $j \in [1,M]$,

$$\Delta\log S_j(n) \sim N((r - \tfrac{1}{2}\sigma^2)T, \sigma^2 T).$$

To now generate *backward* in time from $\Delta\log S_j(n)$, we need $P[\Delta\log S_j(i-1)|\Delta\log S_j(i)]$ for $i = n, n-1, \ldots, 1$. Note that we know how to generate forward in time, i.e., we know $P[\Delta\log S_j(i)|\Delta\log S_j(i-1)]$, which follows from the more general forward law,

$$P[\Delta\log S(t+\Delta t)|\Delta\log S(t)] \sim N(\Delta\log S(t) + (r - \tfrac{1}{2}\sigma^2)\Delta t, \sigma^2\Delta t).$$

We would like a similar backward law for $P[\Delta\log S(t-\Delta t)|\Delta\log S(t)]$. The next exercise develops this backward law.

Exercise 6.5

Let $u_t = \Delta\log S(t)$ and let $u_{t-\Delta t} = \Delta\log S(t-\Delta t)$. Show that

$$P[u_{t-\Delta t}|u_t] = \frac{P[u_t|u_{t-\Delta t}]P[u_{t-\Delta t}]}{P[u_t]},$$

and hence show that

$$\begin{aligned} P[u_{t-\Delta t}|u_t] &= \frac{1}{2\pi\sigma^2\Delta t(1-\frac{\Delta t}{t})}\exp\left(-\frac{\left(u_{t-\Delta t} - u_t(1-\frac{\Delta t}{t})\right)^2}{2\sigma^2\Delta t(1-\frac{\Delta t}{t})}\right), \\ &\sim N(u_t(1-\tfrac{\Delta t}{t}), \sigma^2\Delta t(1-\tfrac{\Delta t}{t})) \end{aligned}$$

Note that the mean is future value decreased by a factor $1 - \frac{\Delta t}{t}$, and the variance also decreases by the same factor. Note also that there is no dependence on r in the backward law. Where has the r dependence gone, has it totally disappeared – i.e., if we can generate the paths backward without r then we should be able to generate paths forward without r?

Do the backward paths converge to 0 – this is necessary if the algorithm is correct, as the forward paths would begin at 0.

We can now restate the full algorithm to take backward path generation into account, which is more memory efficient.

```
1: **Algorithm:** Pricing the American Put using LSM and backward path generation.
2: Select $\Delta t$ and generate $M$ prices $\{x_j\}_{j=1}^M$ at time step $n$, corresponding to the paths
   $\{p_j\}_{j=1}^M$. $x_j \sim N((r - \frac{1}{2}\sigma^2)T, \sigma^2 T)$.
3: Initialize $\pi^*(n) = K$ and a vector $\mathbf{v}$ of size $M$ to $v_j = (K - Se^{x_j})^+$ for $j \in [1, M]$.
4: for $i = n - 1$ to $0$ do
5:     for $j = 1$ to $M$ do
6:         $x_j \leftarrow N\left(x_j(1 - \frac{1}{i+1}), \sigma^2 \Delta t(1 - \frac{1}{i+1})\right)$.
7:         $v_j \leftarrow e^{-r\Delta t} v_j$.
8:     Obtain $\hat{V}_h(S; \mathbf{a}_i)$ by fitting $\mathbf{a}_i$ to the examples $\{Se^{x_j}, v_j\}_{j=1}^M$:
9:     Set $\pi^*(i)$ to the solution of $\hat{V}_h(S; \mathbf{a}_i) = V_{ex}(S)$.        $//\pi^*(i) < K$
10:    for $j = 0$ to $M$ do
11:        if $x_j \leq \log \frac{\pi^*(i)}{S}$ then
12:            $v_j \leftarrow (K - Se^{x_j})^+$
13: Generate a new set of $L$ paths.
14: Compute the expected discounted cash flows w.r.t. $\pi^*$ and these new paths.
```

Note that there is a slight degeneracy as the algorithm above gets closer to $i = 0$. At $i = 0$, all the x_j are 0, and so all the examples have Se^{x_j} equal to the same value. This means that the fitting will be degenerate, and the way to compute $V_h(S, 0)$ is to simply average. For this reason, since the main goal is to obtain the exercise function in the first part of the algorithm, it is probably better to generate the paths all starting from different S's. Steps [13] and [14] for pricing must all generate paths from the same initial price though. This is easy to incorporate into the algorithm and is left as an exercise.

Exercise 6.6

Update the algorithm to compute the optimal exercise threshold function when the initial paths are generated from *different starting prices*.

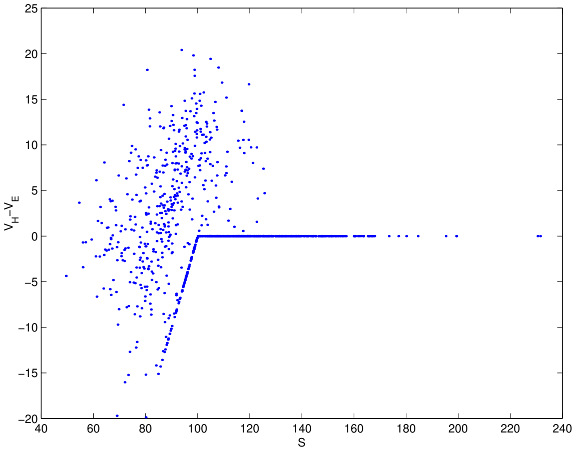
6.2 Optimal Thresholds – No Least Squares

Examining the algorithm in the previous section, we see that the main purpose served by the least squares fitting (steps [6], [7]) is to obtain the exercise threshold. We now describe how to avoid the need for the least squares fit by giving an efficient non-parametric algorithm for obtaining the *optimal* threshold. The entire algorithm will run as in the previous section, with the exception that the least square fit is not needed, and the exercise threshold $\pi^*(i)$ is set to its optimal value.

To see how this will work, observe that the optimal threshold π^* satisfies

$$f(\pi^*) = V_h(\pi^*) - V_{ex}(\pi^*) = 0.$$

A more useful observation is that for $S < \pi^*$, $f(S) < 0$ as it is better to exercise, and for $S > \pi^*$, $f(S) > 0$. We have the samples $\{S_j(i-1)\}$ and $\{e^{-r\Delta t}V(S_j(i))$, which are unbiased realizations of V_h. We thus have noisy realizations of $f(S)$, as shown in the figure below,



While there is definitely a trend, going from negative to positive, it is not at all clear where to place the threshold, which is why we first need to formalize the problem. We have the stock prices $S_j(i-1)$, and we define $f_j = e^{-r\Delta t}V(S_j(i)) - \max\{K - S_j(i-1), 0\}$. The plot above is a plot of f_j versus S_j.

It suffices to consider the possible thresholds 0, and $\{S_j\}$, with the convention that if the threshold is S_k then the assertion is that $f(S) < 0$ for all $S \leq S_k$. Let θ denote the threshold, then $\theta \in \{0, S_1, \ldots, S_M\}$. A point is an error if it is left of the threshold and has value > 0, or it is right of the threshold and has value < 0. Thus, for a particular threshold θ, we define the error of that threshold as

$$\begin{aligned}\mathcal{E}(\theta) &= \sum_{S_j \leq \theta} \max(f_j, 0) - \sum_{S_j > \theta} \min(f_j, 0), \\ &= \sum_{S_j \leq \theta} f_j^+ - \sum_{S_j > \theta} f_j^-.\end{aligned}$$

The problem we would like to solve efficiently is to find the optimal threshold θ, the one which minimizes $\mathcal{E}(\theta)$. If we assume that the S_j have been pre-sorted, then the following observations lead to a linear time algorithm to solve this problem. Start with $\theta = 0$. In this case, $\mathcal{E}(0) = -\sum_{S_j} f_j^-$. We now process points from left to right in increasing order and update $\mathcal{E}$ by adding f_j. Specifically, we claim that

$$\mathcal{E}(S_j) = \mathcal{E}(0) + \sum_{k=1}^{j} f_j.$$

Exercise 6.7

Prove the above claim, and use it to develop a linear time algorithm to find the optimal threshold (assuming that the S_j are pre-sorted).

[Hint: Use induction. First show that $\mathcal{E}(S_j) = \mathcal{E}(S_{j-1}) + f_j$.]

The running time of this optimal threshold algorithm will be dominated by the sort operation, which is $O(M \log M)$. The remainder of the algorithm is exactly as in the previous section, i.e., the optimal threshold function needs to be run for each time step, yielding a total computational complexity of $O(nM \log M)$. Thus the computational trade off is a factor $\log M$ versus the d^2 for the regression. The advantage is that no parameteric form for V_h need be assumed, and more importantly, the optimal thresholds are obtained.

7 Pricing By Optimizing Bounds

8 Optimal Exercise from Pricing

9 Dividend Paying Stocks and Put-Call Parity