Нахождение цен и ее оптимизация для европейских и американских опционов в биномиальной модели.

Индивидуальный проект

Горская Елена, 695 группа

2 мая 2019

1 Введение

В данной работе рассматриваются европейский и американский опционы в биномиальной модели. Также рассмотрена возможная оптимизация для американского опциона.

Стоит отметить, что в биномиальной модели существует единственная мартингальная мера. В случае европейского опциона процесс дисконтированной цены опциона является мартингалом по мартингальной мере. В то же время в американском опционе данный процесс является супермартингалом. Если держатель данного опциона пропустит оптимальный момент погашения и предъявит опцион позже, то он может потерять прибыль (что и происходит в неравенстве для супермартингалов). Однако данный процесс ведет себя как мартингал в любой момент времени, не являющийся оптимальным для предъявления.

Будет рассмотрена биномиальная модель, алгоритмы для европейского и американского опционов в этой модели. Кроме того, будет выведена оценка сложности алгоритмов нахождения цены европейских и американских опционов. Также покажем, что американский опцион колл не имеет смысла предъявлять к исполнению раньше последнего периода n биномиальной модели, и его цена совпадает с ценой европейского опциона в той же модели и с теми же параметрами.

В практической части данной работы представлены алгоритмы нахождения цены европейского опциона (за квадратичное и линейное время), а также для нахождения цены американского опциона за квадратичное время. Кроме того, рассмотрена и оптимизация для американского алгоритма. Также предъявлены алгоритмы нахождения цен европейских и американских опционов колл и пут по заданным параметрам биномиальной модели и страйку K.

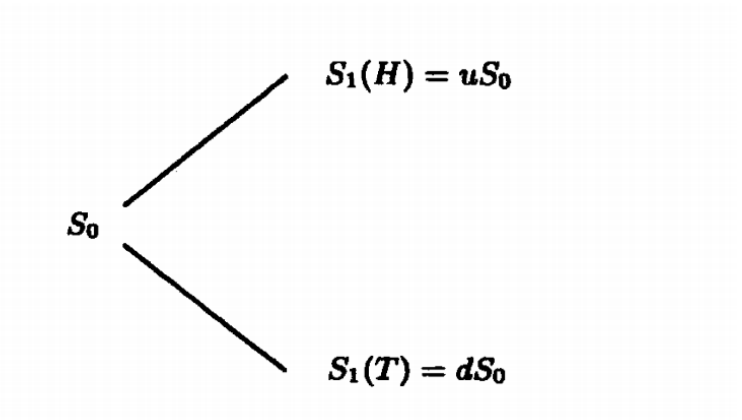
2 Обзор литературы

Большая часть данной работы основывалась на книге Shreve S.E. “Stochastic Calculus for Finance 1”. В ней содержится основная информация о биномиальной модели и о хеджировании. Также в данном источнике рассмотрены основные сведения о европейских и американских опционах применительно к биномиальной модели. Кроме того, в указанной книге разобрана возможность оптимизации американского алгоритма.

3 Биномиальная модель

Для начала рассмотрим, что из себя представляет биномиальная модель.

Пусть у нас есть некий период времени, начинающийся в момент времени 0 и заканчивающийся в момент времени 1. В нулевой момент времени у нас есть базовый актив цены $S_0 > 0$. В момент времени 1 цена базового актива примет одно из двух значений $S_1(H) > 0$ или $S_1(T) > 0$. Можно представить, что мы подбрасываем монетку, и цена актива определяется тем, что на этой монете выпадет (head/tail).
Пусть вероятность выпадения орла (head) равна $p(H) = p > 0$, тогда вероятность выпадения решки составляет $p(T) = q = 1 - p$. Также рассмотрим два положительных числа:
$u = \dfrac{S_1(H)}{S_0}$,
$d = \dfrac{S_1(T)}{S_0}$.
Мы предполагаем, что $u \geq d$. Если же u (up factor) окажется меньше d (down factor), то мы можем поменять стороны монеты местами и достигнем нужного неравенства.



Положим $r > -1$ – это процентная ставка. В биномиальной модели с одним периодом для отсутствия арбитража необходимо, чтобы выполнялось равенство:

$$0 < d < 1 + r < u$$

Очень часто рассматривается $d = \dfrac{1}{u}$, но все же это выполнено не всегда.
Несмотря на кажущуюся простоту биномиальная модель используется довольно часто, поскольку при достаточном числе периодов она дает довольно хорошее приближение моделей с непрерывным временем. Кроме того, с математической точки зрения, при рассмотрении биномиальной модели достаточно просто говорить о мартингалах, условном математическом ожидании и явлении арбитража.

Рассмотрим теперь более сложную биномиальную модель, в которой не один период, а несколько. Предположим, что мы подбрасываем монетку несколько раз подряд: если выпадает орел, то цена базового актива возрастает в u раз, если же выпадает решка, то цена падает, изменяя свое значение в d раз. Кроме того, пусть r – значение процентной ставки. Единственное ограничение, которое мы накладываем на все параметры, – это описанное выше условие отсутствия арбитража.

Итак, пусть стоимость базового актива в нулевой момент времени составляет $S_0 > 0$. Тогда в момент времени 1 стоимость будет:

$S_1(H) = uS_0$, если после первого броска выпал орел.

$S_1(T) = dS_0$, если после первого броска выпала решка.

Во второй момент времени соответственно:

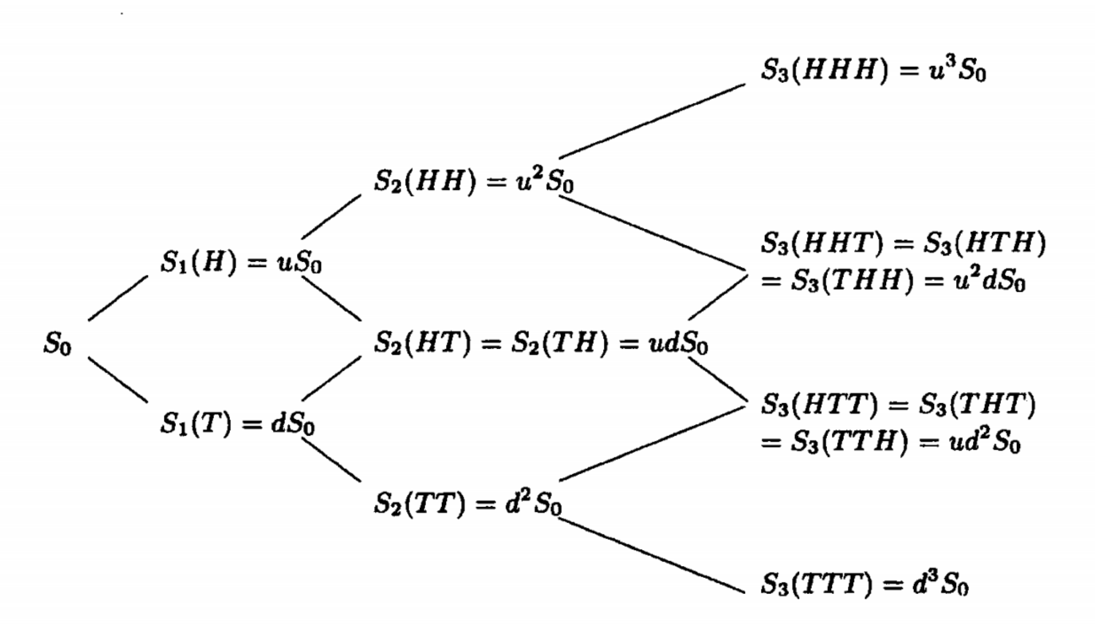
$S_2(HH) = uS_1(H) = u^2 S_0$

$S_2(HT) = dS_1(H) = duS_0$

$S_2(TH) = uS_1(T) = udS_0$

$S_2(TT) = dS_1(T) = d^2 S_0$

Посмотрим, как будет выглядеть биномиальная модель с 3 периодами:



Кроме того, в будущем нам пригодится следующая теорема.

Теорема (о репликации):

Рассмотрим биномиальную модель с N периодами, такую что выполнено $0 < d < 1+r < u$. Пусть также:

$\widetilde{p} = \dfrac{1+r-d}{u-d}$, $\widetilde{q} = \dfrac{u-1-r}{u-d}$ (где $\widetilde{p}$ и $\widetilde{q}$ – это нейтральные к риску вероятности).

Тогда $V_n(\omega_1, ..., \omega_n) = \dfrac{1}{1+r} (\widetilde{p} V_{n+1}(\omega_1, ..., \omega_n, H) + \widetilde{q} V_{n+1}(\omega_1, ..., \omega_n, T))$, где V_n – это цена в момент времени n.

4 Европейский и американский опционы

4.1 Европейский опцион

Европейский опцион колл (European call option) – это право (но не обязательство) купить актив в определенный момент времени (дата погашения) по цене K, называемой *страйком (strike)*.
Европейский опцион пут (European put option) – это право (но не обязательство) продать актив в определенный момент времени (дата погашения) по цене K.
Рассмотрим для начала европейский опцион колл в биномиальной модели с одним периодом. Нас будет интересовать случай, когда $S_1(T) < K < S_1(H)$. Если выпала решка (tail), то не имеет смысла использовать данное право покупки, а вот если выпал орел (head), то мы можем купить актив и тогда получим прибыль в размере $S_1(H) - K$. Таким образом, цена европейского опциона колл будет равна $(S_1 - K)^+$.
Теперь рассмотрим биномиальную модель с n периодами. Для европейского опциона колл также верна формула:
$V_n(\omega_1, ..., \omega_n) = (S_n(\omega_1, ..., \omega_n) - K)^+$.
Таким образом, в n-периодической биномиальной модели цена европейского опциона колл составит $(S_n - K)^+$.
Аналогично получим, что цена европейского опциона пут будет равна $(K - S_n)^+$.

Рассмотрим в общем виде функцию выплат $g(x)$, дающую в момент n прибыль $g(S_n)$.
Таким образом, европейский алгоритм:

$$v_n(s) = max(g(s), 0)$$
$$v_k(s) = \frac{1}{1+r}\left(\widetilde{p}v_{k+1}(us) + \widetilde{q}v_{k+1}(ds)\right),\ k = n-1, n-2, ...0,$$

где $\widetilde{p} = \dfrac{1+r-d}{u-d}$, $\widetilde{q} = \dfrac{u-1-r}{u-d}$.
В указанном алгоритме $v_k(s)$ – это цена опциона в момент времени k, где s – стоимость актива в этот момент времени.

4.2 Американский опцион

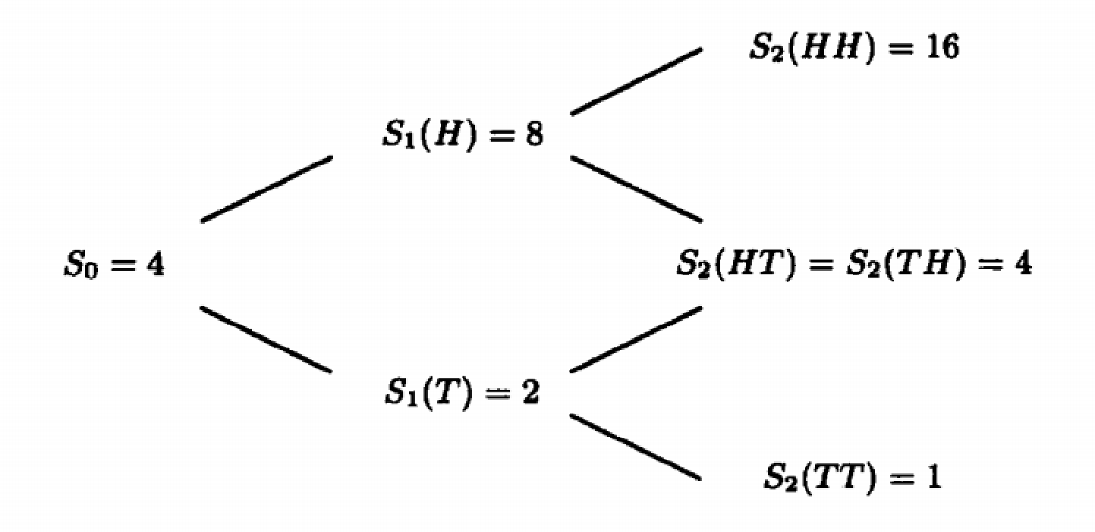
Американский опцион колл (American call option) – это право (но не обязательство) купить актив в любой момент времени $[0..n]$ по цене K.
Американский опцион пут (American put option) – это право (но не обязательство) продать актив в любой момент времени $[0..n]$ по цене K.
Если m – момент предъявления опциона, то в случае опциона колл будет прибыль $(S_m - K)^+$, в то время как в случае опциона пут будет $(K - S_m)^+$.
Рассмотрим американский алгоритм:

$$v_n(s) = max(g(s), 0)$$
$$v_k(s) = max\left(g(s), \frac{1}{1+r}\left(\widetilde{p}v_{k+1}(us) + \widetilde{q}v_{k+1}(ds)\right)\right),\ k = n-1, n-2, ...0.$$

Для лучшего понимания рассмотрим на примере.

Пример 1

Рассмотрим изображенную на рисунке биномиальную модель с двумя периодами. Пусть процентная ставка составит $r = \dfrac{1}{4}$, откуда получим $\widetilde{p} = \widetilde{q} = \dfrac{1}{2}$. Рассмотрим американский опцион пут со сроком погашения 2 и страйком 5. Таким образом, при предъявлении в момент m будет получена прибыль $5 - S_m$.



Таким образом, получается $g(s) = 5 - s$, и американский алгоритм имеет вид:

$$v_2(s) = max(5 - s, 0)$$
$$v_k(s) = max\left(5 - s, \frac{2}{5}\left(v_{k+1}(2s) + v_{k+1}(\frac{s}{2})\right)\right), \ k = 1, 0.$$

В частности, получим:

$v_2(16) = 0$

$v_2(4) = 1$

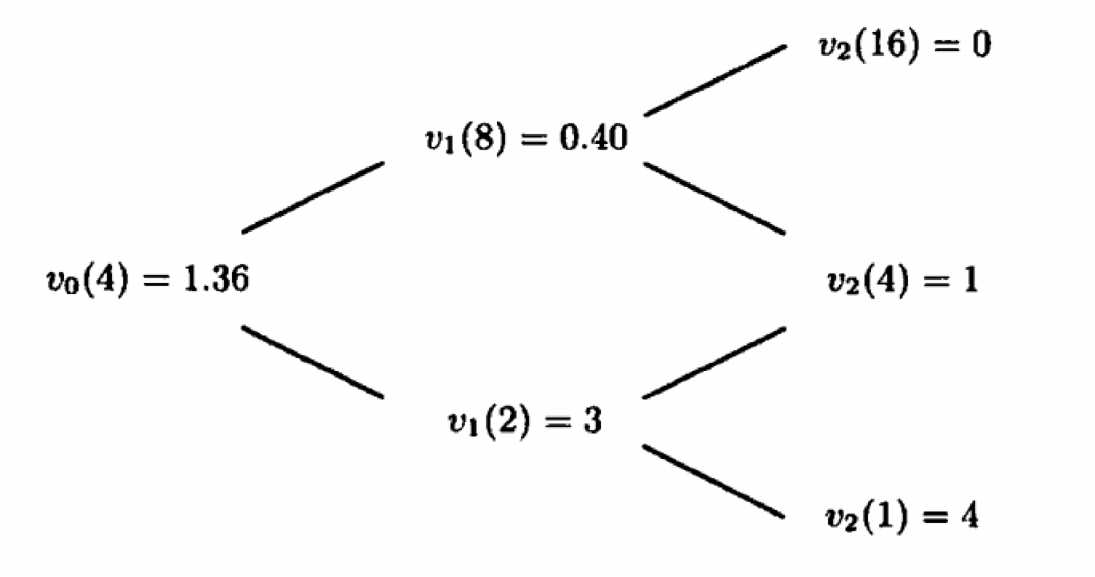
$v_2(1) = 4$

$v_1(8) = max\left(5 - 8, \frac{2}{5}\left(v_2(16) + v_2(4)\right)\right) = max(-3, \frac{2}{5}(1 + 0)) = 0.40$

$v_1(2) = max\left(5 - 2, \frac{2}{5}\left(v_2(4) + v_2(1)\right)\right) = max(3, \frac{2}{5}(4 + 1)) = max(3, 2) = 3$

$v_0(4) = max\left(5 - 4, \frac{2}{5}\left(v_1(8) + v_1(2)\right)\right) = max(1, \frac{2}{5}(0.40 + 3)) = max(1, 1.36) = 1.36$

Так, мы получим результат:



Таким образом, начальная цена американского опциона пут составляет $v_0(4) = 1.36$.

Замечание:
Для европейского опциона пут мы получим совершенно другие значения. Значение $v_1(8)$ останется прежним, а вот $v_1(2)$ станет значительно меньше. Соответственно, $v_0(4)$ также заметно снизится:

$$v_1(8) = \frac{2}{5}\left(v_2(16) + v_2(4)\right) = \frac{2}{5}(1+0) = 0.40$$
$$v_1(2) = \frac{2}{5}\left(v_2(4) + v_2(1)\right) = \frac{2}{5}(4+1) = 2$$
$$v_0(4) = \frac{2}{5}\left(v_1(8) + v_1(2)\right) = \frac{2}{5}(0.40+2) = 0.96$$

Как мы видим, цена действительно ниже, чем у американского опциона пут.

5 Сложность вычислений

5.1 Сложность вычислений для американского опциона

Теорема: Сложность нахождения цены американского опциона составляет $O(n^2)$.

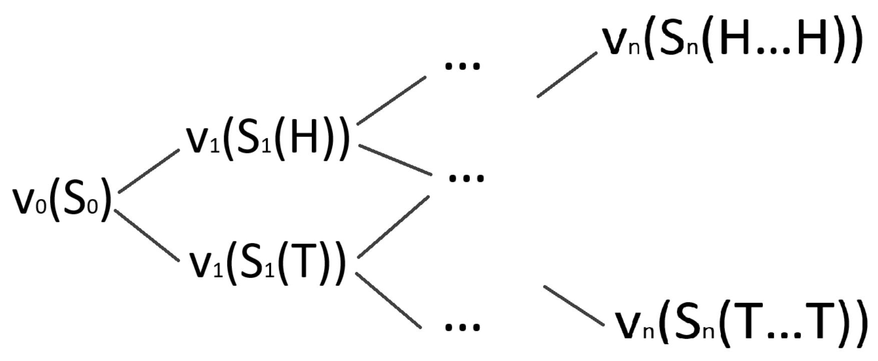
Доказательство:

$\square$

Рассмотрим функцию выплат $g(s)$. В случае американского опциона она будет равна $g(s) = s - K$ или $g(s) = K - s$ при рассмотрении колл и пут опционов соответственно.
Также рассмотрим уже указанный в предыдущем разделе алгоритм нахождения цены в нулевой момент времени.

$$v_n(s) = max(g(s), 0)$$
$$v_k(s) = max\left(g(s), \frac{1}{1+r}\left(\widetilde{p}v_{k+1}(us) + \widetilde{q}v_{k+1}(ds)\right)\right), \; k = n-1, n-2, ...0,$$

Первая строчка очевидным образом вытекает из определения g(s). Вторая строка же получается из того, что процесс марковский и мы можем написать, что $V_n = v_n(X_n)$ (где X_n – стоимость портфеля в момент времени n), а также из того, что в общем виде имеет место формула $v_k(s) = \dfrac{1}{1+r}\left(\widetilde{p}v_{k+1}(us) + \widetilde{q}v_{k+1}(ds)\right)$.
Итак, вернемся к рассмотрению предложенного алгоритма и построим соответствующее дерево:



Отметим, что значение v_n не зависит от пути и, например,
$v_n(TH...HH) = v_n(HT...HH) = ... = v_n(HH...TH) = v_n(HH...HT)$.
Поэтому в последнем периоде (под номером n) будет $(n+1)$ значений: $v_n(S_n(HH...H))$, $v_n(S_n(HH...T))$, ..., $v_n(S_n(TT...T))$. Таким образом, в каждом периоде m будет $(m+1)$ значений для v_m.
Итак, у нас n периодов, в каждом необходимо посчитать не более $(n+1)$ значений. Согласно алгоритму начинаем двигаться от конца (от периода n) к началу (нулевому периоду). При известных значениях в следующем $(m+1)$-ом периоде, значения в m-ом периоде считаются за $O(1)$ по формуле из алгоритма. Таким образом, нам не более $n \cdot (n+1) = n^2 + n$ раз

нужно провести вычисления, каждое из которых занимает $O(1)$. Как итог, v_0 мы вычислим за $O(n^2)$.
■

5.2 Сложность вычислений для европейского опциона

Теорема: Сложность нахождения цены европейского опциона составляет $O(n)$.

Доказательство:

□
Рассмотрим функцию выплат $g(s)$. В случае европейского опциона она также будет равна $g(s) = s - K$ или $g(s) = K - s$ при рассмотрении колл и пут опционов соответственно. Вот только использовать мы в этот раз будем уже не американский, а европейский алгоритм, который также был указан ранее:

$$v_n(s) = max(g(s), 0)$$
$$v_k(s) = \frac{1}{1+r}\left(\widetilde{p}v_{k+1}(us) + \widetilde{q}v_{k+1}(ds)\right),\ k = n-1, n-2, ...0,$$

Дальше есть два способа доказательства. Первый – более неформальный, но в то же время более понятный. Второй – более формальный, но требующий знания дополнительных теорем.

Способ 1
В отличие от американского опциона, в европейском мы при подсчете v_{k-1} используем значение v_k напрямую, без взятия максимума. Поэтому мы просто можем подставить это значение:

$$v_{k-1}(s) = \frac{1}{1+r}\left(\widetilde{p}v_k(us) + \widetilde{q}v_k(ds)\right) =$$
$$= \frac{1}{1+r}\left(\widetilde{p}\cdot\frac{1}{1+r}\left(\widetilde{p}v_{k+1}(u^2s) + \widetilde{q}v_{k+1}(uds)\right) + \widetilde{q}\cdot\frac{1}{1+r}\left(\widetilde{p}v_{k+1}(dus) + \widetilde{q}v_{k+1}(d^2s)\right)\right) =$$
$$= \left(\frac{1}{1+r}\right)^2\left(\widetilde{p}^2\, v_{k+1}(u^2s) + 2\,\widetilde{p}\,\widetilde{q}\, v_{k+1}(uds) + \widetilde{q}^2\, v_{k+1}(d^2s)\right) =$$
$$= \left(\frac{1}{1+r}\right)^3\left(\widetilde{p}^3\, v_{k+2}(u^3s) + 3\,\widetilde{p}^2\,\widetilde{q}\, v_{k+2}(u^2ds) + 3\,\widetilde{p}\,\widetilde{q}^2\, v_{k+2}(ud^2s) + \widetilde{q}^3\, v_{k+2}(d^3s)\right) = ...$$

Таким образом, для v_0 можем записать:

$$v_0(s) =$$
$$=\left(\frac{1}{1+r}\right)^n\left(\widetilde{p}^n\, v_n(u^ns) + C_n^1\,\widetilde{p}^{n-1}\,\widetilde{q}\, v_n(u^{n-1}ds) + ... + C_n^{n-1}\,\widetilde{p}\,\widetilde{q}^{n-1}\, v_n(ud^{n-1}s) + \widetilde{q}^n\, v_n(d^ns)\right) =$$
$$= \left(\frac{1}{1+r}\right)^n \textstyle\sum_{k=0}^n C_n^k\,\widetilde{p}^{n-k}\,\widetilde{q}^k\, v_n(u^{n-k}d^ks)$$

Каждое из v_n мы посчитаем за $O(1)$ по первой строке формулы для европейского алгоритма. Всего такие вычисления надо провести $(n+1)$ раз, результаты сложить и домножить на $\left(\frac{1}{1+r}\right)^n$, что и даст итоговую сложность $O(n)$.

Способ 2
Из теоремы 2.4.5 в "Stochastic Calculus for Finance" знаем, что $\dfrac{X_k}{(1+r)^k}$ - мартингал по риск-нейтральной мере и $\dfrac{X_k}{(1+r)^k} = \widetilde{E}_k \left[\dfrac{X_{k+1}}{(1+r)^{k+1}}\right]$,
где $\widetilde{E}_k$ – это условное математическое ожидание по мартингальной мере, основанное на информации в момент времени k, а X_k – стоимость портфеля в момент времени k.
Из свойства мартингалов получим:

$$\frac{X_k}{(1+r)^k} = E_k\left[\frac{X_n}{(1+r)^n}\right] = E_k\left[\frac{V_n}{(1+r)^n}\right]$$

Таким образом:

$$\frac{V_k}{(1+r)^k} = \widetilde{E}_k\left[\frac{V_n}{(1+r)^n}\right]$$

Или же:

$$V_k = \widetilde{E}_k\left[\frac{V_n}{(1+r)^{n-k}}\right]$$

В том числе:

$$V_0 = \widetilde{E}_0\left[\frac{V_n}{(1+r)^n}\right] = \left(\frac{1}{1+r}\right)^n \widetilde{E}_0(V_n) =$$
$$= \left(\frac{1}{1+r}\right)^n \sum_{k=0}^n C_n^k\, \widetilde{p}^{n-k}\, \widetilde{q}^k\, v_n(u^{n-k}d^k s)$$

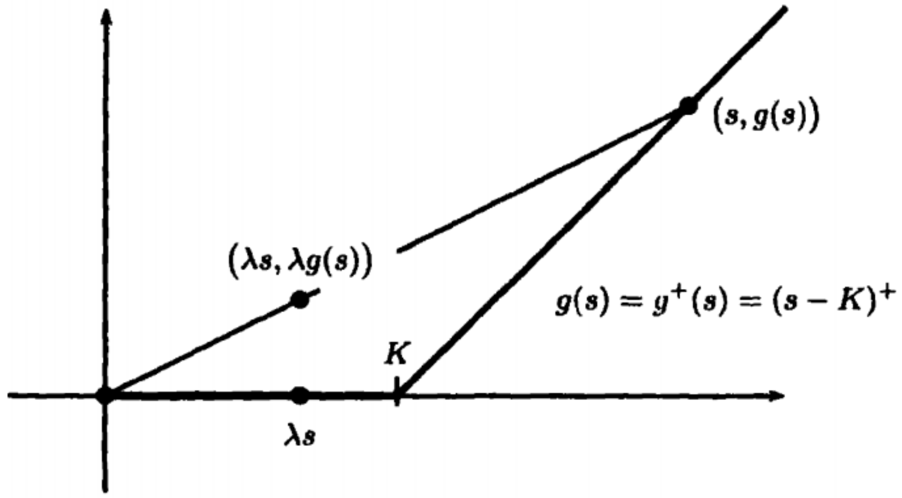
Последнее равенство получаем из определения математического ожидания и того, что значение v_n не зависит от пути.
Далее, аналогично рассуждениям из первого способа получим, что нам нужно вычислить $(n+1)$ значений, причем каждое считаем за $O(1)$. Итого, общий ответ для v_0 мы посчитаем за $O(n)$.
■

5.3 Оптимизация для американского опциона колл

Американский опцион пут иногда бывает выгодно продавать не в последний момент времени n, а раньше. В то же время в случае опциона колл при отсутствии дивидендов не имеет смысла предъявлять опцион до момента n.
Рассмотрим это в общем виде. Пускай у нас имеется *выпуклая* функция $g : [0, \infty) \longrightarrow R$, удовлетворяющая условию $g(0) = 0$. Это значит, что для любых $s_1 \geq 0$, $s_2 \geq 0$ и $0 \leq \lambda \leq 1$ выполнено:

$$g(\lambda s_1 + (1-\lambda)s_2) \leq \lambda g(s_1) + (1-\lambda)g(s_2).$$

Так, в случае опциона колл со страйком K мы имеем: $g(s) = (s-K)^+$ – выпуклая:



Теорема: Рассмотрим биномиальную модель с n периодами, где $0 < d < 1 + r < u$ и процентная ставка неотрицательна ($r \geq 0$). Пускай функция выплат (в американской модели) $g(s)$ – выпуклая и $g(0) = 0$. Тогда стоимость данной ценной бумаги в нулевой момент времени равна стоимости ценной бумаги в европейской модели с той же функцией выплат и сроком погашения n.

Доказательство:
□
Как уже было упомянуто в предыдущей теореме, для европейской ценной бумаги верно равенство:
$V_0^E = \widetilde{E}\left[\frac{1}{(1+r)^n} max(g(S_n), 0)\right]$, поскольку в европейской модели $V_n = max(g(S_n), 0)$.
По определению, для американского процесса верно:
$V_0^A = \max_{\tau \in S_0} \widetilde{E}\left[I_{\tau \leq n} \frac{1}{(1+r)^\tau} g(S_\tau)\right]$, где τ – момент предъявления.
Рассмотрим теперь функцию g. Про нее не говорилась, что она не должна принимать отрицательные значения. Поэтому введем функцию:

$$g^+(s) = max(g(s), 0).$$

Эта функция принимает только неотрицательные значения, причем сохранилось свойство $g^+(0) = max(g(0), 0) = 0$. Кроме того, это тоже выпуклая функция.
Поскольку сама g – выпуклая, то:

$$g(\lambda s_1 + (1-\lambda)s_2) \leq \lambda g(s_1) + (1-\lambda)g(s_2) \leq \lambda g^+(s_1) + (1-\lambda)g^+(s_2)$$
$$\text{для всех } s_1 \geq 0,\ s_2 \geq 0 \text{ и } 0 \leq \lambda \leq 1.$$

В то же время из определения функции g^+:

$$0 \leq \lambda g^+(s_1) + (1-\lambda)g^+(s_2)$$

Таким образом:

$$g^{+}(\lambda s_1+(1-\lambda)s_2)=max(0,g(\lambda s_1+(1-\lambda)s_2)\leq \lambda g^{+}(s_1)+(1-\lambda)g^{+}(s_2).$$

Данное неравенство и доказывает выпуклость функции g^{+}. Кроме того, если мы возьмем $s_1=s$ и $s_2=0$, то получим:

$$g^{+}(\lambda s)\leq \lambda g^{+}(s) \text{ для всех } s\geq 0 \text{ и } 0\leq\lambda\leq 1.$$

Кроме того, мы знаем, что процесс $\dfrac{S_k}{(1+r)^k}$ – это мартингал по риск-нейтральной мере (вероятности $\widetilde{p}$, $\widetilde{q}$). Так что выполнено:

$$S_k=\widetilde{E}_k\left[\frac{1}{1+r}S_{k+1}\right]$$

А также:

$$g^{+}(S_k)=g^{+}\left(\widetilde{E}_k\left[\frac{1}{1+r}S_{k+1}\right]\right)$$

Применим неравенство Йенсена (для условных математических ожиданий):

$$g^{+}\left(\widetilde{E}_k\left[\frac{1}{1+r}S_{k+1}\right]\right)\leq \widetilde{E}_k\left[g^{+}\left(\frac{1}{1+r}S_{k+1}\right)\right]$$

В неравенстве $g^{+}(\lambda s)\leq \lambda g^{+}(s)$ положим $\lambda=\dfrac{1}{1+r}$ и получим:

$$g^{+}\left(\frac{1}{1+r}S_{k+1}\right)\leq \frac{1}{1+r}g^{+}(S_{k+1})$$

За счет свойства условного математического ожидания:

$$\widetilde{E}_k\left[g^{+}\left(\frac{1}{1+r}S_{k+1}\right)\right]\leq \widetilde{E}_k\left[\frac{1}{1+r}g^{+}(S_{k+1})\right]$$

Таким образом, получили:

$$g^{+}(S_k)\leq \widetilde{E}_k\left[\frac{1}{1+r}g^{+}(S_{k+1})\right]$$

Если мы домножим обе части неравенства на $\dfrac{1}{(1+r)^k}$, то получим свойство субмартингалов:

$$\frac{1}{(1+r)^k}\,g^{+}(S_k)\leq \widetilde{E}_k\left[\frac{1}{(1+r)^{k+1}}g^{+}(S_{k+1})\right]$$

для дисконтированного процесса $\dfrac{1}{(1+r)^k}\,g^{+}(S_k)$. Поскольку этот процесс является субмартингалом, то можем применить *Теорему 2* (указана после данной теоремы). Указанная теорема говорит, что для любого момента остановки τ:

$$\widetilde{E}\left[\frac{1}{(1+r)^{min(n,\tau)}}g^{+}(S_{min(n,\tau)})\right]\leq \widetilde{E}\left[\frac{1}{(1+r)^{n}}\,g^{+}(S_n)\right]=V_0^E$$

Если $\tau \le n$, то:

$$I_{\tau \le n} \frac{1}{(1+r)^{\tau}} g(S_\tau) = \frac{1}{(1+r)^{min(n,\tau)}} g(S_{min(n,\tau)}) \le \frac{1}{(1+r)^{min(n,\tau)}} g^+(S_{min(n,\tau)})$$

Если $\tau = \infty$, то:

$$I_{\tau \le n} \frac{1}{(1+r)^{\tau}} g(S_\tau) = 0 \le \frac{1}{(1+r)^{min(n,\tau)}} g^+(S_{min(n,\tau)})$$

В каждом из случаев имеем:

$$\widetilde{E}\left[I_{\tau \le n} \frac{1}{(1+r)^{\tau}} g(S_\tau)\right] \le \widetilde{E}\left[\frac{1}{(1+r)^{min(n,\tau)}} g^+(S_{min(n,\tau)})\right] \le V_0^E.$$

Поскольку это неравенство выполнено для любого момента остановки $\tau \in S_0$, то:
$V_0^A = \max_{\tau \in S_0} \widetilde{E}\left[I_{\tau \le n} \frac{1}{(1+r)^{\tau}} g(S_\tau)\right] \le V_0^E$
А поскольку из определения $V_0^E \le V_0^A$, то имеем требуемое равенство:

$$V_0^A = V_0^E$$

■

Следствие: Цена американского опциона колл в нулевой момент времени равна цене европейского опциона колл и также может быть посчитана за $O(n)$.

Теорема 2 (б/д): Пусть X_k, $k = 0, 1, \ldots n$ – субмартингал, а τ – момент остановки. Тогда $EX_{min(k,\tau)} \le EX_k$. Если X_k – супермартингал, то $EX_{min(k,\tau)} \ge EX_k$. Если же X_k – мартингал, то $EX_{min(k,\tau)} = EX_k$

6 Практическая часть

В практической части данной работы рассмотрим реализацию описанных выше алгоритмов на языке программирования Python.
Для начала необходимо импортировать одну дополнительную библиотеку:

```
import numpy as np
```

6.1 Европейский алгоритм

Сначала рассмотрим, как будет выглядеть европейский алгоритм за квадратичное время, если на вход программе подаются параметры биномиальной модели и функция выплат:

```
def eur_alg_square(r, u, d, s, n, g):
    p_neut = (1 + r - d)/(u - d)
    q_neut = 1 - p_neut
    v = np.zeros((n + 1, n + 1))

    for i in range(0, n+1):
        v[n][i] = max(g((u ** i) * (d ** (n - i)) * s), 0)

    for period in range(n-1, -1, -1):
        for j in range(0, period + 1):
            v[period][j] = (1 / (1 + r)) * (p_neut * v[period + 1][j] + q_neut * v[period + 1][j + 1])
    return v[0][0]
```

В данной реализации мы рассматриваем приведенный в самом начале европейский алгоритм и проходимся по всему дереву, чтобы найти цену опциона. Однако данные вычисления можно провести и за линейное время, что было доказано в теоретической части проекта. В этом случае алгоритм будет выглядеть следующим образом:

```
def eur_alg_linear(r, u, d, s, n, g):
    p_neut = (1 + r - d)/(u - d)
    q_neut = 1 - p_neut
    v = np.zeros(n + 1)

    for i in range(0, n + 1):
        v[i] = max(g((u ** i) * (d ** (n - i)) * s), 0)

    v_0 = 0
    for i in range(0, n + 1):
        v_0 += (p_neut ** (n-i)) * (q_neut ** i) * bin_coef(n, i) * v[n-i]

    v_0 *= (1 / (1 + r)) ** n

    return v_0
```

Для этой реализации необходима функция вычисления биномиальных коэффициентов:

```python
def factorial(n):
    if (n == 0):
        return 1
    else:
        return n * factorial(n-1)

def bin_coef(n, m):
    return factorial(n) / ((factorial(n - m)) * factorial(m))
```

6.2 Американский алгоритм

В случае американского алгоритма необходимо будет пройти по всему дереву, таким образом сложность составит $O(n^2)$:

```python
def amer_alg(r, u, d, s, n, g):
    p_neut = (1 + r - d)/(u - d)
    q_neut = 1 - p_neut
    v = np.zeros((n + 1, n + 1))

    for i in range(0, n+1):
        v[n][i] = max(g((u ** i) * (d ** (n - i)) * s), 0)

    for period in range(n-1, -1, -1):
        for j in range(0, period + 1):
            v[period][j] = max(
                g((u ** j) * (d ** (period - j)) * s),
                (1 / (1 + r)) * (p_neut * v[period + 1][j] + q_neut * v[
period + 1][j + 1])
            )
    return v[0][0]
```

Однако для данного алгоритма существует и оптимизация, которая рассматривалась в теоретической части настоящей работы. Для применения этой оптимизации необходимо, чтобы функция выплат g была выпуклой. Проверить выпуклость функции в узлах дерева можно при помощи следующего алгоритма:

```python
def is_convex_fast(g, s, u, d, n, cur_period):
    is_conv = True

    node_values = []
    for period in range(cur_period, n+1):
        for j in range(0, period+1):
            node_values.append(s * (u**j) * (d**(period-j)))

    for value_1 in node_values:
        for value_2 in node_values:
            if (g((value_1+value_2) / 2)-(g(value_1)+g(value_2))/2 > 1e-7):
                is_conv = False

    return is_conv
```

Тогда оптимизированный алгоритм будет иметь вид:

```python
def amer_alg_opt(r, u, d, s, n, g):

    # if g is convex
    if (is_convex_fast(g, s, u, d, n, 0) and g(0) == 0):
        return eur_alg_linear(r, u, d, s, n, g)

    # if g is partly convex
    if (g(0) == 0):
        p_neut = (1 + r - d)/(u - d)
        q_neut = 1 - p_neut
        v = np.zeros((n + 1, n + 1))

        # values in the last period
        s_last = []
        for i in range(n+1):
            s_last.append(s * (u ** j) * (d ** (n - j)))

        for i in range(0, n+1):
            v[n][i] = max(g((u ** i) * (d ** (n - i)) * s), 0)

        is_partly_convex = False
        convex_indexes = []
        convex_period = 0

        # we look for the earliest period where we can apply optimization
        for period in range(0, n):
            for j in range(0, period + 1):
                if (is_convex_fast(g, s, u, d, n, period)):
                    is_partly_convex = True
                    convex_indexes.append(j)
                    convex_period = period
            if(is_partly_convex):
                break

        if(is_partly_convex):
            for j in range(0, convex_period + 1):
                if (j in convex_indexes):
                    s_current = (u ** j) * (d ** (period - j)) * s
                    v[period][j] = eur_alg_linear(r, u, d, s_current, n-period, g)
                else:
                    s_current = (u ** j) * (d ** (period - j)) * s
                    v[period][j] = amer_alg(r, u, d, s_current, n-period, g)

            for period in range(convex_period - 1, -1, -1):
                for j in range(0, period + 1):
                    v[period][j] = v[period][j] = max(
                g((u ** j) * (d ** (period - j)) * s),
                (1 / (1 + r)) * (p_neut * v[period + 1][j] + q_neut * v[period + 1][j + 1]))

        else:
            v[0][0] = amer_alg(r, u, d, s, n, g)

```

```
        return v[0][0]

    else:
        return amer_alg(r, u, d, s, n, g)
```

6.3 Европейский опцион

Рассмотрим непосредственно алгоритмы для европейских опционов колл и пут в биномиальной модели. Соответствующие функции выплат будут иметь следующий вид:

```
def eur_call(K, s):
    return (s - K)

def eur_put(K, s):
    return (K - s)
```

Таким образом, алгоритм для вычисления цены европейских опционов колл и пут будет работать за $O(n)$:

```
def eur_option_call_price(r, u, d, s, n, K):
    def g(s):
        return eur_call(K, s)
    v_0 = eur_alg_linear(r, u, d, s, n, g)
    return v_0

def eur_option_put_price(r, u, d, s, n, K):
    def g(s):
        return eur_put(K, s)
    v_0 = eur_alg_linear(r, u, d, s, n, g)
    return v_0
```

На вход подаются параметры биномиальной модели, число периодов и значение страйка K.

6.4 Американский опцион

Для американских опционов колл и пут функции выплат будут иметь вид:

```
def amer_call(K, s):
    return (s - K)

def amer_put(K, s):
    return (K - s)
```

Заметим, что для опциона колл функция является выпуклой, поэтому для него возможна оптимизация, и работать алгоритм вычисления цены американского опциона колл будет за линейное время. Итого, получим следующую реализацию:

```python
def amer_option_call_price(r, u, d, s, n, K):
    v_0 = eur_option_call_price(r, u, d, s, n, K)
    return v_0

def amer_option_put_price(r, u, d, s, n, K):
    def g(s):
        return amer_put(K, s)
    v_0 = amer_alg_opt(r, u, d, s, n, g)
    return v_0
```

Более подробный код и ряд примеров содержатся в приложенном к данной работе файле.