Московский государственный университет имени
М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра системного анализа

Выпускная квалификационная работа.

«Численные методы построения вогнутой оболочки функций»

Студент 415 группы
М. Г. Зинченко

Научный руководитель
С.Н. Смирнов

Москва, 2020

1 Введение

Вогнутые оболочки используются в различных приложениях. В первую очередь нас будет интересовать их применимость в финансовой математике, а точнее в задаче суперхеджирования, изложенной в статье [1]. В этой статье построена модель с рядом вполне логичных ограничений, позволяющая получить рекуррентные уравнения для нахождения цены опциона. Согласно [3] решение этих уравнений сводится к нахождению вогнутой оболочки функции. Соответственно, целью работы будет исследование применимости различных известных численных методов для задачи суперхеджирования с наиболее часто используемыми функциями выплат по опциону, будут рассмотрены как случай классических опционов, так и Rainbow-опционов, выплата которых зависит от нескольких активов.

В этой работе мы подробно разберем несколько численных методов, протестируем их производительность, отметим их достоинства и недостатки. Нами будет рассмотрен Алгоритм Грэхема, впервые упомянутый в 1972 в статье [6]. Это одна из первых работ, посвященных построению выпуклых оболочек. Алгоритм, описанный в этой статье был несколько упрощен, но принцип, по которому он работает остался тем же. Позднее была опубликована статья с алгоритмом Джарвиса [7]. Основным его преимуществом по сравнению с алгоритмом Грэхема считается высокая скорость работы при малом количестве крайних точек в оболочке.

Для функции многих переменных будет рассмотрено преобразование Лежандра-Фенхеля, его основные свойства, продемонстрировано как его можно использовать в задаче построения вогнутой оболочки. Преобразование Лежандра-Фенхеля достаточно полно разобрано в книге [8]. В этой книге оно рассматривается в приложениях к электролитам и подробно описывается случай функции двух переменных. Мы рассмотрим преобразование Лежандра-Фенхеля не так подробно, сделаем больший акцент на его применение, приведем ряд теорем без доказательств исключительно для обоснования алгоритма. Также разберем метод быстрой оболочки (QuickHull) для функций многих переменных. Этот метод описан в статье [9]. Также в этой статье можно найти сравнение этого метода с рандомизованными алгоритмами построения выпуклой оболочки в n-мерном пространстве.

2 Вогнутая оболочка функции

В этом разделе мы введем ряд базовых терминов, введем понятие вогнутой оболочки и сделаем важное замечание.

Определение 1 *Пересечение всех выпуклых множеств, содержащих заданное множество S называется выпуклой оболочкой множества S и обозначается через* $\operatorname{conv} S$.

Определение 2 *Пусть $f : \mathbb{R}^n \to \overline{\mathbb{R}}$. Тогда множество*

$$\operatorname{epi} f := \{(x, y) \in \mathbb{R}^n \times \mathbb{R} | y \geqslant f(x)\}$$

называется надграфиком f.

Определение 3 *Функция $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ называется выпуклой, если ее надграфик* $\operatorname{epi} f$ *выпуклый.*

Определение 4 *Выпуклой оболочкой невыпуклой функции f называется функция $g = \operatorname{conv} f$, такая что $g(x) = \inf\{\mu | (x, \mu) \in \operatorname{conv}(\operatorname{epi} f)\}$.*

Замечание. Далее в работе мы часто будем говорить именно о выпуклой оболочке, поскольку этот термин чаще встречается в используемой нами литературе.

Определение 5 *Вогнутой оболочкой функции f будем называть функцию $\hat{f} = -\operatorname{conv}(-f)$.*

В силу приведенного определения, найти вогнутую оболочку функции, зная ее выпуклую оболочку, не составляет большого труда.

3 Детерминистский подход к суперхеджированию

В этом разделе мы опишем задачу, решение которой сведется к вычислению вогнутых оболочек функций. Нас будет интересовать задача ценообразования для опционов в рамках гарантированного детерминистского подхода к суперхеджированию.

Обратимся к статье [1]. В ней введены предположения, позволяющие построить детерминистскую модель рынка и получить рекуррентные уравнения для вычисления цены опциона. Следующие уравнения будем называть уравнениями Беллмана-Айзекса.

$$V_N^*(\overline{x}_N) = g_N(\overline{x}_N) \tag{1}$$

$$V_{t-1}^*(\overline{x}_{t-1}) = g_{t-1}(\overline{x}_{t-1}) \vee \inf_{h \in D_t(\overline{x}_{t-1})} \sup_{y \in K_t(\overline{x}_{t-1})} \left[V_t^*(\overline{x}_{t-1}, y + x_{t-1}) - hy \right], \tag{2}$$

$$t = N, \ldots, 1.$$

Здесь $\overline{x}_{t-1}$ — предыстория цен базовых активов. Для базовых активов рассматривается аддитивная модель ценообразования, в которой приращения цен активов ΔX_t лежат в множествах, задаваемых выпуклозначными многозначными отображениями $K_t(\overline{x}_{t-1})$.

V_t — стоимость в момент t портфеля, который формирует продавец опциона, получивший в начальный момент премию V_0 и использующий стратегию $h = (h_1, \ldots, h_N)$ с целью хеджирования обусловленного обязательства, то есть $V_t = h_t x_t$.

V_t^* — точная нижняя грань стоимости V_t портфеля в момент времени t, гарантирующего исполнение текущих и будущих обязательств, касающихся потенциальных выплат при исполнении опциона, для стратегий суперхеджирования.

$g_t(\overline{x}_t)$ — функция выплат по опциону.

D_t — многозначное отображение, задающее ограничения на структуру портфеля h_t в момент времени t.

Стратегии $h = (h_1, \ldots, h_N)$ будем называть суперхеджирующими, если для них

$$V_t \geqslant g_t, \quad t = 1, \ldots, N.$$

Как и в статье [3] будем рассматривать класс $\mathcal{P}_t(\cdot)$ — смешанное расширение класса чистых стратегий. Обозначим

$$\rho_t(\overline{x}_{t-1}) = \inf_{h \in D_t(\overline{x}_{t-1})} \sup_{Q \in \mathcal{P}_t(\overline{x}_{t-1})} \int \left[V_t^*(\overline{x}_{t-1}, y + x_{t-1}) - hy \right],$$

Величина $\rho_t(\overline{x}_{t-1})$ входит в уравнения Беллмана-Айзекса для смешанных стратегий, которые можно записать в сокращенном виде (см. [2]):

$$V_N^*(\overline{x}_N) = g_N(\overline{x}_N),$$
$$V_{t-1}^*(\overline{x}_{t-1}) = g_{t-1}(\overline{x}_{t-1}) \vee \rho_t(\overline{x}_{t-1}), \quad t = 1, \ldots, N.$$

Зафиксируем историю цен и обозначим

$$f(y) = V_t^*(x_0, \ldots, x_{t-1}, y + x_{t-1}).$$

Потребуем, чтобы $f(y)$ была полунепрерывна сверху.
Приведем важное определение из [4].

Определение 6 *Под арбитражной возможностью на шаге $t \in \{1, \ldots, N\}$ будем понимать следующее:*

1. *Найдется допустимая стратегия $h^* \in D_t(\cdot)$, такая что $h^* y \geqslant 0$ для всех $y \in K_t(\cdot)$.*
2. *Найдется $y^* \in K_t(\cdot)$, такое что $h^* y^* > 0$.*

Тогда, согласно [5], при условии отсутствия арбитражных возможностей и торговых ограничений, верно следующее:

$$\rho_t(\overline{x}_{t-1}) = \sup_{Q \in \mathcal{P}^*(K_t(\cdot)) : \int yQ(dy) = 0} \int V_t^*(x_0, \ldots, x_{t-1}, y + x_{t-1}) Q(dy). \tag{3}$$

Здесь $\mathcal{P}_t^*(K_t(\cdot))$ — класс мер, сосредоточенных на конечном множестве точек из $K_t(\cdot)$.

Но, согласно [3], в условиях приведенных выше предположений, верно

$$\sup_{Q \in \mathcal{P}^*(K_t(\cdot)) : \int yQ(dy) = 0} \int V_t^*(x_0, \ldots, x_{t-1}, y + x_{t-1}) Q(dy) = \hat{f}_t(0), \tag{4}$$

где $\hat{f}_t$ — вогнутая оболочка функции f на рассматриваемом компакте X_t — множестве возможных значений цены актива в момент времени t.

В итоге, из формул (3) и (4) получаем:

$$\rho_t(\overline{x}_{t-1}) = \hat{f}_t(0). \tag{5}$$

Таким образом, от задачи вычисления цены опциона приходим к задаче построения вогнутой оболочки функции.

4 Выпуклая оболочка в двумерном пространстве

В большинстве рассматриваемых нами источниках алгоритм построения выпуклой оболочки приведен для задачи, где на вход подается некоторое неупорядоченное множество точек, но в нашей задаче большую роль играет именно то, что мы строим выпуклую оболочку для функции, которая задана на некоторой упорядоченной сетке. То есть набор точек, подаваемый на вход алгоритму упорядочен. Подобные особенности нашей задачи мы и будем учитывать при составлении алгоритмов. Таким образом, мы получим несколько усовершенствованные и адаптированные под нашу задачу алгоритмы.

Итак, пусть задана функция $f : \mathbb{R} \to \mathbb{R}$. Пусть также даны точки $\{x_1, \ldots, x_N\}$, при этом $x_1 < x_2 < \cdots < x_N$. Точки на плоскости обозначим через p_i:

$$p_i = (x_i, f(x_i)),\ i = 1, \ldots, N.$$

4.1 Алгоритм Джарвиса

Этот алгоритм также известен, как алгоритм «заворачивания подарка». Такая аналогия делает его интуитивно понятным. Входная последовательность точек представляет собой подарок, который мы заворачиваем в оболочку. При этом подарочная упаковка натягивается на крайние точки множества.

Приведем более формальное описание. Построим этот алгоритм рекуррентно. Пусть $p_{i_{k-1}}, p_{i_k}$ — вычисленные на предыдущем шаге алгоритма точки выпуклой оболочки. Тогда следующей точкой станет точка $p \in \{p_{i_k+1}, \ldots, p_N\}$, максимизирующая угол $\angle p_{i_{k-1}} p_{i_k} p$. Если точек-максимизаторов несколько, то будем брать ту, у которой координата по оси абсцисс самая большая. Зададим точки p_{i_0}, p_{i_1} следующим образом: $p_{i_0} = (x_1, f(x_1) + 1), p_{i_1} = p_1$. Такой выбор начальных точек объясняется тем, что точка p_1 — крайняя, а точка p_{i_0} входит в надграфик, но, поскольку задача состоит в построении $\operatorname{conv} f$, а не $\operatorname{conv}(\operatorname{epi} f)$, она нужна только для подсчета угла и не будет входить в результирующую последовательность точек. Алгоритм заканчивает свою работу, когда в результирующую последовательность добавлена последняя точка $p_{i_h} = p_N$.

В классической версии алгоритма на k-ой итерации вычисление максимума требует сравнения углов для каждой еще не включенной в результирующую последовательность точки. В этом сучае понадобится провести $N \cdot h$ сравнений (h — количество точек в выпуклой оболочке). В нашем алгоритме все точки, расположенные до выбранной на k-ой итерации не учитываются в сравнении на $(k+1)$-ой, поскольку на данном этапе можно точно сказать, что они являются внутренними. Это может дать небольшой прирост в скорости работы программы. Точнее, в случае если точки выпуклой оболочки распределены по начальной последовательности равномерно, такой ход позволит сократить до

$$\sum_{k=1}^{h-1} \frac{N}{h} \cdot k = \frac{N}{h} \cdot \frac{(h-1)h}{2} = \frac{N(h-1)}{2}$$

операций, то есть ускорить программу почти в два раза. Однако в худшем случае, если все точки расположены на первых h позициях, никакого преимущества от этого действия мы не получим. В любом из этих случаев сложность алгоритма будет составлять $O(nh)$.

4.2 Алгоритм Грэхема

Для данного алгоритма потребуется стек. Считаем, что в начальный момент времени он пустой.

Опишем алгоритм.

1. Добавим в стек точку $p_1 = (x_1, f(x_1))$.
2. Добавим в стек точку $p_2 = (x_2, f(x_2))$.
3. Рассматриваем очередную точку $p_i = (x_i, f(x_i))$ из последовательности $(p_3, \ldots, p_N)$
4. Проверяем, образуют ли последние две точки из стека и рассматриваемая точка последовательности левый или правый поворот.
   - Если в стеке осталась только одна точка, то рассматриваемая на данном шаге точка заносится в стек, мы переходим к шагу 3 и рассматриваем следующую точку.
5. Если они образуют правый поворот, то точка, находящаяся на вершине стека лежит внутри выпуклой оболочки. Тогда мы убираем ее из стека и возвращаемся к шагу 4.
6. Иначе, добавляем рассматриваемую точку в стек, переходим на шаг 3 и рассматриваем следующую точку.
7. Когда все точки рассмотрены, алгоритм заканчивает свою работу. Результатом его работы будет содержимое стека.

В классическом алгоритме на вход подается некоторое неупорядоченное множество точек $\mathbb{R}^2$, поэтому в качестве первого шага предполагается сортировка точек, в связи с этим он имеет сложность $O(n \log n)$. Но в нашем случае мы считаем, что точки $(x_1, \ldots, x_N)$ уже находятся в порядке возрастания, как стандартная сетка по оси X, поэтому сложность такого алгоритма будет $O(n)$. Это объясняется тем, что каждая точка может быть расссмотрена не более двух раз: один раз, когда она входит в левый поворот и добавляется в стек, и один раз, когда она входит в правый поворот, то есть лежит внутри выпуклой оболочки, и убирается из стека.

4.3 Сравнение двумерных алгоритмов

Как видно, у алгоритма Грэхема есть очевидное преимущество в связи с тем, что его сложность линейна. Но, учитывая то, что операция определяющая вхождение вершины в левый поворот сложнее чем вычисление угла, и учитывая предложенную для алгоритма Джарвиса модификацию, возможно, что в случаях, когда количество точек выпуклой оболочки h мало, алгоритм Джарвиса будет работать быстрее.

Сравнение алгоритмов будем проводить в интересующей нас области. Обратимся к пункту 3. Для простоты будем считать, что многозначное отображение

$$K_t(\cdot) = [d, u] = \text{const}, \quad d < 0 < u.$$

Тогда цена актива X_t в момент времени t варьируется от $X_{t-1} + d$ до $X_{t-1} + u$. На этом промежутке и нужно будет вычислять вогнутую оболочку. Для численного метода нужно будет задать сетку K_{X_t} на этом сегменте. Количество точек в этой сетке будем обозначать R.

Будем брать различные функции выплат по опционам, вычислять их цену в зависимости от стартовой цены актива и сравнивать скорость работы предложенных методов для различных параметров. Есть три основных параметра, влияющих на скорость и точность работы алгоритма. Первый параметр — количество точек в сетке стартовых цен актива (будем обозначать его M), соответственно чем больше здесь точек, тем точнее будет результирующий график, но и алгоритм будет работать дольше. Второй параметр — количество шагов алгоритма (будем обозначать его N), он отвечает за величину промежутка времени между продажей опциона и датой экспирации, соответственно чем больше этот промежуток, тем дольше работает программа. И третий параметр — количество точек R в сетке K_{X_t}.

1. Сначала рассмотрим бинарные call опционы. Для них функция выплат имеет вид

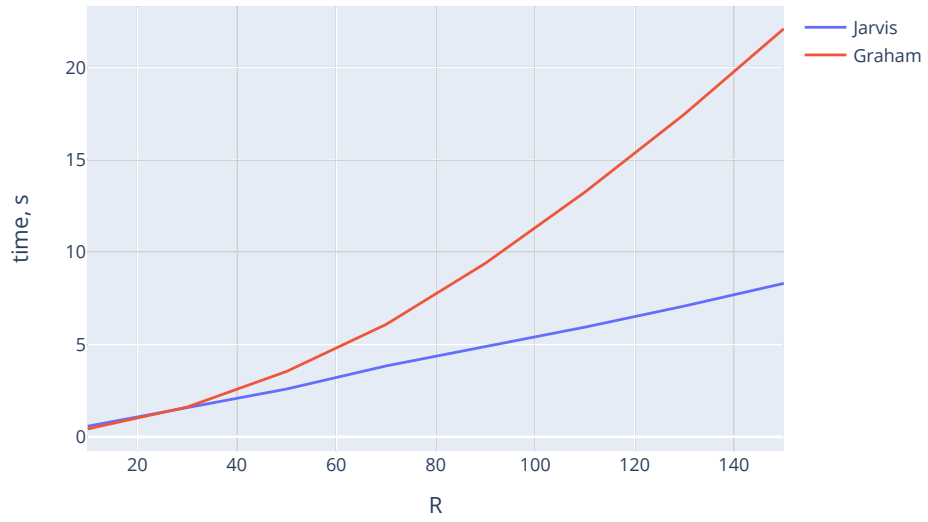
$$g_N(x) = \begin{cases} 0, & x < K; \\ 1, & x \geqslant K. \end{cases}$$

Здесь K — страйк опциона. Возьмем $u = 0.4,\ d = -0.2$. Значение страйка K возьмем равным 3. Зафиксируем количество точек $M = 300$ и будем варьировать количество шагов N от 1 до 3. Количество точек в сетке K_{X_t} возьмем $R = 120$. Ниже продемонстрированы графики зависимости цены опциона от стартовой цены актива, полученные в результате работы программы.

Как видно по графикам, кривая цены имеет один разрыв, после которого она имеет гладкую форму с N - 1 перегибами.

Однако, в случае, если количество точек R в сетке мало, то возникает заметная неточность, выражаемая в «ступенчатой» структуре кривой. Это продемонстрировано на следующем графике. Здесь количество шагов $N = 2$, а количество точек $R = 30,\ 150$.

Теперь построим график сравнения скорости работы программы для различных методов построения вогнутой оболочки. Зафиксируем параметры $N = 2$, $M = 300$ и будем варьировать R от 10 до 150.



Как видно по графику, при малых значениях R алгоритм Грэхема быстрее, но с увеличением R ситуация меняется и лидирующим становится алгоритм Джарвиса. При малых значениях R алгоритм Джарвиса проигрывает, потому что он требует чуть больше инициализационных действий. А для больших R он срабатывает быстрее в силу того, что функции, для которых вычисляется оболочка содержат мало крайних точек, соответственно, сложность алгоритма становится почти линейной, при этом каждая проводимая операция проще чем у алгоритма Грэхема.

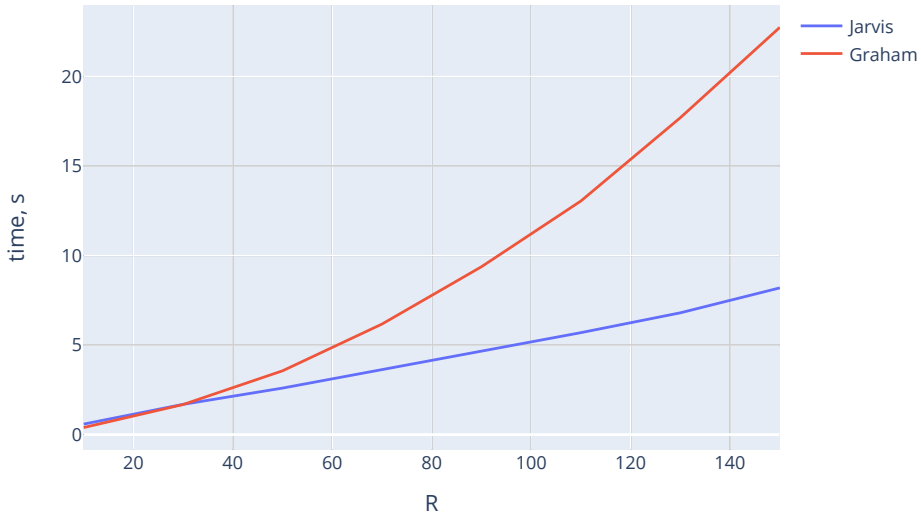
2. Теперь рассмотрим ванильные call опционы. Для них функция выплат имеет вид

$$g_N(x) = \max\{(x - K), 0\}$$

Возьмем $u = 0.5,\ d = -0.2$. Значение страйка K возьмем равным 3. Зафиксируем количество точек $M = 300$ и будем варьировать количество шагов N от 1 до 3. Количество точек в сетке K_{X_t} возьмем $R = 50$. Ниже продемонстрированы графики зависимости цены опциона от стартовой цены актива, полученные в результате работы программы.

Как видно по графикам, кривая цены кусочно-линейна. Причем она совпадает с функцией g_N всюду за исключением одного интервала, на котором также, как и в случае с бинарными опционами, имеет N - 1 перегибов.

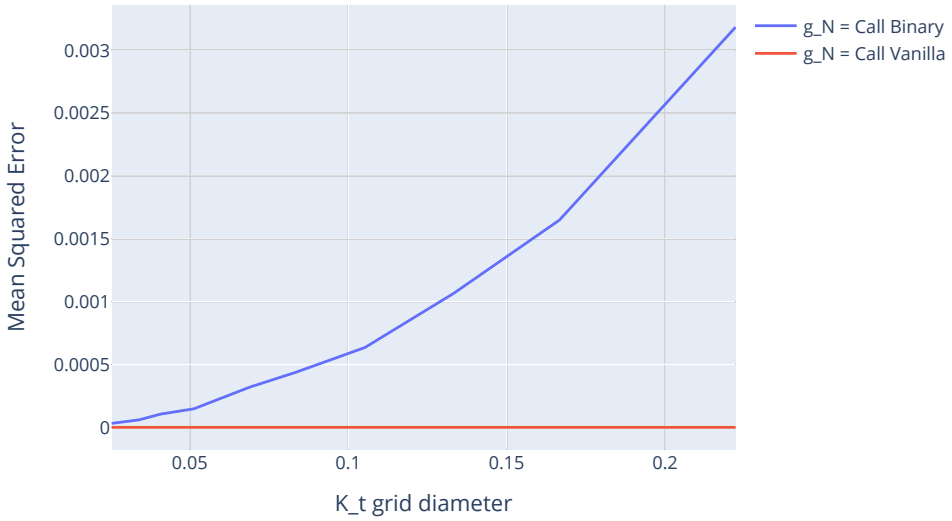
Теперь построим график сравнения скорости работы программ для различных методов построения вогнутой оболочки.



Здесь функции, для которых считается оболочка также содержат мало крайних точек, поэтому опять же выигрывает алгоритм Джарвиса.

4.4 Сходимость численных методов

Ранее было отмечено, что для бинарных опционов при малом количестве точек R в сетке K_{X_t} имеет место неточность вычислений. Попробуем оценить величину ошибки. Рассмотрим подробнее случай $N = 1$. В качестве точного решения будем брать результат вычислений при достаточно большом количестве точек $R = 300$ и вычислять среднеквадратическую ошибку с результатом, полученным при меньшем количестве точек. На следующем графике представлено сравнение сходимости численных методов для случаев бинарных и ванильных опционов. По оси X — диаметр разбиения сетки K_{X_t}, по оси Y — среднеквадратическая ошибка.



Как видно, для ванильных опционов разбиение сетки не играет роли. Это связано с тем, что функция выплат для ванильных опционов выпукла на любом сегменте. Поэтому для вычисления вогнутой оболочки на множестве K_{X_t} достаточно всего двух крайних точек, соответственно и получается, что ошибка нулевая и для ванильных опционов численный метод сходится сразу, чего нельзя сказать о бинарных опционах.

4.5 Влияние волатильности на цену опциона

Разность параметров d и u задает меру непредсказуемости движения цены актива на рынке. Соответственно, чем больше будет неопределенность, тем больший риск на себя берет продавец опциона, а тогда и цена опциона должна быть выше. Проверим справедливость данного свойства для ванильных опционов при количестве шагов $N = 2$.

Так и получилось. Цена более волатильного актива оказалась больше.

Теперь продемонстрируем это же свойство для бинарных опционов.

5 Алгоритмы в d-мерном пространстве

Теперь рассмотрим алгоритмы в d-мерном пространстве. Алгоритм быстрой оболочки применяется для более общего случая построения оболочки по точкам, а преобразование Лежандра-Фенхеля применяется только для функций.

Сравнение методов будем проводить в трехмерном пространстве.

5.1 Алгоритм быстрой оболочки (QuickHull)

Алгоритм быстрой оболочки может применяться в общем случае в d - мерном пространстве $\mathbb{R}^d$.

Везде далее подпространство размерности $(d-1)$ будем называть *гранью*, а подпространство размерности $(d-2)$ — *ребром*. В рамках алгоритма каждая грань будет представлена как единичный вектор нормали и смещение грани относительно начала координат. Также потребуется ввести расстояние со знаком от точки до грани. Оно будет вычисляться как скалярное произведение в $\mathbb{R}^d$ точки и вектора нормали плюс смещение грани. Если полученное расстояние окажется больше нуля, будем говорить, что точка расположена над гранью, если меньше нуля, то под гранью. Приведем важную теорему, на которой основана работа алгоритма.

Теорема 1 *Пусть H — выпуклая оболочка в $\mathbb{R}^d$ и пусть p будет точкой из $(\mathbb{R}^d - H)$. Тогда F является гранью* $\mathrm{conv}(p \cup H)$ *если и только если*

1. *F — грань из H и p находится под F или*
2. *F — не грань из H и F состоит из точки p и вершин ребра в H, полученного пересечением граней, одна из которых расположена над точкой p, а другая — под ней.*

Перейдем к описанию алгоритма.

- построим симплекс из $d+1$ точки (точки выбираются таким образом, чтобы они все не лежали на одной грани);
- для каждой грани F:
  - для каждой нераспределенной точки p:

    если p расположена над F:

    добавить p к *множеству внешних точек*;
- для каждой грани F с непустым *множеством внешних точек*:
  - выбрать самую удаленную точку p из *множества внешних точек* F;
  - инициализировать множество V гранью F;
  - для всех непосещенных соседей N граней из V:

если p расположена над N:

добавить N в множество V;

– множеством H будем считать множество ребер с границы V;

– для каждого ребра R из H:

построить новую грань из R и p;

соединить эту грань с соседями;

– для каждой новой грани F':

для каждой нераспределенной точки q из *множества внешних точек* грани из V:

если q расположена над F', добавить точку q к *множеству внешних точек* F';

– удалить грани из множества V;

Более подробно об этом методе можно прочитать в [9].

Данный алгоритм вычисляет выпуклую оболочку множества, а не функции. Соответственно, чтобы адаптировать его для нашей задачи, необходимо провести ряд дополнительных шагов.

Итак, пусть необходимо вычислить выпуклую оболочку функции f. Множество, на котором осуществляется вычисление выпуклой оболочки функции будем обозначать X. Пусть алгоритму на вход подаются точки из X со значениями $f(X)$. Опишем принцип работы алгоритма.

1. Численно найдем максимальное значение функции f на множестве X. Обозначим его M.

2. Выпуклую оболочку функции, исходя из определения, можно построить путем вычисления выпуклой оболочки ее надграфика. Для численной реализации мы не можем взять весь надграфик, поскольку он является бесконечным множеством, поэтому возьмем его конечное подмножество, содержащее всю необходимую информацию для построения выпуклой оболочки функции:

$$F = \{(x, y) \in X \times \mathbb{R} |\ M + 1 \geqslant y \geqslant f(x)\}.$$

Это осуществляется путем добавления к исходному множеству точек из X со значениями $f(X)$ точек с границы множества X со значениями $M + 1$.

3. Вычислим выпуклую оболочку для множества F при помощи описанного выше алгоритма быстрой оболочки (QuickHull).

4. Из полученного множества точек уберем те, которые были добавлены на шаге 2. Оставшиеся в результате точки и будут составлять выпуклую оболочку функции f.

5.2 Преобразование Лежандра-Фенхеля

Для начала введем само понятие преобразования Лежандра-Фенхеля.
В качестве $\overline{\mathbb{R}}$ будем обозначать расширенное вещественное пространство, т.е.

$$\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}.$$

Определение 7 *Пусть* $f : \mathbb{R}^n \to \overline{\mathbb{R}}$. *Тогда функция*

$$f^* : \mathbb{R}^n \to \overline{\mathbb{R}},$$

$$f^*(\xi) = \sup_{x \in \mathbb{R}^n} \left[\langle x, \xi \rangle - f(x) \right],$$

где $\langle \cdot, \cdot \rangle$ *— скалярное произведение в* $\mathbb{R}^n$, *называется преобразованием Лежандра-Фенхеля функции* f.

Для дальнейшего изложения нам потребуется ряд теорем. Доказательства приведенных ниже утверждений можно найти в [8].

Теорема 2 *Пусть* $f : \mathbb{R}^n \to \overline{\mathbb{R}}$. *Тогда имеем:* $f^{**} = \mathrm{cl}(\mathrm{conv}\, f)$.

Эта теорема является причиной, по которой мы используем преобразование Лежандра-Фенхеля для вычисления выпуклой оболочки функции. Таким образом, дважды применив преобразование Лежандра-Фенхеля к функции, мы получим ее выпуклую оболочку.

Теорема 3 *Пусть* $f : \mathbb{R}^n \to \overline{\mathbb{R}}$. *Обозначим* f^{*i} *преобразование Лежандра-Фенхеля по* i*-му измерению, т.е.*

$$f^{*i} : \mathbb{R}^n \to \overline{\mathbb{R}},$$

$$f^{*i}(x_1, \ldots, x_{i-1}, \xi_i, x_{i+1}, \ldots, x_n) = \sup_{x_i \in \mathbb{R}} \left[x_i \xi_i - f(x_1, \ldots, x_i, \ldots, x_n) \right].$$

Тогда имеем

$$f^* = \left(- \left(\ldots \left(- \left(-f^{*1} \right)^{*2} \right)^{*3} \ldots \right)^{*(n-1)} \right)^{*n}.$$

Эта теорема дает нам возможность считать преобразование Лежандра-Фенхеля многомерной функции путем подсчета одномерных преобразований по каждому измерению.

Для численного вычисления выпуклой оболочки функции, множество оцениваемых точек должно быть конечным. В связи с этим возникает необходимость ввести дискретный аналог преобразования Лежандра-Фенхеля.

Определение 8 *Пусть* $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ *и* $\Omega \subseteq \mathbb{R}^n$. *Обозначим за* $f_\Omega(x) : \mathbb{R}^n \to \overline{\mathbb{R}}$ *такую функцию, что*

$$f_\Omega(x) = \begin{cases} f(x), & x \in \Omega \\ +\infty, & \textit{иначе} \end{cases}.$$

Определение 9 *Пусть* $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ *и* $\Omega_N \subset \mathbb{R}^n$ *— конечно. Тогда функция* $f^*_{\Omega_N}$ *— дискретное преобразование Лежандра-Фенхеля функции* f *по множеству* Ω_N.

Определение 10 *Обозначим* $I_{X_N}(f)$ *функцию, которая совпадает с линейной интерполянтой* f *на интервале* $[x_1, x_N]$ *в узлах* X_N *и равна* $+\infty$ *иначе.*

Теорема 4 *Пусть* $f : \mathbb{R} \to \overline{\mathbb{R}}$ *и пусть* $X_N \subset \mathbb{R}$ *— конечно. Тогда* $[\operatorname{conv} I_{X_N}(f)]^* = [I_{X_N}(f)]^* = f^*_{X_N}$.

Теорема 5 *Пусть задан конечный набор точек* $P_i = (x_i, y_i) \in \mathbb{R}^2$, $i = 1, \ldots, N$, *таких, что* $x_1 < \cdots < x_N$ *и даны* $c_0, c_N \in \overline{\mathbb{R}}$. *Пусть* g *— кусочно-линейная непрерывная функция, которая интерполирует точки* P_i *и продлевается влево и вправо с коэффициентами наклона (возможно бесконечными)* c_0 *и* c_N *соответственно, то есть*

$$g(x) = \begin{cases} y_1 + c_0(x - x_1), & x < x_1 \\ y_i + c_i(x - x_i), & x_i \leqslant x \leqslant x_{i+1},\ i = 1, \ldots, N-1 \\ y_N + c_N(x - x_N), & x > x_N \end{cases}$$

где значения c_i *задаются следующим образом:*

$$c_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad i = 1, \ldots, N-1.$$

Пусть g *— выпукла, то есть* $c_0 \leqslant \cdots \leqslant c_N$. *Тогда*

$$g^*(\xi) = \begin{cases} +\infty, & \xi < c_0 \text{ или } \xi > c_N, \\ x_i\xi - y_i, & c_{i-1} \leqslant \xi \leqslant c_i,\ i = 1, \ldots, N. \end{cases}$$

Последние две теоремы позволяют считать дискретное преобразование Лежандра-Фенхеля для одномерной функции.

Для вычисления выпуклой оболочки необходимо дважды взять дискретное преобразование Лежандра-Фенхеля от функции. Второе преобразование мы считаем на интересующей нас сетке (в точках, где мы хотим вычислить выпуклую оболочку) Ω_N, которую мы можем выбрать произвольным образом. Далее будем называть эту сетку основной. Но когда мы берем преобразование в первый раз, выбор сетки, на которой будут осуществляться вычисления, не очевиден и играет критическую роль. Эту сетку будем называть сопряженной и обозначать S_N. Важность выбора сопряженной сетки объясняется тем, что в общем случае $\left(f^*_{\Omega_N}\right)^*_{S_N} \neq \left(f^*_{\Omega_N}\right)^* = \operatorname{conv} f_{\Omega_N}$ (см. [8]). Тем не менее, приведенная далее теорема покажет, что существует такая сетка S_N, что $\left(f^*_{\Omega_N}\right)^*_{S_N} = \operatorname{conv} f_{\Omega_N}$ для всех $x \in \Omega_N$.

Для следующей теоремы нам понадобятся понятия субградиента и субдифференциала.

Определение 11 *Вектор ξ называется субградиентом функции f в точке x, если*

$$f(z) \geqslant f(x) + \langle \xi, z - x \rangle \quad \forall z.$$

Определение 12 *Субдифференциалом функции f в точке x называется множество всех субградиентов функции f в этой точке. Субдифференциал обозначается $\partial f(x)$.*

Итак, приведем формулировку теоремы о выборе сопряженной сетки. Для простоты изложения рассмотрим случай, когда функция f — двумерная. Дальнейшие результаты легко переносятся на многомерный случай.

Теорема 6 *Пусть $f : \mathbb{R}^2 \to \overline{\mathbb{R}}$ и пусть $\Omega_N = X_N \times Y_N$ — конечное подмножество $\mathbb{R}^2$. Пусть f конечна на Ω_N. Фиксируем $y \in \mathbb{R}$, пусть*

$$g_y := f_{\Omega_N}(\cdot, y),$$
$$\xi_y^- := \max \partial \operatorname{conv} g_y(\min X_N), \quad \xi^- := \min_{y \in Y_N} \xi_y^-,$$
$$\xi_y^+ := \min \partial \operatorname{conv} g_y(\max X_N), \quad \xi^+ := \max_{y \in Y_N} \xi_y^+.$$

Теперь фиксируем $x \in \mathbb{R}$ и пусть

$$h_x := f_{\Omega_N}(x, \cdot),$$
$$\eta_x^- := \max \partial \operatorname{conv} h_x(\min Y_N), \quad \eta^- := \min_{x \in X_N} \eta_x^-,$$
$$\eta_x^+ := \min \partial \operatorname{conv} h_x(\max Y_N), \quad \eta^+ := \max_{x \in X_N} \eta_x^+.$$

Тогда множество $S = C \times D := [\xi^-, \xi^+] \times [\eta^-, \eta^+]$ содержит оптимальную относительно Ω_N сопряженную сетку. В частности это значит, что для всех $(x, y) \in \Omega_N$

$$\operatorname{conv} f_{\Omega_N}(x, y) = \left(f^*_{\Omega_N}\right)^*_S (x, y).$$

Чтобы задать сопряженную сетку рассмотрим подробнее выражение

$$\xi_y^- = \max \partial \operatorname{conv} g_y(\min X_N). \tag{6}$$

Далее нам потребуется знать $\operatorname{conv} g_y$, вычислить ее мы можем при помощи описанного ранее алгоритма Грэхема. Считаем, что элементы множества $X_N = \{x_1, \ldots, x_N\}$ упорядочены, поэтому $minX_N = x_1$. Раскроем выражение (6) по определению субдифференциала, получим

$$\max \partial \operatorname{conv} g_y(x_1) = \max \{\xi : \operatorname{conv} g_y(z) \geqslant \operatorname{conv} g_y(x_1) + \xi(z - x_1), \quad \forall z \in \Omega_N\} =$$

$$= \{\text{Отметим, что при } z = x_1 \text{ данное неравенство верно для любых } \xi\} =$$

$$= \max \left\{ \xi : \xi \leqslant \frac{\operatorname{conv} g_y(z) - \operatorname{conv} g_y(x_1)}{z - x_1}, \quad \forall z \in \Omega_N \right\} = \{\text{В обозначениях Теоремы 5}\} =$$

$$= \max \{\xi : \xi \leqslant c_i, \quad \forall i = 1, \ldots, N-1\} =$$

$$= \{\text{Поскольку функция выпуклая, то } c_1 < c_2 < \cdots < c_{N-1}\} = c_1$$

Соответственно теперь для получения ξ^- необходимо взять минимум ξ_y^- по $y \in Y_N$. Аналогично получаем, что $\xi_y^+ = \min \partial \operatorname{conv} g_y(\max X_N) = c_{N-1}$. Теперь можно взять равномерное разбиение отрезка $[\xi^-, \xi^+]$ и посчитать в точках этой сетки преобразование Лежандра-Фенхеля.

Для других измерений необходимо выполнить ту же процедуру и, пользуясь Теоремой 3, получить многомерное преобразование Лежандра-Фенхеля. Затем остается только посчитать второе преобразование в точках основной сетки Ω_N и аналогичным образом воспользоваться Теоремой 3. Таким образом, получим выпуклую оболочку функции f.

5.3 Сравнение трехмерных алгоритмов

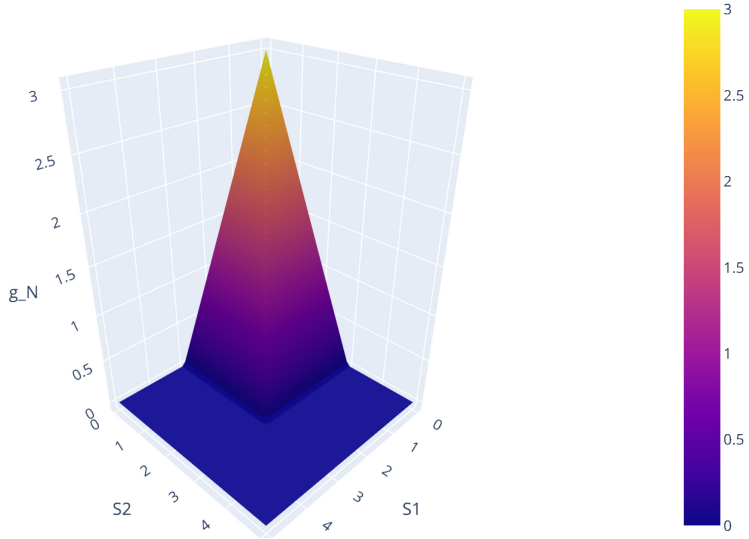
Аналогично с пунктом 4.3 проведем сравнение описанных выше алгоритмов. Будем считать, что множество K_t состовляет прямоугольник:

$$K_t = [d_1, u_1] \times [d_2, u_2].$$

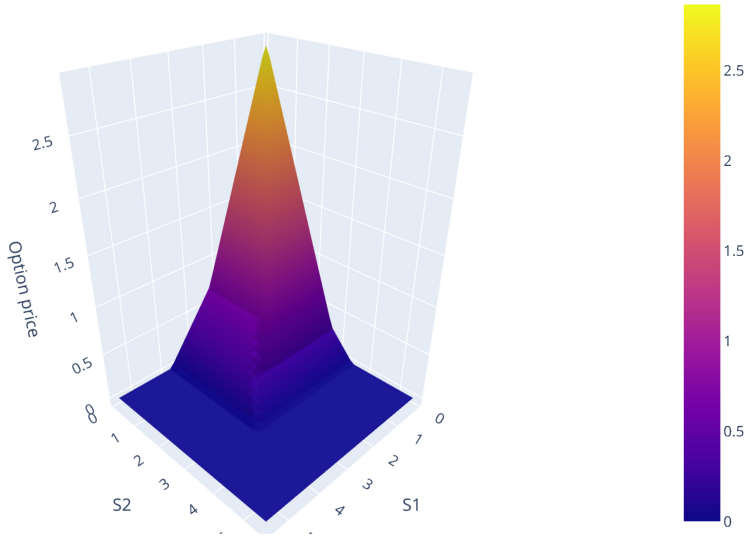
Рассмотрим опционы put-on-max. Для них функция выплат имеет вид

$$g_N(S_1, S_2) = \max\{K - \max\{S_1, S_2\}, 0\}.$$

На графике эта функция выглядит следующим образом.

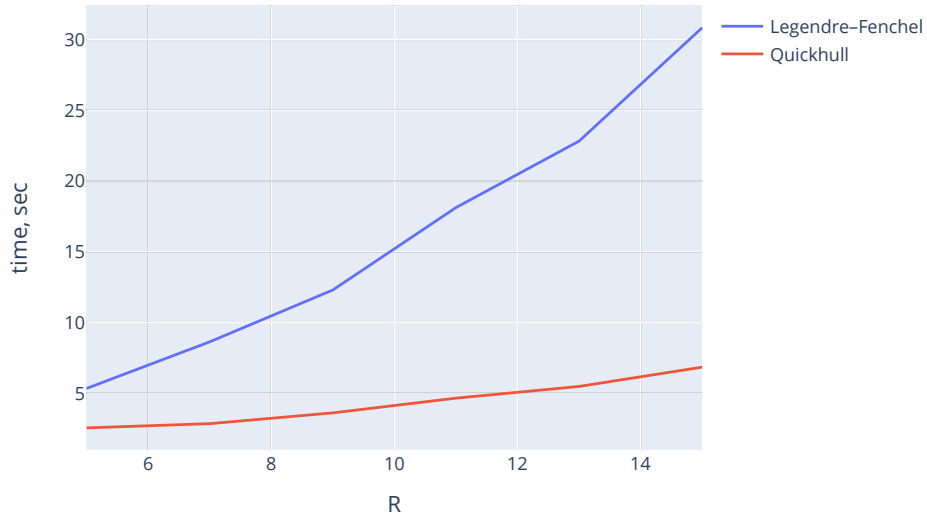


Для первого актива S_1 возьмем $u_1 = 0.7,\ d_1 = -0.8$. Для второго актива S_2 возьмем $u_2 = 0.3,\ d_2 = -0.5$. Значение страйка K возьмем равным 3. Ниже продемонстрирован график зависимости цены опциона от стартовых цен активов, полученный в результате работы программы при $N = 1,\ M = 40,\ R = 15$.



Данный случай похож на результат для аналогичной функции выплат в двумерном пространстве (см. п. 4.3, случай ванильных опционов). Полученная поверхность состоит из плоскостей. Она совпадает со своей функцией выплат за исключением области в районе основания. Там плоскости имеют меньший угол наклона. Отметим, что в этом участке плоскость вдоль оси S_1 расположена выше плоскости вдоль оси S_2, то есть цена опциона для менее стабильного актива получилась больше.

Теперь построим график сравнения скорости работы программ для различных методов построения вогнутой оболочки.



Как видно по графику, алгоритм быстрой оболочки работает быстрее и с увеличением количества точек разница в скорости растет.

6 Заключение

Мы протестировали различные численные методы для нашей задачи суперхеджирования. В трехмерном случае получилось, что алгоритм быстрой оболочки работает быстрее при любом количестве шагов и любом количестве точек сетки. В двумерном же случае оказалось, что алгоритм Грэхема, несмотря на свою линейную сложность, не оказался самым быстрым. Для выбранных нами функций алгоритм Джарвиса подошел лучше.

Таким образом, при выборе метода для моделирования стоит обращать внимание на то, к какому классу принадлежит рассматриваемая функция и насколько много крайних точек содержится в ее оболочке и на основании этого выбирать соответствующий метод.

Список литературы

[1] Смирнов С. Н. Гарантированный детерминистический подход к суперхеджированию: модель рынка, торговые ограничения и уравнения Беллмана – Айзекса // Математическая Теория Игр и ее Приложения. 2018. Том 10. №4. С. 59-99.

[2] Смирнов С.Н. Гарантированный детерминистский подход к суперхеджированию: смешанные стратегии и игровое равновесие // Математическая Теория Игр и ее Приложения. 2020. Том 12. №1. С. 60—90.

[3] Смирнов С.Н. Гарантированный детерминистский подход к суперхеджированию: наиболее неблагоприятные сценарии поведения рынка и проблема моментов // Математическая Теория Игр и ее Приложения 2020 (в печати).

[4] Смирнов С.Н. Гарантированный детерминистский подход к суперхеджированию: свойства «безарбитражности» рынка // Математическая Теория Игр и ее Приложения. 2019. Том 11. №2. С. 68–95.

[5] Smirnov S. N. A Guaranteed Deterministic Approach to Superhedging: A Game Equilibrium in the Case of No Trading Constraints // Journal of Mathematical Sciences 2020 (in print).

[6] Graham, R.L. An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set // Information Processing Letters. 1972. Volume 1. №4. P. 132-133.

[7] Jarvis, R. A. On the identification of the convex hull of a finite set of points in the plane // Information Processing Letters. 1973. Volume 2. №1. P. 18-21.

[8] Lorenzo Contento. The Discrete Legendre-Fenchel Transform and its application to phase separation in electrolytes// Hyper Articles en Ligne. 2012. P. 12 - 39.

[9] Barber, C. B., D.P. Dobkin, and H.T. Huhdanpaa. The Quickhull Algorithm for Convex Hulls // ACM Transactions on Mathematical Software. 1996. Volume 22. №4. P. 469 - 474.