Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный университет»
Математико-механический факультет
Кафедра статистического моделирования

Отчёт о научно-исследовательской практике

Миллер Анастасия Александровна

Устранение экспоненциальной сложности оценки стоимости бермудского опциона

Научный руководитель:
д. ф.-м. н., профессор С. М. Ермаков

Санкт-Петербург
2014

Оглавление

Введение

Метод оценки американских опционов с конечным числом дат погашения, основанный на моделировании дерева событий (метод случайных деревьев), был предложен в [1] ещё в 2004 году. Этот метод моделирует изменение состояния базового актива через случайные деревья, разветвляющиеся в каждой из возможных дат раннего погашения опциона. При анализе деревьев могут быть получены две оценки: смещённая вверх и смещённая вниз, являющиеся асимптотически несмещёнными и дающие доверительный интервал для истинной цены опциона.

Вычислительная сложность этих оценок — $O(b^s)$, где b — количество ветвей дерева (моделируемых вариантов изменения цены опциона), s — количество шагов алгоритма (дат погашения опциона). Рассмотрим реализацию и анализ одного из способов снизить вычислительную сложность этих оценок.

Глава 1

Алгоритм

Рассматриваем опцион с s датами исполнения, общий период времени положим равным 1, т.е. имеем $\{t_k\}_{k=1}^{s}$ — набор моментов времени. Смоделированное состояние актива (на который выписан опцион) в момент времени t_k описывается ${}_{k}^{i_1\cdots i_k}$, где $\forall\, k \in 1 : s \quad i_k \in 1 : b$ (i_j указывает на номер узла, выбранный на j-ом шаге).

Начиная с некоторого момента t_k, когда общее число состояний на шаге достигнет некоторого n, мы перестанем генерировать дочерние вершины ко всем состояниям. В следующий момент времени, t_{k+1}, мы будем иметь всё так же n состояний, а не bn.

В том случае, когда состояние актива S является числом в $\mathbb{R}^1$, в качестве параметра X, распределение которого нас интересует, можно использовать само S, иначе можно использовать $h(S)$.

Деля интервал $\left[\min_{i\in 1:n} X_i; \max_{i\in 1:n} X_i + \frac{1}{n}\right)$ на k равных частей $[a_{k-1}, a_k)$, где $a_0 = \min_{i\in 1:n} X_i$, $a_k = \max_{i\in 1:n} X_i$, мы можем определить частоты

$$f_k = \frac{1}{n} \# \left\{ X_i | X_i \in [a_{k-1}, a_k) \right\}$$

попадания событий в различные части отрезка. Из состояний, сгруппированных на отрезке $[a_{k-1}, a_k)$, мы также можем создать некоторый «средний арифметический» вектор, cooринаты которого будут являться средним арифметическим координат всех состояний, оказавшихся на данном отрезке, и уже для этого нового среднего состояния — представителя отрезка — генерировать дочерние вершины в количестве $n \cdot f_k$. Для всех состояний, оказавшихся в этом отрезке, дочерними вершинами будут являться все вершины, полученные от их представителя.

Таким образом, количество рассматриваемых состояний не увеличится.

Глава 2

Реализация

В качестве параметра, распределение которого будет анализироваться, я использовала цену актива (реализацию винеровского случайного процесса, где каждое следующее состояние получается из предыдущего как $p_0 \cdot (1 + a \cdot \triangle t + \sigma \cdot \varepsilon \cdot \sqrt{\triangle t})$, где p_0 — цена актива в предыдущий момент времени, $\triangle t = 1/s$, a и σ означают доходность и волатильность цены акции соответственно и являются константами, ε — случайная величина со стандартным нормальным распределением).

На рис. 2.1 можно видеть, как выглядит генерируемое исходным методом дерево.

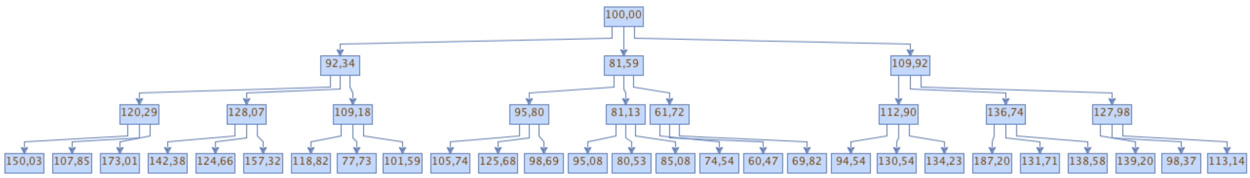

Рис. 2.1. Дерево, генерируемое при использовании метода, описываемого Глассерманом (цифры в узлах — стоимость актива)

В своей реализации я разделяю генерацию дерева и подсчёт оценок, ему соответствующих, так как оценки, в отличие от исходных деревьев, у меня не отличаются от оценок у Броади и Глассермана. Вначале существовала надежда сравнивать «полные» и «урезанные» деревья, но она не оправдалась из-за слишком больших требований к памяти у классических деревьев.

Момент, после которого стоит переходить на линейную модель генерации дочерних вершин, определился как $k = \left\lfloor {}^{\log n}/_{\log b} \right\rfloor$ (b — количество ветвей у узла в «экспоненциальном режиме», n — ширина дерева в «линейном режиме»). Реализацию можно увидеть в приложении A.1.

Дерево, генерируемое усечённым образом, можно увидеть на рис. 2.2.

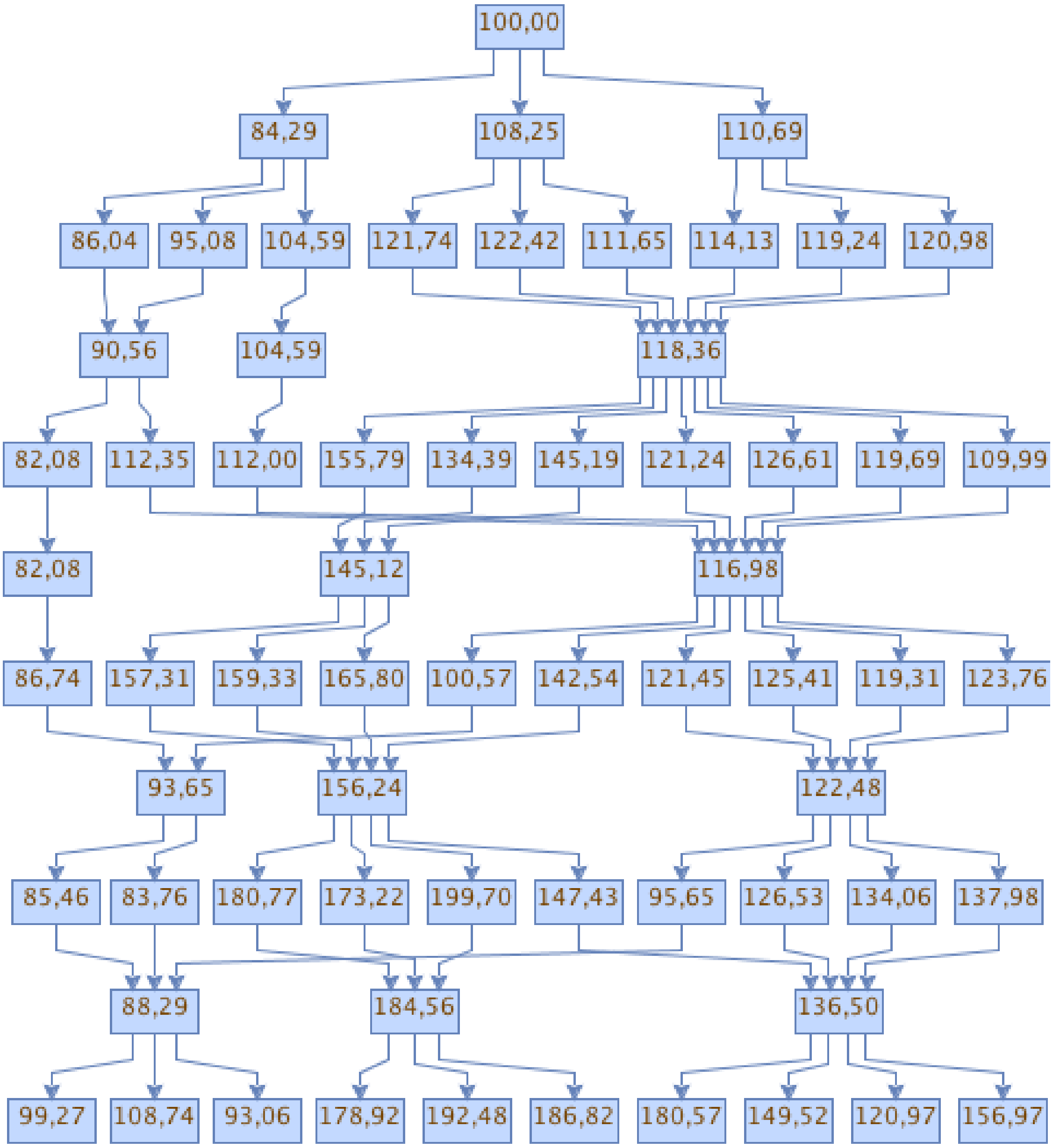

Рис. 2.2. Дерево, генерируемое усечённым методом (цифры — стоимость актива; ширина дерева $n = 10$, количество секторов $k = 3$)

Глава 3

Результаты

Целью увеличения доступного для обсчёта числа дат исполнения было максимально приблизиться к американскому опциону (опциону с возможностью исполнения в любой момент в оговорённом промежутке времени). Неизвестными факторами (поведение которых не было очевидным на стадии создания упрощённого метода) были ширина дерева n и количество «столбцов гистограммы» k. Также было неясно, существует ли сходимость метода, сопоставимая со сходимостью исходного метода.

Испытания сходимости метода были проведены по алгоритму 1.

```
startSteps = 50
for i ∈ 1 : 100 do
    x_1 = ∞
    x_0 = −∞
    step = 0
    while |x_1 − x_0| > ϵ and step<1000 do
        x_0 = x_1
        x_1 = estimateAsset(steps=startSteps+step, width=50, sectors=k)
        step += 1
    end
    if step == 1000 then
        в этом испытании алгоритм не сошёлся
    else
        в этом испытании алгоритм сошёлся
    end
end
```

Алгоритм 1: Проверка сходимости оценки в испытании

Для верхней оценки стоимости опциона результаты можно увидеть на рис. 3.1, для нижней оценки — на рис. 3.2.

Вероятность (выборочная) сходимости верхней и нижней оценки представлена на гра-

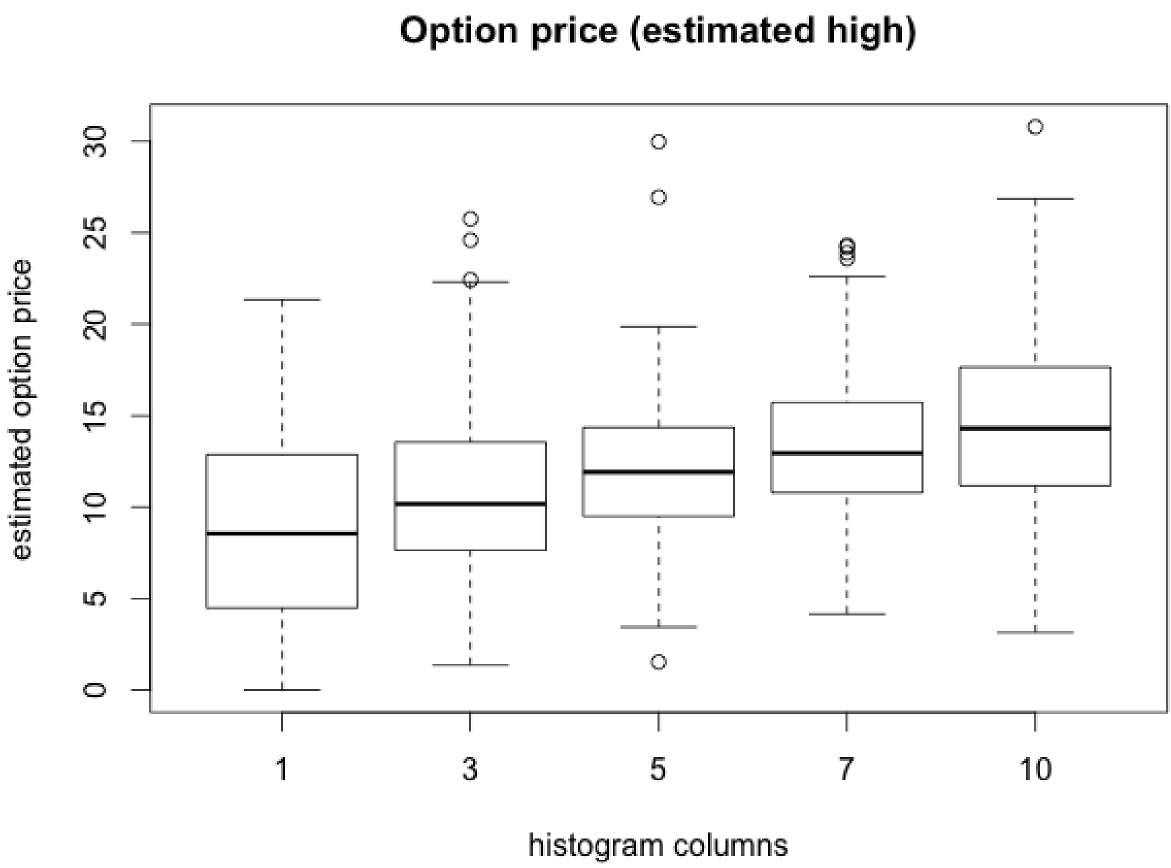

Рис. 3.1. Распределение верхней оценки стоимости опциона

фиках 3.3 и 3.4 соответственно.

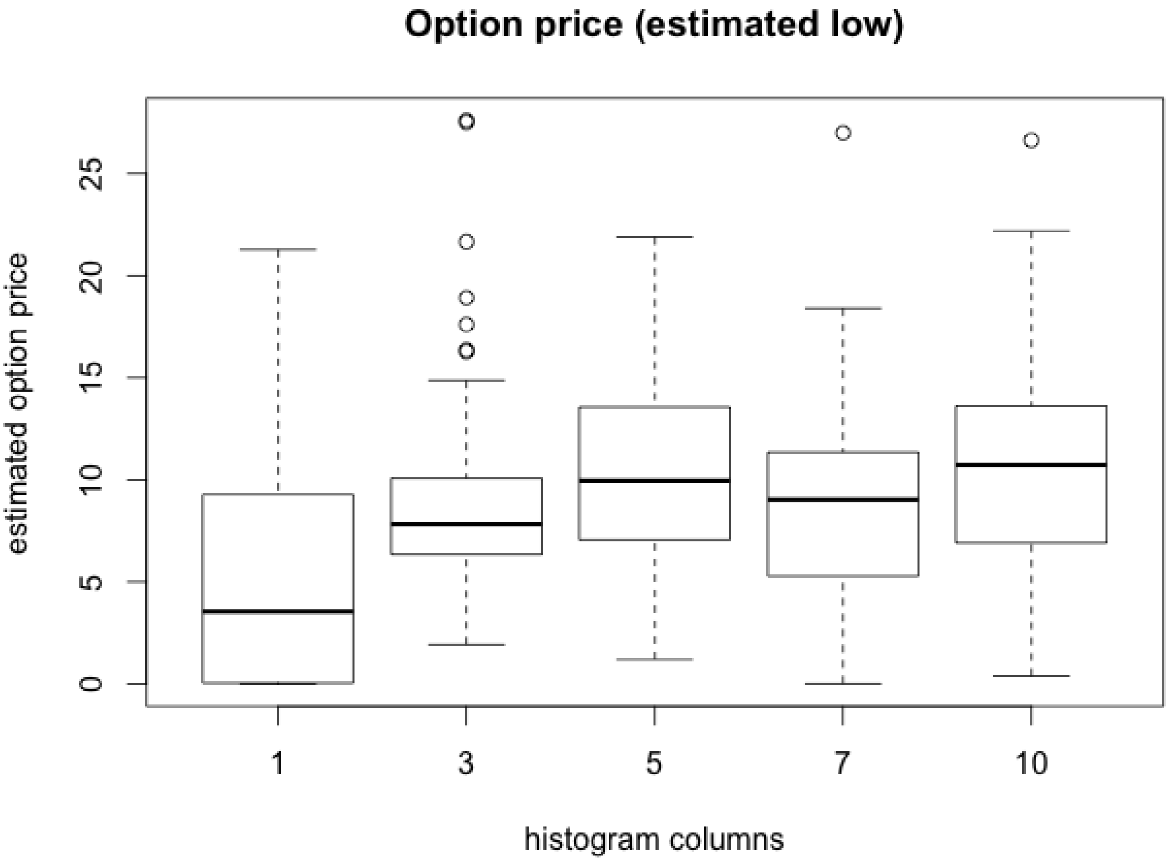

Рис. 3.2. Распределение нижней оценки стоимости опциона

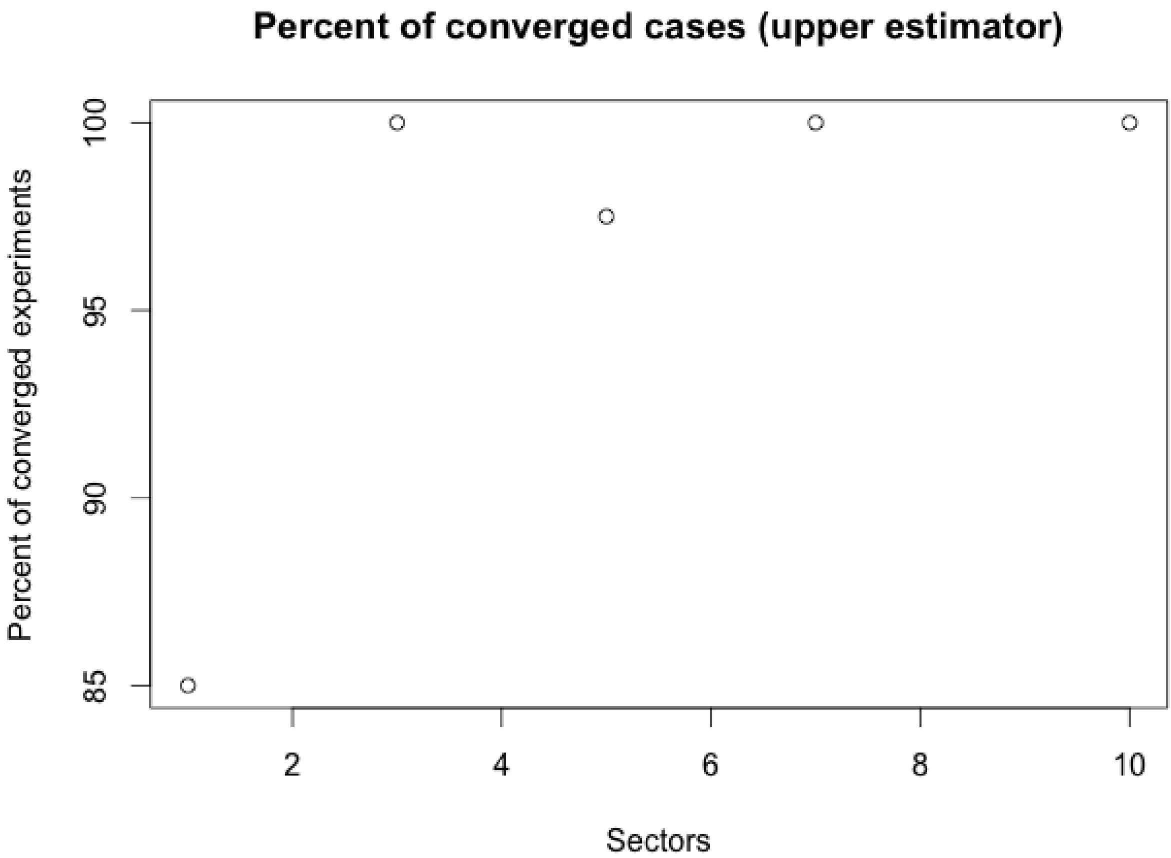

Рис. 3.3. Процент случаев, в которых верхняя оценка сошлась, по отношению к общему числу испытаний

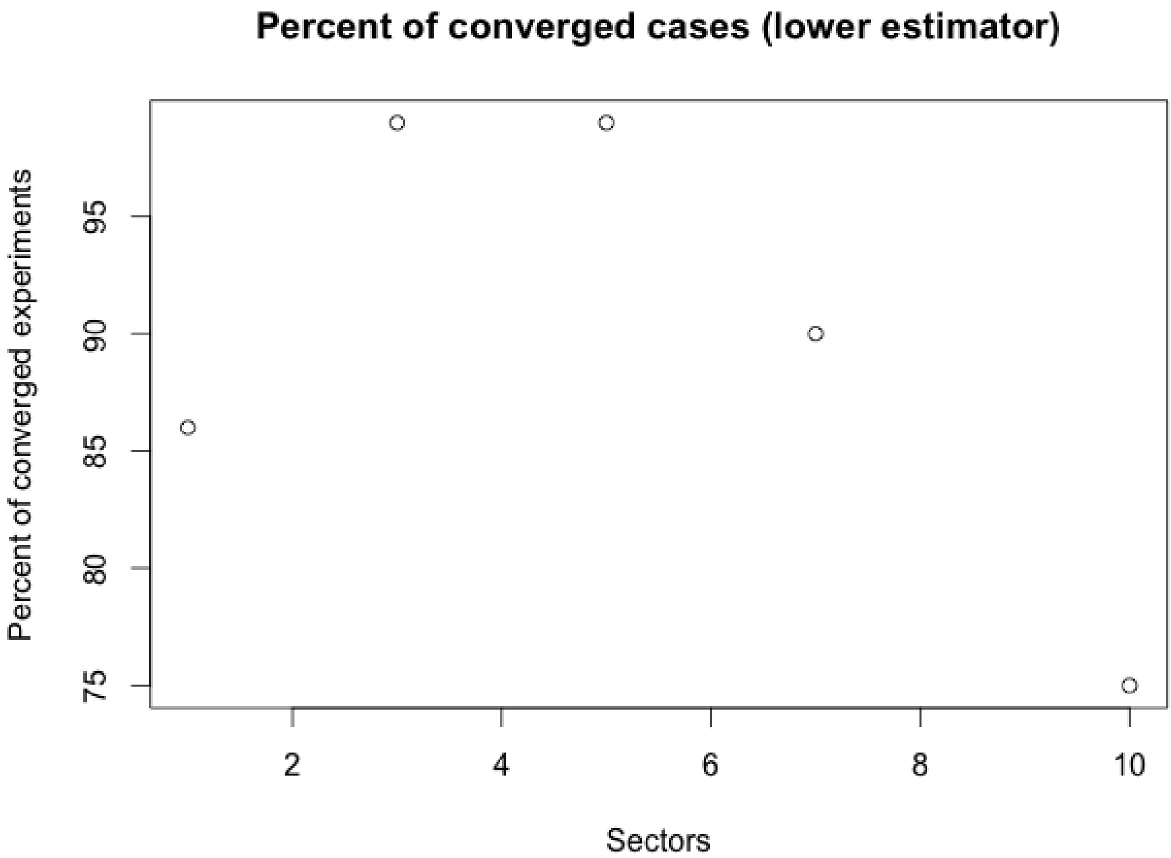

Рис. 3.4. Процент случаев, в которых нижняя оценка сошлась, по отношению к общему числу испытаний

Заключение

Как видно, оценки имеют большую дисперсию, причём если для нижней оценки оптимальное разбиение на подмножества кажется равным 3 (для ширины в 50 узлов), то для верхней оценки локальный минимум не очевиден.

Дальнейшие планы

1. Закончить рассмотрение оценки по гистограмме, в т.ч. найти аналитически математическое ожидание оценки (похожий случай уже рассмотрен в [2])
2. Рассмотреть оценку по кластерам (предполагаемый алгоритм кластеризации рассмотрен в [3])
3. Рассмотреть другие оценки

Список литературы

1. Glasserman Paul. Monte Carlo Methods in Financial Engineering. — Springer, 2004.
2. Ермаков Сергей Михайлович. Метод Монте-Карло и смежные вопросы. — Наука, 1975.
3. Arthur David, Vassilvitskii Sergei. k-means++: The Advantages of Careful Seeding // SODA. — 2007. — URL: http://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf.
4. Broadie M., Glasserman P. Pricing American-style securities by simulation // Journal of Economic Dynamics and Control. — 1997. — Vol. 21. — P. 1323–1352.
5. Broadie Mark, Glasserman Paul, Jain Gautam. Enhanced Monte Carlo estimates for american option prices // Journal of Derivatives. — 1997. — Vol. 5, no. 1 (Fall). — P. 25–44.

Приложение A

Реализация на Java

Листинг A.1. Генерирование дерева состояний актива, на который выписан опцион

```
public static ImitatedAsset generateAssetByHistogram(int width, int branch,
    int steps, int sectors, double initialPrice){
    timedelta = 1. / steps;
    int expSteps = (int) Math.floor(Math.log(width) / Math.log(branch));
    ImitatedAsset[] nodes = new ImitatedAsset[width];
    ImitatedAsset ans = generateTreeAssetsToModeling(branch, expSteps,
        initialPrice, nodes);

    ImitatedAsset[] new_nodes = generateFirstRow(width, sectors, nodes);
    nodes = new_nodes;

    // +1 because of one step that was done outside the cycle
    for (int step = expSteps + 1; step < steps; step++) {
        new_nodes = generateRow(width, (step + 1 == steps), sectors, nodes);
        nodes = new_nodes;
    }
    return ans;
}
private static ImitatedAsset[] generateRow(int width, boolean lastRow, int
    sectors, ImitatedAsset[] nodes) {
    ImitatedAsset[] new_nodes;
    sortArrayWithNulls(nodes);
    double sector = getSectorWidth(sectors, nodes);
    double min = extremalValue(nodes, -1);
    double sum = 0;
    int amount = 0;
    int k = 0;
    new_nodes = new ImitatedAsset[width];
    for (int j = 0; j < width; j++){ // iterating over {{nodes}}
        if (nodes[j].price > min + (k+1) * sector) { // reached the end of
            the sector
            generateBlock(nodes, new_nodes, lastRow, j-amount, j, amount,
                sum/amount);
```

```
                k++;
                amount = 0;
                sum = 0;
            }
            sum += nodes[j].price;
            amount++;
        }
        generateBlock(nodes, new_nodes, lastRow, width-amount, width, amount,
            sum/amount);
        return new_nodes;
    }
    private static void generateBlock(ImitatedAsset[] nodes, ImitatedAsset[]
        new_nodes, boolean lastRow, int start, int end, int children, double
        price, int new_start){
        ImitatedAsset asset = new ImitatedAsset(price, children, false); //
            intermediate asset will definitely have children
        for (int i = start; i < end; i++ ){ // assign average node as a child to
            the previous generation
            nodes[i].children[0] = asset;
        }
        for (int i = 0; i < children; i++){ // generating new nodes
            asset.children[i] = new ImitatedAsset(getRandomPrice(asset.price), 1,
                lastRow);
            new_nodes[new_start+i] = asset.children[i];
        }
    }
```