Computational Study of Pricing American Options Using a New Dynamic Programming Approach

A Project Report For MS534 – Application of Management Science Methods

Yuanlei Zhang
November 25, 2005

1 Introduction

This project report investigates the problem of finding the minimum initial cost of replicating a terminal payoff in an imperfect market with transactions costs and trading restrictions. Since finding this minimum initial cost is equivalent to pricing an option, this problem can also be looked as an instance of the classic option-pricing problem.

The organization of this project report is as following: Section 2 is a literature review on various approaches of pricing options (mostly American options) with transactions costs; Section 3 describes the dynamic programming model developed by Dr. Edirisinghe as an efficient approach to study the option-pricing problem and discusses several extensions to the basic DP model; In Section 4, computational experiments on the Java implementation of the dynamic programming model (with extensions) are described, results are presented and analyses are made; Section 5 concludes this project report. Implementation details can be found in the appendices.

2 Literature Review

In this section, various approaches of pricing options (mostly American options) with transactions costs are reviewed. In general, these approaches can be grouped into two major categories based on their model assumptions on the time process:

- Continuous time process
  M.A.H Dempster and J.P. Hutton presented a continuous time model in their paper “Pricing American Stock Options by Linear Programming” published on August 24, 1999, where the numerical solution of finite difference approximations to American option pricing problems is investigated using the simplex solution of a linear programming formulation. They adopted well-known results for the American put option and modeled the stock price under the equivalent martingale (risk neutral) measure as:

\[
\frac{dS(t)}{S(t)} = r dt + \sigma d\tilde{W}(t), t \in [0,T],
\]
where $S(0) > 0$, $\sigma > 0$ is the constant volatility of the stock, and $\tilde{W}$ is a Wiener process under this measure. In order to develop an appropriate linear program, they first studied equivalent formulations of an American option problem as a free boundary problem, a linear complementarity problem and a variational inequality, and derived an abstract linear program. Then they demonstrated that through discretising space and time by standard finite differences, the abstract linear program becomes an ordinary linear program and is solvable by simplex algorithm.

In the paper “American Option Pricing with Transaction Costs” published on October 29, 2003, Valeri. I. Zakamouline developed another continuous-time model to find the investor’s reservation option prices and corresponding early exercise policies of American style options in the market with proportional transaction costs, using the CARA utility based approach. By assuming a Geometric Brownian Motion price process, where $dS_t = \mu S_t dt + \sigma S_t dB_t$, a new formulation of the problem in terms of quasi-variational HJB inequalities and an original discretization schemes based on the new formulation were suggested and implemented, for computing reservation purchase and write prices of American-style put options and reservation purchase prices of American-style call options. As a result, Valeri found out that in the market with transaction costs the holder of an American-style option exercises this option earlier as compared to the case with no transaction costs. The higher level the transaction costs is, or the higher risk avers the option holder is, the earlier an American option is exercised.

A third continuous time model was proposed by Richard H. Stockbridge in his paper “Characterizing Option Prices by Linear Programs” to illustrate how the pricing of options can be characterized in terms of the solution of a linear program. The linear program he formulated takes as its variables the occupation measures corresponding to the evolution of the asset prices and auxiliary processes as needed. A formulation of the model in terms of general stochastic processes is presented, along with the linear programming formulation for the European and American options under the Geometric Brownian Motion model. However, the paper solely considers pricing of options in a complete market.

- Discrete time process

A discrete time model was introduced in the paper “Stochastic Dominance Bounds on American Option Prices in Markets with Frictions” published by George M. Constantinides and Stylianos Perrakis on June 15, 2004. This paper applies stochastic dominance arguments to derive bounds on the prices of American options on either an index or index futures in the presence of transaction costs. The bounds are derived by applying the weak notion of stochastic dominance and are independent of the trader’s particular initial portfolio and time-separable utility function. The model they introduced considers two primary securities: a riskless bond and a risky stock, along
with a cash-settled American call or put option written on the stock. Before deriving
the bounds, they made two important assumptions about the price process:
1. The dividend yield parameters \( \{ \gamma_t \}_{t=0, \ldots, T} \) satisfy the condition \( 0 \leq \gamma_t < 1 \) and are
deterministic and known to the trader at time 0
2. The rates of return are independently distributed with conditional mean return
\[
R_t = E \left[ (1 + \gamma_{t+1}) \frac{S_{t+1}}{S_t} \right] \text{ known to the trader at time 0}
\]
Following these assumptions, stochastic dominance bounds on the reservation
purchase and write prices of American put and call options are derived. They found
out that the bounds on the reservation purchase price of American puts and the
reservation write price of American calls are tight and inferred that plausible
proportional transaction costs cannot account for the volatility smile in an otherwise
Black-Scholes environment.

Krzysztof Tokarz and Tomasz Zastawniak presented another discrete time model in
their paper “Dynamic Programming Algorithms for the Ask and Bid Prices of
American Options under Small Proportional Transaction Costs” published on August
24, 2004, which is a dynamic programming model for computing the ask and bid
prices of American contingent claims according to a binary tree price process in the
presence of small proportional transaction costs. Along with the pricing algorithms,
iterative procedures for computing optimal hedging strategies for the writer as well as
for the buyer of an American option are provided. The authors developed the pricing
algorithms based on backward induction, which involves the solution of an
optimization problem at each tree node. However, this paper only considers
contingent claims with cash delivery and assumes implicitly that the option buyer or
seller has no position in the underlying asset initially.

Yet another effort of developing discrete time model for the option-pricing problem
was shown in the recent paper “On the Hedging of American Options in Discrete
Time Markets with Proportional Transaction Costs” published in May 2005 by Bruno
BOUCHARD and Emmanuel TEMAM, where a general discrete time financial
market with proportional transaction costs is considered and a dual formulation for the
set of initial endowments that allow to super-hedge some American claim is provided.
In developing the dual formulation, the authors adopted the general framework of
C-valued processes as well as the price process where risky assets evolve on a finite
dimensional tree. The work they presented is an extension to both the existing
super-replication theorem for European contingent claims and the existing model with
constant transaction costs.
3 Dynamic Programming Approach

In the paper “Optimal Replication of Options With Transactions Costs and Trading Restrictions” published by Dr. Edirisinghe in 1993, a dynamic programming model was proposed, as an efficient approach to study the classic option-pricing problem, under an imperfect market with transactions costs and trading restrictions. Before getting into the details of the DP model, a general discrete time model of the option-pricing problem is presented below, which was also developed by Dr. Edirisinghe in his 1993 paper.

Note: Variable definitions are the same as in the original paper and omitted here for simplicity.

3.1 The general discrete time model

The option-pricing problem with transactions costs and trading restrictions can be formulated as:

$$\min \bar{\alpha}(0) S(0) + \beta(0)$$

Subject to:

1) The self-financing constraints (including both proportional and fixed transactions costs):

$$[\alpha(t, j) - \alpha(t-1, i)] S(t, j) + \beta(t, j) - \alpha(t-1, i), t = 1, \ldots, T - 1$$

$$\forall j \in F_r, j \subseteq i, i \in F_{t-1}$$

2) The lot size restrictions:

$$\alpha(t, j) - \alpha(t-1, i) = M \delta_{\alpha} \quad \text{and} \quad \beta(t, j) - \beta(t-1, i) = M \delta_{\beta}$$

3) The position limits on trading:

$$\alpha_{\min} \leq \alpha(t, j) \leq \alpha_{\max} \quad \text{and} \quad \beta_{\min} \leq \beta(t, j) \leq \beta_{\max}$$

4) The terminal constraints:

$$\alpha(T-1, i) S(T, j) + \beta(T-1, i) \geq C(T, j), \quad \forall j \in F_r, j \subseteq i, i \in F_{T-1}$$

Given the nature of the self-financing constraints, this problem formulated as above is a nonlinear optimization problem. Also, the lot size constraints and the fixed trading costs constraints introduce integer programming to this problem.
An equivalent linear programming formulation of the above nonlinear model can be developed to solve the option-pricing problem exactly. However, solving the exact LP model is computationally difficult for large values of the revision frequency $T$, because the optimal trading strategy depends on the entire history of the stock price and the number of constraints and decision variables in the LP model grows exponentially in $T$. Moreover, fixed transactions costs and lot size restrictions introduce integer constraints into the LP model, which makes solving the problem using the LP approach computationally infeasible for large scale problems.

As a way of overcoming the inefficiency of the linear programming model, the basic dynamic programming model developed by Dr. Edirisinghe in 1993 is presented next, which is computationally efficient and accounts for integer constraints appropriately.

Note: Variable definitions are the same as in the original paper and omitted here for simplicity.

3.2 The basic DP model

The basic DP model is a two-stage optimization model. In the first stage, a self-financing trading strategy $\{\alpha, \beta\}$ is determined, which minimizes the expected value of terminal deficits, for a given level of initial wealth $w$:

$$J_0(w) \equiv \min_{\{\alpha, \beta\}} E_0^p \max [C(T) - \alpha(T-1)S(T) - \beta(T-1), 0],$$

such that

$$\alpha(0)S(0) + \beta(0) = w,$$

and subject to the self-financing constraints, the lot size constraints and the position limits on trading.

In the second stage, the smallest initial wealth $w^*$ is found, such that the minimized expected deficit at $T$ is 0, that is, $J_0(w^*) = 0$:

$$w^* \equiv \inf_{w \in \mathcal{R}} \{w : J_0(w) = 0\}$$

The first stage problem can be solved via a backward dynamic recursion. The definition of the value function is recursive in nature, as shown below:

$$J_t^*(a, b, S(t, j)) \equiv \min_{\alpha(t, j), \beta(t, j)} \sum_{k=j} \frac{1}{2} J_{t+1}^*(\alpha(t, j), \beta(t, j), S(t+1, k)),$$

And the boundary condition at $T$ is:

$$J_T^*(a, b, S(T, j)) \equiv \max [C(T, S(T, j)) - aS(T, j) - b, 0], \forall j \in F_T.$$
Evaluations of the above value function are subject to the following constraints:

1) The self-financing constraints (including both proportional and fixed trading costs):
\[
[\alpha(t, j) - a]S(t, j) + \frac{1}{2}[\alpha(t, j) - a]S(t, j) + \frac{1}{2}[\beta(t, j) - b] + \phi(t, j) \leq 0
\]

2) The lot size constraints:
\[
\alpha(t, j) - a = M\delta_\alpha \quad \text{and} \quad \beta(t, j) - b = M\delta_\beta
\]

3) The position limits on trading:
\[
\alpha_{\min} \leq \alpha(t, j) \leq \alpha_{\max} \quad \text{and} \quad \beta_{\min} \leq \beta(t, j) \leq \beta_{\max}
\]

The second stage problem can be solved using a simple univariate search, which searches for the minimum initial wealth \( w^* = \min_{\{a, b\}} aS(0) + b \) that satisfies \( J_0^*(a, b, S(0)) = 0 \).

The computational efficiency of the DP approach comes from the fact that, for a given grid of \((a, b)\), the number of times the value function is computed is quadratic, not exponential, in \( T \). This is because the number of distinct stock prices on the event tree grows only quadratically in \( T \), and the value function is a function only of the stock price and not of the path followed to get there.

Another advantage of the DP approach is that the various additional constraints of fixed transactions costs, lot size restrictions and position limits can be accounted for as “easily” as the self-financing constraints, without introducing too much extra computational burdens.

The basic DP model can be further extended to approximate the stock price movement more accurately as a multinomial tree, and to simulate American options, either with or without a fixed dividend payment.

### 3.3 Multinomial stock price movement

In the basic DP model described above, a binomial tree is used to approximate the stock price movement. Although a binomial model is sufficient in some situations, more accurate approximation can be achieved through introducing the multinomial stock price movement. Extending the basic model to incorporate multinomial stock price movement only needs slight modifications to the recursive evaluations of the value function \( J^*_t(\cdot) \), as shown below (for a \( N \)-nomial model):

\[
J^*_t(a, b, S(t, j)) \equiv \min_{a(t, j), b(t, j)} \frac{1}{N} \sum_{k=J_{t+1}} J^*_{t+1}(\alpha(t, j), \beta(t, j), S(t+1, k))
\]
where \( j_{d_1}, j_{d_2}, \ldots, j_{d_N} \in F_{i+1} \) are \( N \) successor elements of the element \( j \in F_i \), and
\[
S(t + 1, j_{d_i}) = d_i S(t, j), i = 1, \ldots, N.
\]

The extended DP model with multinomial stock price movement will achieve higher accuracy at the cost of more computational complexity.

3.4 American options
The basic DP model can also be extended to simulate American options. The essence of an American option is that the option can be exercised at any time up to the expiration date, not like a European option, where the option can be exercised only on the expiration date. Because of American option’s early exercise possibility, deficits have to be introduced not only at terminal nodes, but also at intermediate nodes. Therefore, for any intermediate node, a nodal deficit should be considered in addition to the expected deficit from the future, in order to evaluate the actual deficit (the value function \( J^*_t(.) \)) at that node. Nodal deficits can be defined in the same manner as terminal deficits:
\[
DFT_t(a, b, S(t, j)) \equiv \max[C(t, S(t, j)) - aS(t, j) - b, 0], \forall j \in F_i
\]

Also, let \( EJ_t(a, b, S(t, j)) \) denote the expected deficit from the future, where
\[
EJ_t(a, b, S(t, j)) \equiv \min_{\alpha(t, j), \beta(t, j)} \sum_{k = j_{d_1}, \ldots, j_{d_N}} J^*_{t+1}(\alpha(t, j), \beta(t, j), S(t + 1, k))
\]

The evaluation of the value function \( J^*_t(.) \) should then be revised as:
\[
J^*_t(a, b, S(t, j)) = \max\{DFT_t(a, b, S(t, j)), EJ_t(a, b, S(t, j))\}
\]

As seen from above, the actual deficit at any particular node will be the maximum of the nodal deficit (if the option is exercised at that node) and the expected deficit from the future (if the option is not exercised at that node).

3.5 American options with a fixed dividend payment
The previous discussion about American options does not consider the dividend payment that is usually associated with holding a stock. To simulate the real-world situations more closely, a fixed dividend payment can be introduced to push further the analysis of American options.

The payment of dividend can be modeled as a real-valued function of the number of stocks held at hand:
\[
DIV_t(a) \equiv C_t * a
\]
where $C_t$ is the coefficient of dividend payment at time period $t$, which is a fixed value for a given time period.

With fixed dividend payment introduced to American options, the evaluation of the value function $J^*_t(.)$ should then be revised as:

$$J^*_t(a,b,S(t,j)) = \max\{DFT_t(a,b,S(t,j)) + DIV_t(a), EJ_t(a,b,S(t,j))\}$$

where the fixed dividend payment is considered as an adjustment to the nodal deficit.

4 Computational Experiments and Results

The dynamic programming model with extensions is implemented in Java as a standalone computer program (see Appendix A, B and C for implementation details). Experiments are done to examine the effects of multinomial stock price movement, American execution with and without a fixed dividend payment on pricing options. By comparing the results of these experiments, valuable findings are identified and analyses are performed accordingly to the best knowledge of the author.

4.1 Common experiment conditions

There are several configurable input parameters to the program, and the following of them are fixed at the specified level for all the experiments (see Appendix D for explanations on the input parameters):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specified Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>8</td>
</tr>
<tr>
<td>$A_l$</td>
<td>-1.0</td>
</tr>
<tr>
<td>$A_h$</td>
<td>+1.0</td>
</tr>
<tr>
<td>$A_n$</td>
<td>200</td>
</tr>
<tr>
<td>$B_l$</td>
<td>-1.0</td>
</tr>
<tr>
<td>$B_h$</td>
<td>+1.0</td>
</tr>
<tr>
<td>$B_n$</td>
<td>200</td>
</tr>
<tr>
<td>$K$</td>
<td>1.0</td>
</tr>
<tr>
<td>$S_0$</td>
<td>1.0</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>0.10</td>
</tr>
<tr>
<td>$\Delta A$</td>
<td>0.1</td>
</tr>
<tr>
<td>$\Delta B$</td>
<td>0.03</td>
</tr>
</tbody>
</table>
4.2 Verification of the shape of the $J_0(w)$ function

Recall from section 3.1, the minimized expected terminal deficit $J_0(w)$ is a function of the initial wealth $w$, as defined below:

$$J_0(w) \equiv \min_{\alpha, \beta} E_0^p \max[C(T) - \alpha(T-1)S(T) - \beta(T-1), 0]$$

Theoretically, the $J_0(w)$ function is a non-increasing convex function of $w$. Before any other experiments are carried out, an initial experiment is done to verify the shape of the $J_0(w)$ function, which also serves the purpose of examining the correctness of the implementation program.

By setting the rest of the input parameters as $D=2$, $C=[0, 0, 0, 0, 0, 0, 0, 0, 0]$, $\theta=0.01$, $\phi=0.00$, $\text{americanOption}=true$ and running the experiment, data points for plotting the $J_0(w)$ function are obtained. Plotting these data points in Microsoft Excel, the following figure is generated:

The above figure verifies the non-increasing convex property of the $J_0(w)$ function and provides evidence to the correctness of the implementation program.
4.3 Comparison of trinomial case to binomial case

To examine the effect of multinomial stock price movement on pricing options, the trinomial case is compared to the binomial case. Experiment conditions and results are shown below:

<table>
<thead>
<tr>
<th>americanOption= false, phi=0.00</th>
<th>Binomial (D=2)</th>
<th>Trinomial (D=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$w^*$</td>
<td>$\alpha(0)$</td>
</tr>
<tr>
<td>theta=0.01</td>
<td>0.11</td>
<td>0.42</td>
</tr>
<tr>
<td>theta=0.03</td>
<td>0.12</td>
<td>0.52</td>
</tr>
<tr>
<td>theta=0.05</td>
<td>0.13</td>
<td>0.50</td>
</tr>
<tr>
<td>theta=0.07</td>
<td>0.13</td>
<td>0.54</td>
</tr>
<tr>
<td>theta=0.09</td>
<td>0.14</td>
<td>0.53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>americanOption= false, theta=0.01</th>
<th>Binomial (D=2)</th>
<th>Trinomial (D=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$w^*$</td>
<td>$\alpha(0)$</td>
</tr>
<tr>
<td>phi=0.00</td>
<td>0.11</td>
<td>0.42</td>
</tr>
<tr>
<td>phi=0.05</td>
<td>0.15</td>
<td>0.54</td>
</tr>
<tr>
<td>phi=0.10</td>
<td>0.15</td>
<td>0.54</td>
</tr>
<tr>
<td>phi=0.15</td>
<td>0.15</td>
<td>0.54</td>
</tr>
<tr>
<td>phi=0.25</td>
<td>0.15</td>
<td>0.54</td>
</tr>
</tbody>
</table>

The results show that adopting multinomial stock price movement in the dynamic programming model will generally produce better minimums. As seen from the above tables, smaller $w^*$'s are achieved in the trinomial case than in the binomial case, under all variations of the proportional as well as the fixed transaction cost factor. The reason is that the multinomial stock price movement is a more accurate approximation to the real-world situation, although it’s computationally more complex.
## 4.4 Comparison of American options to European options

To examine the effect of American execution on pricing options, American options without a fixed dividend payment are compared to European options. Experiment conditions and results are shown below:

<table>
<thead>
<tr>
<th>D=2, phi=0.00</th>
<th>European Options (americanOption=false)</th>
<th>American Options (americanOption=true)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$w^*$</td>
<td>$\alpha(0)$</td>
</tr>
<tr>
<td>theta=0.01</td>
<td>0.11</td>
<td>0.42</td>
</tr>
<tr>
<td>theta=0.03</td>
<td>0.12</td>
<td>0.52</td>
</tr>
<tr>
<td>theta=0.05</td>
<td>0.13</td>
<td>0.50</td>
</tr>
<tr>
<td>theta=0.07</td>
<td>0.13</td>
<td>0.54</td>
</tr>
<tr>
<td>theta=0.09</td>
<td>0.14</td>
<td>0.53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D=2, theta=0.01</th>
<th>European Options (americanOption=false)</th>
<th>American Options (americanOption=true)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$w^*$</td>
<td>$\alpha(0)$</td>
</tr>
<tr>
<td>phi=0.00</td>
<td>0.11</td>
<td>0.42</td>
</tr>
<tr>
<td>phi=0.05</td>
<td>0.15</td>
<td>0.54</td>
</tr>
<tr>
<td>phi=0.10</td>
<td>0.15</td>
<td>0.54</td>
</tr>
<tr>
<td>phi=0.15</td>
<td>0.15</td>
<td>0.54</td>
</tr>
<tr>
<td>phi=0.25</td>
<td>0.15</td>
<td>0.54</td>
</tr>
</tbody>
</table>

The above results do not show any difference between American options and European options, since the same $w^*$'s are achieved in the American options case as in the European options case, under all variations of the proportional as well as the fixed transaction cost factor. The underlying reason for this finding may be complicated and beyond the scope of the author's knowledge, but it is suspected that American options really are not different from European options in the absence of dividend payment. In other words, when no dividends are paid out, it is never optimal to exercise the option earlier than the expiration date.
4.5 Comparison of American options with and without dividend payment

To examine the effect of dividend payment on pricing American options, American options with and without a fixed dividend payment are compared. Experiment conditions and results are shown below:

<table>
<thead>
<tr>
<th></th>
<th>Without Dividend</th>
<th>With Dividend</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(C=[0, 0, 0, 0, 0, 0, 0, 0])</td>
<td>(C=[0, 0, 0, 0, 0.05, 0, 0, 0])</td>
</tr>
<tr>
<td>w*</td>
<td>w*</td>
<td></td>
</tr>
<tr>
<td>alpha(0)</td>
<td>alpha(0)</td>
<td>alpha(0)</td>
</tr>
<tr>
<td>beta(0)</td>
<td>beta(0)</td>
<td></td>
</tr>
<tr>
<td>theta=0.01</td>
<td>0.11</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>0.42</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td>-0.31</td>
<td>-0.37</td>
</tr>
<tr>
<td>theta=0.03</td>
<td>0.12</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>0.52</td>
<td>0.70</td>
</tr>
<tr>
<td></td>
<td>-0.40</td>
<td>-0.52</td>
</tr>
<tr>
<td>theta=0.05</td>
<td>0.13</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>-0.37</td>
<td>-0.58</td>
</tr>
<tr>
<td>theta=0.07</td>
<td>0.13</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>0.54</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>-0.41</td>
<td>-0.24</td>
</tr>
<tr>
<td>theta=0.09</td>
<td>0.14</td>
<td>0.29</td>
</tr>
<tr>
<td></td>
<td>0.53</td>
<td>-0.01</td>
</tr>
<tr>
<td></td>
<td>-0.39</td>
<td>0.30</td>
</tr>
</tbody>
</table>

The results show that American options with a fixed dividend payment are generally more expensive than those without dividend payment, and therefore are more expensive than European options (since American options without dividend payment are not different from European options, as observed in the previous section 4.4). Under all variations of the proportional as well as the fixed transaction cost factor, greater $w^{*}$'s are obtained in the case with dividend than in the case without dividend.

This finding can be explained by the introduction of more constraints in the evaluation of American options with dividend payment, which shrinks the feasible region and in turn worsens the minimum. Economically, this also makes sense, as American options offer the buyers more flexibility in execution and therefore should be more expensive than European options.
5 Conclusions

The dynamic programming model developed by Dr. Edirisinghe is a valuable and effective approach of studying the classic option-pricing problem, under an imperfect market with transactions costs and trading restrictions. This DP model has the following advantages:

- It is computationally efficient, as the number of times the value function is computed is quadratic (not exponential) in the revision frequency $T$, for a given grid of stock and bond positions;
- Various additional integer constraints such as fixed transactions costs, lot size restrictions and position limits can be accounted for appropriately and as “easily” as the self-financing constraints;
- Extensions to the DP model such as approximating the stock price movement by a multinomial tree and simulating American options with and without dividend payment are relatively easy to make.

By examining the results obtained from the computational studies, it is suspected that there are no differences between American options and European options, when dividend payment is left out of consideration. What this means is that when no dividends are paid out on the stocks, it is never optimal to exercise the option early before reaching the expiration date. However, if dividend payment is considered, American options will be more expensive than European options, which makes sense economically because American options offer the buyers more flexibility in execution and therefore should be more expensive than European options.

Acknowledgement

I would like to thank Dr. Edirisinghe for offering me such a great opportunity to practice and learn, for guiding me through the entire process of this computational study, and for being so patient and thoughtful when helps are needed. I enjoyed working on this project very much and I really learned a lot.
Appendix A: Data structure

The data structure discussed here is the data structure used to store the evaluated values of the value function $J_t^*(a,b,S(t,j))$. Because there are four parameters $t,a,b,S(t,j)$ to the value function, the evaluated values have to be stored in a four-dimensional array that is indexed by these four parameters. The data structure developed is a structural representation of this four-dimensional array. The basic building block of the data structure is a DPState class, as illustrated below:

The way of using DPState to build the complete data structure is illustrated in the following diagram:

By constructing the data structure this way, the four-dimensional array of evaluated values is indexed in the order of $t,S(t,j),a,b$, which is computationally efficient.
Appendix B: Approximations

The dynamic programming model is by its nature an approximation to the exact linear programming model. Therefore, implementing the DP model requires making several important approximations as explained below. The accuracy of the DP model will depend on how these approximations are carried out and the level of precision they can achieve.

B1 Bounding and making discrete the state space for $a$ and $b$
The state space for stock positions $a$ and bond positions $b$ is first normalized and bounded as $-1 \leq a \leq +1, -1 \leq b \leq +1$. This state space is then made discrete by dividing the range $[-1, +1]$ into $N_s$ equal-length segments, where the level of precision (as determined by $N_s$) is configurable.

B2 Simulating multinomial stock price movement
The effect of multinomial stock price movement can be simulated by choosing appropriate values for $d_1, d_2, \ldots, d_N$, where $d_i = \exp\left(\frac{\sigma}{\sqrt{T}}\right)$ and $d_N = \frac{1}{d_1}$. The rest of the $d’s$ can be determined by calculating the values of the inner points that divide the range $[d_1, d_N]$ into $N-1$ segments with equal-length $S_s = \frac{d_1 - d_N}{N-1}$.

B3 Making discrete the state space for stock prices
When the stock price movement is binomial, the number of distinct stock prices at time period $t$ grows linearly with $t$, and we can easily enumerate all distinct stock prices. However, when the stock price movement becomes multinomial, the number of distinct stock prices at time period $t$ grows a lot faster with $t$, and it’s no longer practical to enumerate exactly all distinct stock prices. In these cases, approximations are necessary. For a given time period $t$, the maximum stock price is $d_1^t * S_0$ and the minimum stock price is $d_N^t * S_0$. Therefore, the state space for stock prices can be made discrete by dividing the range $[d_1^t * S_0, d_N^t * S_0]$ into equal-length segments, where the length of each segment is a fixed value $S_s * S_0$. 
Appendix C: Computational tricks

Although implementing the dynamic programming model is straightforward, given that the underlying mathematical logics are fully understood, there are still several computational tricks worth mentioning.

C1 Converting indices to actual values

Mathematically, the value function matrix \( \text{funcJ} \) (as illustrated in Appendix A) is indexed by the actual values of stock and bond positions \( a \) and \( b \), and these values are real numbers ranging from –1 to +1. However, indices \( a_i \) and \( b_i \) of a matrix in the implementing programming language are required to be integer numbers starting from 0. Therefore, index-to-actual-value conversions need to be performed as the following:

\[
a = a_i + a_i \times (a_a - a_i) / N_s
\]

\[
b = b_i + b_i \times (b_h - b_i) / N_s
\]

where \( a_i = b_i = -1 \), \( a_h = b_h = +1 \) and \( N_s \) is the number of segments used to make discrete the state space for \( a \) and \( b \).

Furthermore, these conversions can be computed incrementally, which has been proved to be much more efficient than computed directly.

C2 Linear interpolation of the evaluated values of \( J^*(.) \)

Recall from section 3.2, the evaluation of the value function is defined as:

\[
J^*(a, b, S(t, j)) = \min_{a(t, j), \beta(t, j)} \sum_{k=1}^{N_s} J^*_{t+1}(\alpha(t, j), \beta(t, j), S(t+1, k))
\]

Because the state space for stock prices is made discrete through approximation, when evaluating the above value function \( J^*(.) \), the stock price \( S(t+1, k) \) of a successor \( k \) may not correspond exactly to a single discretization point of stock prices and may instead fall between two discretization points \( S(t+1, u) \) and \( S(t+1, d) \). In this case, \( J^*_{t+1}(\alpha(t, j), \beta(t, j), S(t+1, k)) \) is not defined and therefore has to be linearly interpolated by \( J^*_{t+1}(\alpha(t, j), \beta(t, j), S(t+1, u)) \) and \( J^*_{t+1}(\alpha(t, j), \beta(t, j), S(t+1, d)) \).

The following formula is used for linear interpolation:
\[ J^*_t+1(\alpha(t, j), \beta(t, j), S(t+1,k)) = J^*_t+1(\alpha(t, j), \beta(t, j), S(t+1,d)) + \]
\[
(J^*_t+1(\alpha(t, j), \beta(t, j), S(t+1,u)) - J^*_t+1(\alpha(t, j), \beta(t, j), S(t+1,d))) \cdot \frac{S(t+1,k) - S(t+1,d)}{S(t+1,u) - S(t+1,d)}
\]

C3 Storing only the evaluated values of two successive time periods

As determined by its recursive definition, the value function \( J^*_t(.) \) has to be evaluated backwardly starting from the end of the planning horizon, and the evaluated values of time period \( t \) have to be saved in order to evaluate the values of time period \( t-1 \). However, storing evaluated values for all time periods requires a lot of space (computer memory). As the total number of time periods \( T \) grows, the problem of mass storage becomes severer, and the program cannot even run for \( T\geq5 \) on the testing platform. To save storage space and enable the program to run regardless of \( T \), only the evaluated values of two successive time periods are stored. The sufficiency of this method can be justified by observing the fact that to evaluate any value of \( J_t^*(.) \), only the evaluated values of \( J_{t+1}^*(.) \) are needed. This method can be efficiently implemented by using a common programming trick, where two pointers “current” and “next” are introduced to keep track of the values of the current time period (being evaluated) and those of the next time period (already evaluated), and these two pointers are swapped after all the values of the current time period are evaluated.
Appendix D: Running the Java program

The program is implemented in Java. JDK 1.2 or above is required to run the program. The latest JDK can be obtained from Sun’s website at the following URL: http://java.sun.com/downloads/index.html

D1 Program Input

The input to the program is a plain text file containing various configurable parameters, as explained below:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>T = 8</td>
<td>Revision frequency / Planning horizon</td>
</tr>
<tr>
<td>D = 3</td>
<td>Degree of multinomial stock price movement</td>
</tr>
<tr>
<td>C = [0, 0, 0, 0, 0.05, 0, 0, 0, 0]</td>
<td>Coefficient vector of fixed dividend payment</td>
</tr>
<tr>
<td>Al = -1.0</td>
<td>Lower limits on stock position ( a )</td>
</tr>
<tr>
<td>Ah = +1.0</td>
<td>Higher limits on stock position ( a )</td>
</tr>
<tr>
<td>An = 200</td>
<td>Number of segments to divide the range of ( a ) into</td>
</tr>
<tr>
<td>Bl = -1.0</td>
<td>Lower limits on bond position ( b )</td>
</tr>
<tr>
<td>Bh = +1.0</td>
<td>Higher limits on bond position ( b )</td>
</tr>
<tr>
<td>Bn = 200</td>
<td>Number of segments to divide the range of ( b ) into</td>
</tr>
<tr>
<td>K = 1.0</td>
<td>Strike price</td>
</tr>
<tr>
<td>S0 = 1.0</td>
<td>Initial stock price</td>
</tr>
<tr>
<td>sigma = 0.10</td>
<td>( \sigma ) (( d_1 = \exp\left(\frac{\sigma}{\sqrt{T}}\right) ))</td>
</tr>
<tr>
<td>theta = 0.01</td>
<td>( \theta ) - Proportional transaction cost factor</td>
</tr>
<tr>
<td>phi = 0.0</td>
<td>( \phi ) - Fixed transaction cost factor</td>
</tr>
<tr>
<td>deltaA = 0.1</td>
<td>( \delta_a ) - Lot size for the stock</td>
</tr>
<tr>
<td>deltaB = 0.03</td>
<td>( \delta_b ) - Lot size for the bond</td>
</tr>
<tr>
<td>americanOption = true</td>
<td>American options switch</td>
</tr>
<tr>
<td>EPS = 1.0E-10</td>
<td>A very small number to approximate zero</td>
</tr>
</tbody>
</table>

D2 Program Execution

The main program is “DPSolver_Final.java”, which can be compiled with the Java compiler “javac” and executed with the Java interpreter “java”, both of which come with the standard JDK.

To compile: javac DPSolver_Final.java
To execute: java DPSolver_Final “input file” [> “output file”]
Note: “input file” and “output file” should be substituted with actual file paths and “output file” is optional. If “output file” is not specified, program output will be displayed on the standard output.

D3 Program Output
A typical output of the program could be:

minInitVal = 0.1400000000
minInitA = 0.5300000000
minInitB = -0.3900000000

Time spent: 102 seconds