Санкт–Петербургский государственный университет
Кафедра математической теории игр и статистических решений

Новгородцев Виталий Александрович

Магистерская диссертация

Применение стохастических моделей к финансовым продуктам

Направление 01.04.02
Прикладная математика и информатика
Магистерская программа:
Исследование операций и системный анализ

Научный руководитель:
кандидат физико-математических наук,
старший преподаватель МТИиСР,
Кумачева Сурия Шакировна

Рецензент:
управляющий директор,
член совета директоров,
Закрытое акционерное общество
«Инвестиционная компания «Ленмонтажстрой»»,
Клещев Александр Игоревич

Санкт-Петербург
2022 г.

Содержание

Введение

В последние годы финансовые рынки, в частности, фондовые, получили бурное развитие. Нередко клиенты просят у банка продать определенный дериватив, который не торгуется на бирже. Поэтому банку необходимо оценить дериватив не слишком дорого, чтобы клиент продолжил сотрудничество, и не слишком дешево, чтобы получить неплохую прибыль.

При оценивании деривативa возникают трудности в выборе моделей, их калибровке [21] и последующей оценке стоимости деривативa при помощи методов Монте-Карло или решения уравнений в частных производных. В связи с этим целью данной работы будет применение стохастических моделей к оцениванию деривативов в случае их зависимости от группы акций и их сравнении на примере структурных нот.

Вопросу калибровок моделей к ценам европейских опционов и оцениванию опционов посвящено огромное количество работ, например [23], [25], [11]. В работах [13], [26] рассмотрены методы калибровок корреляционной структуры в случае многомерной моделей Хестона и последующее моделирование методом Монте-Карло. В случае многомерной модели Бейтса возникает вопрос в моделировании совместной динамики прыжков генерируемых совокупностью пуассоновских процессов. Поэтому одной из задач данной работы является построение метода моделирования совместной динамики прыжков и амплитуд прыжков в многомерной модели Бейтса и последующее сравнение с методом моделирования без учета совместной динамики.

Глава 1. Теоретическая часть

Дериватив - это ценная бумага [18], цена которой зависит от стоимости других базовых переменных. Обычно такими базовыми переменными являются цены различных рыночных активов, например, акций или на нефть, но могут быть базовые переменные такие как количество выпавшего снега на определенном горнолыжном курорте[18] .

В данной работе в качестве базового актива будем рассматривать цены акций. Цена акции представляет собой в каждый будущий момент времени некоторое случайное значение, поэтому естественно попытаться описать ее динамику с помощью стохастических моделей, которые будут учитывать тот или иной аспект ее поведения. Перед тем, как рассмотреть такие модели, введем несколько определений.

Случайным процессом $S(t)$ назовем [15] семейство случайных величин заданных на вероятностном пространстве $(\Omega, \mathfrak{F}, \mathbb{P})$, параметризованных временной переменной $t \in [0, T] \subset \mathbb{R}$, где T - конец рассматриваемого временного промежутка.

Примером случайного процесса может служить динамика цен акций. В некоторых случаях мы можем знать конкретную реализацию случайного процесса до определенного момента времени. В нашем случае мы знаем конкретный путь цены (историческая цена) рассматриваемой акции до сегодняшнего дня, но чтобы получить будущие значения нужно моделировать будущее, согласно распределению цены акции.

Но как учитывать информацию о прошлых значениях? Вначале сделаем разбиение отрезка времени $[0, T]$ на $t_0 = 0 < t_1 < ... < t_m = T$. Учитывать информацию о стохастическом процессе до определенного времени t_i включительно можно с помощью последовательности сигма алгебр, $\mathfrak{F}(t_i) := \sigma(S(t_j) : 1 \leq j \leq i)$, генерируемых последовательностью $S(t_j)$ для $0 \leq j \leq i$. Такая последовательность сигма алгебр называется фильтрацией. Таким образом, фильтрация описывает доступную информацию в момент времени t_i. Когда мы имеем дело с последовательностью времен наблюдения : $t_0, ..., t_i$, мы по сути имеем дело с фильтрацией $\mathfrak{F}(t_0) \subseteq ... \subseteq \mathfrak{F}(t_i)$.

Процесс $S(t)$ называется $\mathfrak{F}(t)$ - измеримым, если для любого времени $\bar{t} \leq t$, реализации этого процесса известны. Например, мы знаем значение цены акций до сегодняшнего дня точно, но мы не знаем будущих цен, тогда можно сказать, что цена акций измерима до сегодняшнего дня включительно. Однако, если у нас есть модель динамики цены акции, например, описываемая стохастическим дифференциальным уравнением (СДУ), то значение будет $\mathfrak{F}(T)$ - измеримо, если нам известно распределение цены финансового контракта на момент времени T.

Случайный процесс $S(t), 0 \leq t \leq T$ называется [15] адаптированным к фильтрации $(\mathfrak{F}(t))_{t\in[0,T]}$, если $\forall t \in [0,T]$

$$\sigma(S(t)) \subseteq \mathfrak{F}(t).$$

Адаптированный процесс означает, что случайный процесс не может заглядывать в будущее. Действительно, если какое-то множество из $\sigma(S(t))$ окажется за пределами элемента фильтрации $\mathfrak{F}(t)$, в котором содержится вся информация до момента времени t, то получается, что это множество будет иметь непустое пересечение с элементом фильтрации $\mathfrak{F}(\bar{t})$, где $\bar{t} > t$, то есть, с элементом, содержащим и информацию о будущем. Если процесс $S(t)$ адаптирован к фильтрации$(\mathfrak{F}(t))_{t\in[0,T]}$, то он будет и $\mathfrak{F}(t)$ - измеримым.

Рассмотрим вероятностное пространство $(\Omega, \mathfrak{F}, \mathbb{P})$. Процесс $S(t)$ при $t \in [0,T]$ называется [1] мартингалом относительно фильтрации $(\mathfrak{F}(t))_{t\in[0,T]}$ в мере $\mathbb{P}$, если

1) процесс $S(t)$ $\mathfrak{F}(t)$ - измерим $\forall t \in [0,T]$;

2) $\mathbb{E}|S(t)| < \infty, t \in [0,T]$;

3) $\mathbb{E}(S(t)|\mathfrak{F}(s)) = S(s)$, $s, t \in [0,T]$, $s \leq t$.

В определении подразумевается, что лучшее предсказание ожидаемого будущего значения мартингала - это текущее значение. Процесс $W(t)$, $t \in [0,T]$ называется [1] винеровским процессом, если:

1) $W(t_0) = 0$, п.н.;

2) процесс $W(t)$, $t \in [0,T]$, имеет независимые приращения;

3) приращения $W(t) - W(s) \sim \mathcal{N}(0,\, t-s)$, для $0 = t_0 \leq s < t$.

Винеровский процесс является мартингалом. [15]

Случайный процесс называется марковским [15], если условное распределение будущих значений зависит только от текущего значения, а не от исторических данных. Применительно к сфере финансов - это означает, что текущая цена акции содержит всю информацию о предыдущих ценах на акцию [15]. Адаптированный процесс $S(t)$ на фильтрованном вероятностном пространстве обладает марковским свойством, если для любой ограниченной измеримой функции g: $\mathbb{R}^n \to \mathbb{R}$,

$$\mathbb{E}[g(S(t))|\mathfrak{F}(s)] = \mathbb{E}[g(S(t))|S(s)], s \leq t.$$

Сберегательным счетом будем называть процентный депозитный счет, открытый в банке или другом финансовом учреждении. Если процентная ставка равна r, то при непрерывном начислении процентов имеем следующую динамику количества денег M(t) на сберегательном счету:

$$dM(t) = rM(t)dt$$

Если в начальный момент времени $t_0 = 0$ у нас M(0)=1 , то в момент времени t у нас будет $M(t) = e^{rt}$. Если же мы хотим, чтобы в момент времени T было $M(T) = 1$, то необходимо чтобы в начальный момент времени $t_0 = 0$ было e^{-rT}, тогда в момент времени t будет $M(t) = e^{-r(T-t)}$. Процесс $\frac{S(t)}{M(t)}$ будем называть дисконтированным процессом, где $dM(t) = rM(t)dt$, при $M(t_0) = 1, t_0 \leq t$. Везде далее в работе считаем, что процентная ставка является константой.

1.1 Модели динамики акций

Перейдем к основным моделям динамик акций. Обозначим цену акции в момент времени t как $S(t)$. Одной из основных задач является задача определения модели поведения цены акций. Так как динамика цен акций представляет стохастический процесс, то в качестве моделей необходимо выбирать стохастические, например, такие как: модель Блэка и Шоулза[10], модель Хестона[17], модель Бэйтса[15] и другие.

1.1.1 Модель Блэка-Шоулза

В популярной модели Блэка и Шоулза динамика цены базового актива $S(t)$ подчиняется геометрическому броуновскому движению:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW^{\mathbb{P}}(t),\ t \in [t_0, T],$$

где $\mu \in \mathbb{R}$ - параметр дрифта, $\sigma > 0$ - волатильность актива, $W^{\mathbb{P}}(t)$ - винеровский процесс под реальной мерой $\mathbb{P}$. В качестве параметра дрифта выбирают безрисковую ставку $\mu = r$. Это делают, чтобы получить процесс, если дисконтировать который, получится мартингал. То есть, чтобы выполнялось:

$$E^{\mathbb{Q}}[\frac{S(t)}{M(t)}|\mathfrak{F}(s)] = \frac{S(s)}{M(s)},\ s \leq t,$$

где $\mathbb{Q}$ - риск-нейтральная мера, то есть, мера, в которой дисконтированная цена акции является мартингалом. Таким образом, получаем процесс под риск-нейтральной мерой $\mathbb{Q}$:

$$dS(t) = rS(t)dt + \sigma S(t)dW^{\mathbb{Q}}(t). \tag{1}$$

Также для дальнейшего удобства работы с динамикой (1) сделаем преобразование $X(t) = log(S(t))$:

$$dX(t) = (r - \frac{1}{2}\sigma^2)dt + \sigma dW^{\mathbb{Q}}(t). \tag{2}$$

Пример траектории модели (1) с параметрами: r =0.05, $\sigma = 0.3$, S(0)=50, T = 3 :

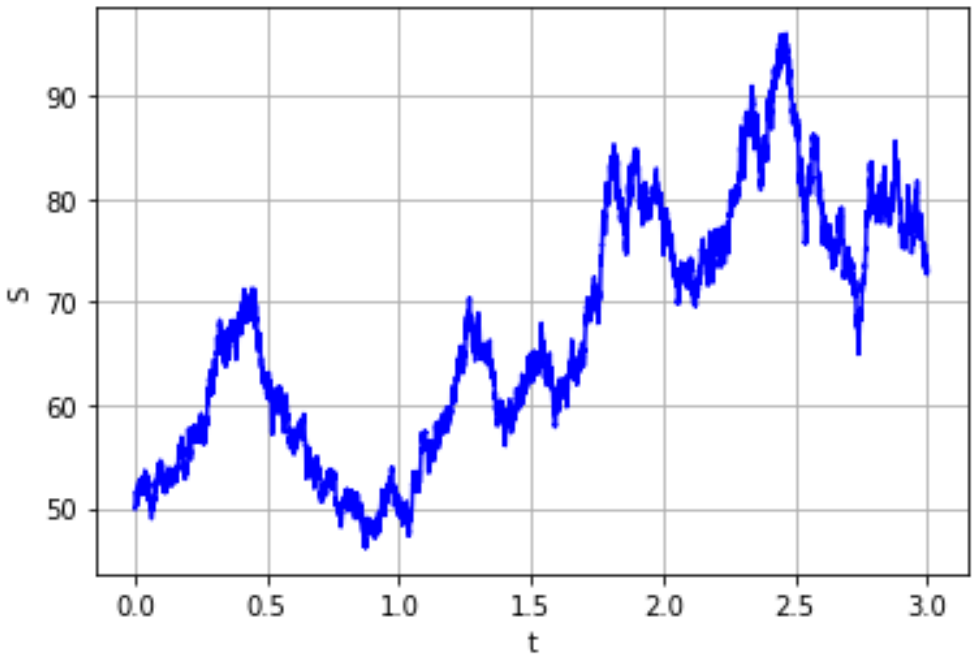

Рис. 1: Возможный путь в модели Блэка-Шоулза

1.1.2 Модель Хестона

В модели Хестона динамика $S(t)$ цены акции подчиняется следующей динамике:

$$\begin{cases} dS(t) = \mu S(t)dt + \sqrt{v(t)}S(t)dW_S^{\mathbb{P}}(t),\ S(t_0) > 0 \\ dv(t) = \kappa(\theta^{\mathbb{P}} - v(t))dt + \gamma\sqrt{v(t)}dW_v^{\mathbb{P}}(t),\ v(t_0) = v_0 > 0. \end{cases}$$

где $v(t)$ - мгновенная дисперсия цены акции;
$dW_S^{\mathbb{P}}(t)$ и $dW_v^{\mathbb{P}}(t)$ - винеровские процессы с корреляцией $\rho_{S,v}$;
μ - параметр дрифта;
$\theta^{\mathbb{P}} > 0$ - значение дисперсии, к которому стремится ожидаемое значение $v(t)$ при стремлении t к бесконечности;
$\kappa > 0$ - частота возвращения $v(t)$ к $\theta^{\mathbb{P}}$;
$\gamma > 0$ - волатильность волатильности $v(t)$.

Динамика для $v(t)$ является моделью CIR (Cox–Ingersoll–Ross). Для строгой положительности процесса $v(t)$ необходимо выполнение следующего условия (Феллера): $2\kappa\theta^{\mathbb{P}} \geq \gamma^2$. На практике, как правило, оно не выполнено, но для моделирования можно пользоваться тем фактом, что $v(t)$

подчиняется нецентральному распределению χ^2. Условное распределение $v(t)$ при условии $v(s)$, s<t, имеет следующий вид [15]:

$$v(t)|v(s) \sim \bar{c}(t,s) \cdot \chi^2(\delta, \bar{\kappa}(t,s)), \tag{3}$$

$$\bar{c}(t,s) = \frac{\gamma^2}{4\kappa}(1 - e^{-\kappa(t-s)}),\ \delta = \frac{4\kappa\bar{v}}{\gamma^2},\ \bar{\kappa}(t,s) = \frac{4\kappa e^{-\kappa(t-s)}}{\gamma^2(1 - e^{-\kappa(t-s)})}v(s)$$

Пример (рисунок(2)) двух траекторий процесса (3) с одинаковыми параметрами: $\kappa = 1$, $\gamma = 0.12$, $\theta^{\mathbb{P}} = 0.25$, $T = 4$, и разными начальными значениями $v_0 = 0.1$ (синяя) и $v_0 = 0.5$ (красная):

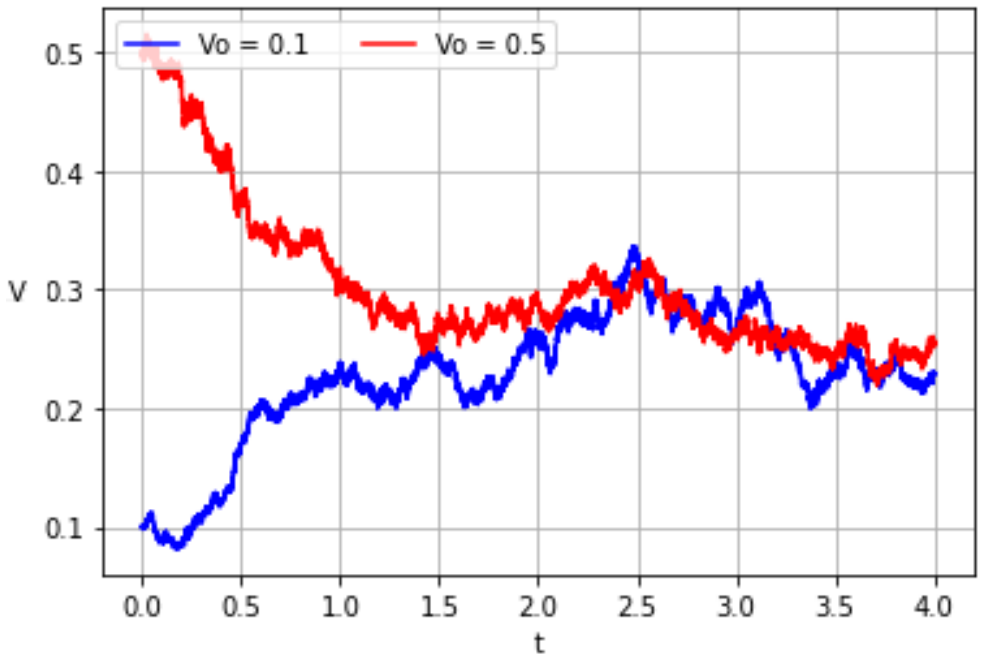

Рис. 2: Возможный путь модели CIR

Из рисунка видно, что со временем значения процесса стремятся к значению $\theta^{\mathbb{P}} = 0.25$. При больших параметрах γ амплитуды колебания процесса вокруг $\theta^{\mathbb{P}}$ становятся больше.

Чтобы дисконтированный процесс был мартингалом, необходимо от реальной меры $\mathbb{P}$ перейти к риск-нейтральной $\mathbb{Q}$. Пойдем путем, предложенным в [15], тогда получим следующую динамику под риск-нейтральной

мерой:

$$\begin{cases} dS(t) = rS(t)dt + \sqrt{v(t)}S(t)dW_S^{\mathbb{Q}}(t),\ S(t_0) > 0 \\ dv(t) = \kappa(\bar{\theta} - v(t))dt + \gamma\sqrt{v(t)}dW_v^{\mathbb{Q}}(t),\ v(t_0) = v_0 > 0, \end{cases} \tag{4}$$

где $\bar{\theta} := \theta^{\mathbb{P}} - \rho_{S,v}\gamma(\mu - r)/\kappa$. Чтобы такой переход имел место, необходимо выполнение $v(t) > 0$, однако известно, что процесс CIR может и принимать значение 0, но затем он впоследствии принимает положительные значения. Отметим, что одна из проблем использования данной модели - это проблема ее калибровки к реальным данным, так как в данном случае необходимо откалибровать 5 параметров (еще один необходимый параметр - это v_0, начальная дисперсия).

Отличие модели Хестона от модели Блэка-Шоулза состоит в том, что волатильность базового актива не является константой, а является случайным процессом, что наиболее приближено к реалиям рынка, так как волатильность цены базового актива в различные моменты времени является разной.

Так же как и в случае модели Блэка-Шоулза, перейдем к процессу X(t) = log(S(t)):

$$\begin{cases} dX(t) = (r - \frac{1}{2}v(t))dt + \sqrt{v(t)}dW_X^{\mathbb{Q}}(t), X(t_0) = log(S(t_0)) \\ dv(t) = \kappa(\bar{\theta} - v(t))dt + \gamma\sqrt{v(t)}dW_v^{\mathbb{Q}}(t),\ v(t_0) = v_0 > 0. \end{cases} \tag{5}$$

При этом для удобства обозначения было принято $dW_X^{\mathbb{Q}}(t) = dW_S^{\mathbb{Q}}(t)$, тогда $\rho_{X,v} = \rho_{S,v}$.

1.1.3 Модель Бейтса

Прямым обобщением предыдущих моделей будет модель Бейтса, записанная в реальной мере $\mathbb{P}$:

$$\begin{cases} dS(t) = (\mu - \xi\mathbb{E}[e^J - 1])S(t)dt + \sqrt{v(t)}S(t)dW_S^{\mathbb{P}}(t) + (e^J - 1)S(t)dX_{\mathcal{P}}(t), \\ dv(t) = \kappa(\theta^{\mathbb{P}} - v(t))dt + \gamma\sqrt{v(t)}dW_v^{\mathbb{P}}(t). \end{cases}$$

В ней добавлено еще одно слагаемое, содержащее приращение пуассоновского процесса $X_{\mathcal{P}}(t)$ с интенсивностью $\xi > 0$ и нормально распределенным размером прыжков J: $J \sim \mathcal{N}(\mu_J, \sigma_J^2)$, соответственно получаем еще 3 дополнительных параметра, остальные параметры имеют такой же смысл, как и в модели Хестона. Процесс $X_{\mathcal{P}}(t)$ независим от размеров прыжков и от броуновских движений $W_S^{\mathbb{P}}(t)$ и $W_v^{\mathbb{P}}(t)$. Таким образом, в данной модели учитываются резкие скачки в цене акций (гэпы), которые могут возникать, например, из-за новости или отчета. Перейдем к риск-нейтральной мере $\mathbb{Q}$. В данном случае таких мер существует бесконечное множество, но мы должны выбрать произвольную риск-нейтральную меру $\mathbb{Q}$, в которой дисконтированный процесс S(t) будет мартингалом. Пользуясь аналогичным преобразованием, как и в модели Хестона, получаем следующую динамику:

$$\begin{cases} dS(t) = (r - \xi\mathbb{E}[e^J - 1])S(t)dt + \sqrt{v(t)}S(t)dW_S^{\mathbb{Q}}(t) + (e^J - 1)S(t)dX_{\mathcal{P}}(t), \\ dv(t) = \kappa(\bar{\theta} - v(t))dt + \gamma\sqrt{v(t)}dW_v^{\mathbb{Q}}(t). \end{cases} \tag{6}$$

Пример (рисунок (3)) траектории модели (6) с параметрами r=0.05, $\kappa = 0.5$, $\gamma = 0.2$, $\bar{\theta} = 0.35$, $v(0) = 0.2$, $\rho_{S,v} = -0.45$, $\xi = 1$, $\mu_J = -0.5$, $\sigma_J = 0.6$, $S_0 = 50$, T = 3:

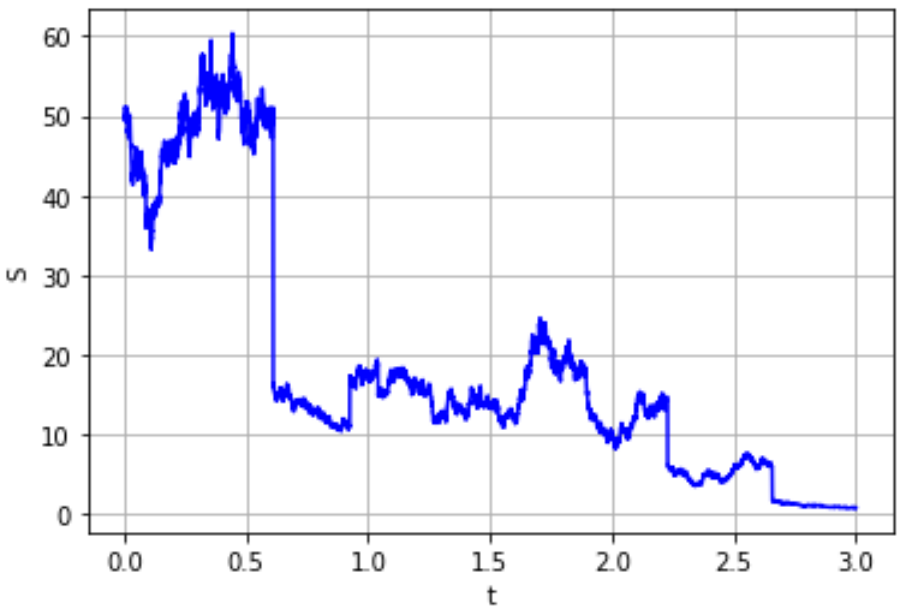

Рис. 3: Возможный путь модели Бейтса

Как и в предыдущих моделях, сделаем переход к процессу $X(t) = log(S(t))$:

$$\begin{cases} dX(t) = (r - \frac{1}{2}v(t) - \xi\mathbb{E}[e^J - 1])dt + \sqrt{v(t)}dW_S^{\mathbb{Q}}(t) + JdX_{\mathcal{P}}(t), \\ dv(t) = \kappa(\bar{\theta} - v(t))dt + \gamma\sqrt{v(t)}dW_v^{\mathbb{Q}}(t). \end{cases} \tag{7}$$

1.2 Оценивание опционов

Модели из предыдущих разделов можно использовать для оценивания деривативов, причем, если деривативы зависят от одних и тех же базовых активов, то можно использовать одни и те же модели базовых активов для их оценивания с теми же параметрами. Чтобы оценить дериватив, нужно прежде всего откалибровать параметры модели. Если калибровать модель к историческим данным базовых активов, то полученная модель будет под реальной мерой $\mathbb{P}$, а нужно, чтобы она была под риск-нейтральной, так как оценка деривативов происходит под риск-нейтральной мерой. Справедливая стоимость деривативa будет лежать между спросом и предложением на рынке. Чем ликвиднее актив, тем меньше будет спрэд между спросом и предложением. Если дериватив ликвиден и известны аналитические формулы, зависящие от параметров моделей, для его оценивания, то можно калибровать цены по формулам к ценам на рынке. Таким образом, можно откалибровать параметры моделей.

В качестве такого деривативa используют опционы. При оценивании опционов используется риск-нейтральная мера, при этом они ликвидны и тогда имеет смысл калибровать модель к их ценам. Таким образом, откалиброванная модель будет сразу под риск-нейтральной мерой. Далее в работе будут использоваться только европейские опционы на акции типа call.

Европейский опцион на акцию типа call [15] дает право держателю опциона купить акцию в определенный момент времени в будущем t = T (дата экспирации) по фиксированной цене K (страйк).

Если в момент времени T цена акции S(T) будет больше цены страйка, то держатель опциона после его реализации получит прибыль S(T) -

K. Если же цена акции окажется меньше цены страйка, то держатель ничего не получает. Таким образом, в момент времени T цена опциона будет равна:

$$V_c(T, S(T)) = H(T, S(T)) := max(S(T) - K, 0),$$

где $V_c(t, S(t))$ обозначает цену европейского опциона типа call на акцию в момент времени t при цене акции S(t), $H(T, \cdot)$ функция выплат по опциону. Далее будем использовать обозначение $V_c(t, S)$, где S=S(t), но нужно иметь в виду, что опцион зависит еще от экспирации T и страйка K.

Чтобы оценить европейский опцион, используем мартингальный подход, то есть, считаем, что дисконтированная цена опциона является мартингалом [15]. Также естественно использовать знание о том, что, как правило, известны выплаты в дату экспирации по опционам. Таким образом получаем, что цена европейского опциона с экспирацией T типа call в момент времени t равна:

$$V_c(t, S) = M(t)\mathbb{E}^{\mathbb{Q}}[\frac{V_c(T, S)}{M(T)}|\mathfrak{F}(t)] = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}[V_c(T, S)|\mathfrak{F}(t)]$$

где $\mathfrak{F}(t) = \sigma(S(s); s \leq t)$, M(t)-количество денег на сберегательном счете в момент времени t с процентной ставкой r, такое, что в начальный момент времени t_0 $M(t_0) = 1$.

Перейдем от процесса S(t) к процессу X(t) = log(S(t)). Обозначим $\tau = T - t_0$, X(t) = x, X(T) = y. Цена опциона в момент времени t_0:

$$V_c(t_0, x) = e^{-r\tau}\mathbb{E}^{\mathbb{Q}}[V_c(T, y)|\mathfrak{F}(t_0)] = e^{-r\tau}\int_{\mathbb{R}} V_c(T, y) f_X(T, y; t_0, x)dy \qquad (8)$$

где $f_X(T, y; t_0, x)$- плотность вероятности перехода. Основная проблема состоит в получении плотности вероятности перехода, так как не для всех процессов это возможно сделать в явном виде. Однако для определенного типа процессов, например, принадлежащих афинному типу, можно получить аналитический вид условной характеристической функции, и затем через нее вычислить функцию плотности перехода. Все процессы, которые рассматривались ранее под риск-нейтральной мерой и записанные в форме

(2), (5), (7) являются марковскими и принадлежат афинному типу. Прежде, чем записать явный вид характеристических функций для динамик (2), (5), (7), запишем формулы для подсчета цены опциона через разложение Фурье. Для удобства обозначим $f_X(y) \equiv f_{X(T)}(y) := f_X(T, y; t_0, x)$.

Плотность $f_X(y)$ и характеристическая функция связаны следующими равенствами:

$$\phi_X(u) = \int_{\mathbb{R}} e^{iyu} f_X(y) dy$$

$$f_X(y) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-iyu} \phi_X(u) du.$$

Разложение Фурье по косинусам на отрезке $[a, b] \subset \mathbb{R}$ функции g имеет следующий вид:

$$g(y) = \frac{\bar{A}_0}{2} + \sum_{k=1}^{\infty} \bar{A}_k \cdot cos(k\pi \frac{y-a}{b-a}), \tag{9}$$

где

$$\bar{A}_k = \frac{2}{b-a} \int_a^b g(y) cos(k\pi \frac{y-a}{b-a}) dy. \tag{10}$$

Так как функция плотности быстро стремится к 0 на бесконечности, то можно интегрирование по всей вещественной оси сузить до интегрирования по отрезку $[a, b] \subset \mathbb{R}$, тогда получаем следующую аппроксимацию характеристической функции:

$$\phi_X(u) = \int_{\mathbb{R}} e^{iyu} f_X(y) dy \approx \int_a^b e^{iyu} f_X(y) dy = \tilde{\phi}_X(u) \tag{11}$$

Далее выберем произвольную константу $w \in \mathbb{R}$, тогда, в силу (11), выполняется следующее:

$$\tilde{\phi}_X(u) e^{iw} = \int_a^b e^{i(yu+w)} f_X(y) dy$$

После взятия вещественной части получим:

$$Re\left\{\tilde{\phi}_X(u)e^{iw}\right\} = Re\left\{\int_a^b e^{i(yu+w)}f_X(y)dy\right\} = \int_a^b cos(uy+w)f_X(y)dy \quad (12)$$

Теперь выберем в качестве $u = \frac{k\pi}{b-a},\ k \in \mathbb{N} \cup 0$ в (12) и умножим на $e^{-i\frac{ka\pi}{b-a}}\frac{2}{b-a}$:

$$\bar{A}_k \equiv \frac{2}{b-a}Re\left\{\tilde{\phi}_X(\frac{k\pi}{b-a})e^{-i\frac{ka\pi}{b-a}}\right\} = \frac{2}{b-a}\int_a^b cos(k\pi\frac{y-a}{b-a})f_X(y)dy$$

Получили определение коэффициента $\bar{A}_k$ (10). В силу (11) получаем, $\bar{A}_k \approx \bar{F}_k$, где:

$$\bar{F}_k := \frac{2}{b-a}Re\left\{\phi_X(\frac{k\pi}{b-a})e^{-i\frac{ka\pi}{b-a}}\right\} \quad (13)$$

Далее заменяем $\bar{A}_k$ на $\bar{F}_k$ в разложении $f_X(y)$ на $[a,b]$:

$$\hat{f}_X(y) \approx \frac{\bar{F}_0}{2} + \sum_{k=1}^{\infty}\bar{F}_k\ cos(k\pi\frac{y-a}{b-a}).$$

Вернемся к формуле (8). Так как функция плотности $f_X(y)$ быстро стремится к 0 при $y \to \pm\infty$, то без сильной потери в точности сужаем интегрирование по всей вещественной оси до интегрирования по отрезку $[a,b] \subset \mathbb{R}$. Получаем следующую аппроксимацию цены опциона:

$$V(t_0,x) \approx V_I(t_0,x) = e^{-r\tau}\int_a^b V(T,y)f_X(y)dy.$$

Как отмечалось ранее, так как обычно функция плотности $f_X(y)$ неизвестна, но зато известна ее характеристическая функция, то функция плотности аппроксимируется с помощью разложения Фурье по косинусам по y, как в (9), (10):

$$\hat{f}_X(y) = \frac{\bar{A}_0(x)}{2} + \sum_{k=1}^{\infty}\bar{A}_k(x)cos(k\pi\frac{y-a}{b-a})$$

$$\bar{A}_k(x) := \frac{2}{b-a}\int_a^b \hat{f}_X(y)cos(k\pi\frac{y-a}{b-a})dy.$$

Стоит отметить, что тут уже явно учитывается зависимость от x (состояния процесса в момент t_0), в силу того, что работа происходит с переходными плотностями. Таким образом, получаем:

$$V_I(t_0,x) = e^{-r\tau}\int_a^b V(T,y)(\frac{\bar{A}_0(x)}{2} + \sum_{k=1}^{\infty}\bar{A}_k(x)cos(k\pi\frac{y-a}{b-a}))dy$$

Меняем местами суммирование и интегрирование (применение теоремы Фубини) и вводим следующие величины:

$$H_k := \frac{2}{b-a}\int_a^b V(T,y)cos(k\pi\frac{y-a}{b-a})dy,$$

H_k - это коэффициенты разложения Фурье по косинусам функции выплат $V(T,y)$. Тогда получаем:

$$V_I(t_0,x) = \frac{b-a}{2}e^{-r\tau}\cdot(\frac{\bar{A}_0(x)\cdot H_0}{2} + \sum_{k=1}^{\infty}\bar{A}_k(x)\cdot H_k)$$

Таким образом, получилось от интеграла произведения функций $f_X(y)$ и V(T,y) перейти к произведению коэффициентов их разложения в ряд Фурье по косинусам.

Так как коэффициенты убывают быстро при достаточно большом k, то мы оставляем только первые N элементов суммы и получаем следующую аппроксимацию:

$$V_{II}(t_0,x) = \frac{b-a}{2}e^{-r\tau}\cdot(\frac{\bar{A}_0(x)\cdot H_0}{2} + \sum_{k=1}^{N-1}\bar{A}_k(x)\cdot H_k) \qquad (14)$$

Теперь заменяем коэффициенты $\bar{A}_k(x)$ в (14) на $\bar{F}_k(x)$, которые опре-

делены в (13), и получаем следующую аппроксимацию, как в [15]:

$$V(t_0,x) \approx V_{III}(t_0,x) = e^{-r\tau}(\frac{H_0}{2} + \sum_{k=1}^{N-1} Re\left\{\phi_X(\frac{k\pi}{b-a})e^{-i\frac{ka\pi}{b-a}}\right\} \cdot H_k) \quad (15)$$

где $\tau = T - t_0$, x - функция начальной цены $S(t_0)$, например, $x = log(S(t_0))$ или $x = log(\frac{S(t_0)}{K})$, и $\phi_X(u) = \phi_X(u;t_0,T) \equiv \phi_X(u,x;t_0,T)$ - условная характеристическая функция при условии x. Формула (15) - это COS формула для достаточно общего множества случайных процессов. Коэффициенты H_k можно получить аналитическим путем для опционов европейского типа. Кроме того для процессов, введенных ранее, (15) можно вычислять одновременно для нескольких страйков с одной экспирацией.

В случае европейских опционов типа call функция выплат будет иметь следующий вид:

$$V(T,y) := max[K(e^y - 1), 0],\ y(T) = \frac{S(T)}{K}$$

Тогда коэффициенты H_k^c будут равны:

$$H_k^c = \frac{2}{b-a}\int_a^b max[K(e^y-1),0]cos(k\pi\frac{y-a}{b-a})dy$$

Тогда после вычисления интеграла получаем:

$$H_k^{call} = \begin{cases} \frac{2}{b-a}K(\chi_k(0,b) - \psi_k(0,b)), & \text{если } a < 0 < b, \\ \frac{2}{b-a}K(\chi_k(a,b) - \psi_k(a,b)), & \text{если } 0 < a < b, \\ 0, & \text{если } a < b < 0, \end{cases}$$

где для $a \le c \le d \le b$:

$$\chi_k(c,d) := \int_c^d e^y cos(k\pi\frac{y-a}{b-a})dy = \frac{1}{1+(\frac{k\pi}{b-a})^2}[cos(k\pi\frac{d-a}{b-a})e^d -$$

$$-cos(k\pi\frac{c-a}{b-a})e^c + \frac{k\pi}{b-a}sin(k\pi\frac{d-a}{b-a})e^d - \frac{k\pi}{b-a}sin(k\pi\frac{c-a}{b-a})e^c],$$

$$\psi_k(c,d) := \int_c^d cos(k\pi\frac{y-a}{b-a})dy,$$

$$\psi_k(c,d) = \begin{cases} [sin(k\pi\frac{d-a}{b-a}) - sin(k\pi\frac{c-a}{b-a})]\frac{b-a}{k\pi}, & \text{если } k \neq 0, \\ (d-c), & \text{если } k = 0. \end{cases}$$

В качестве отрезка отрезка интегрирования можно выбрать следующий отрезок [15]:

$$[a,b] = [-L\sqrt{T}, L\sqrt{T}] \tag{16}$$

где L = [8, 12]. Такой отрезок очень удобен при вычислении цен опционов для нескольких страйков одновременно.

Осталось определить условные характеристические функции для процессов (2), (5), (7).

Пусть $\tau = T - t$, тогда для процесса (2) дисконтированная (домноженная на $e^{-r\tau}$) условная характеристическая функция будет иметь вид [15]:

$$\phi_X^{BS}(u;t,T) = e^{\bar{A}(u,\tau)+\bar{B}(u,\tau)X(t)} = e^{iuX(t)+iu(r-\frac{1}{2}\sigma^2)\tau-\frac{1}{2}u^2\sigma^2\tau-r\tau}.$$

Для процесса (5) [15]:

$$\phi^H_{X(t),v(t)}(u,0;t,T) = e^{\bar{A}_H(u,0,\tau)+\bar{B}(u,0,\tau)X(t)+\bar{C}_H(u,0,\tau)v(t)}$$

$$\bar{B}(u,0,\tau) = iu$$

$$\bar{C}_H(u,0,\tau) = \frac{1-e^{-D\tau}}{\gamma^2(1-ge^{-D\tau})}(\kappa - \gamma\rho_{X,v}iu - D)$$

$$\bar{A}_H(u,0,\tau) = r(iu-1)\tau + \frac{\kappa\bar{\theta}\tau}{\gamma^2}(\kappa - \gamma\rho_{X,v}iu - D) - \frac{2\kappa\bar{\theta}}{\gamma^2}log(\frac{1-ge^{-D\tau}}{1-g})$$

$$D = \sqrt{(\kappa - \gamma\rho_{X,v}iu)^2 + (u^2+iu)\gamma^2}$$

$$g = \frac{\kappa - \gamma\rho_{X,v}iu - D}{\kappa - \gamma\rho_{X,v}iu + D}$$

Для процесса (7) [15]:

$$\phi^B_{X(t),v(t)}(u,0;t,T) = e^{\bar{A}_B(u,0,\tau)+\bar{B}(u,0,\tau)X(t)+\bar{C}_B(u,0,\tau)v(t)}$$

$$\bar{C}_B(u,0,\tau) = \bar{C}_H(u,0,\tau)$$

$$\bar{A}_B(u,0,\tau) = \bar{A}_H(u,0,\tau) - \xi i u \tau (e^{\mu_J + \frac{1}{2}\sigma_J^2} - 1) + \xi\tau(e^{iu\mu_J - \frac{1}{2}\sigma_J^2 u^2} - 1).$$

Таким образом, получили формулы для оценивания call опционов по известным параметрам. Но нам необходимо получить эти параметры. Их можно получить, имея рыночные данные цен опционных контрактов на акции типа call с различными страйками и экспирациями, решив следующую задачу минимизации:

$$\min_{\Psi} \sqrt{\sum_i \sum_j \omega_{i,j} (V_c^{mrkt}(K_i, T_j) - V_c(t_0, S_0; K_i, T_j, \Psi))^2}, \tag{17}$$

где $V_c^{mrkt}(K_i, T_j)$ - рыночная цена опционного контракта типа call со страйком K_i и экспирацией T_j, $V_c(t_0, S_0; K_i, T_j, \Psi)$ - модельная цена опциона, рассчитанная по формуле (15), Ψ - пространство параметров, $\omega_{i,j}$ - веса, которые часто принимают равными 1.

Таким образом, параметры должны быть такими, чтобы минимизировать отклонение между теоретическими ценами и реальными (рыночными). Для этого можно применить, например, генетические алгоритмы, так как они не требуют гладкости минимизируемой функции, но они более требовательные к вычислительной мощности.

1.3 Многомерная динамика

Многие деривативы могут зависеть от нескольких акций, поэтому для описания совместной динамики цен этих акций используются многомерные процессы. При этом нужно учитывать корреляции: между ценами акций, между ценами акций и волатильностями акций, между волатильностями акций. При этом эти корреляции нельзя откалибровать по данным на опционы на одну акцию. Тогда необходимо расширение моделей (1), (4), (6) до многомерного случая. Это можно сделать разными путями, но в данной работе будет использоваться метод расширения одномерной модели Хестона, предложенный в [13], так как тогда модель будет обладать двумя свойствами:

1) Каждая одномерная подмодель Хестона будет формировать одномерную модель Хестона (4), это позволит откалибровать параметры одномерной подмодели и использовать их в многомерной модели;

2) Оставшиеся параметры многомерной модели состоят из корреляции между акциями. В этом случае останется откалибровать только $\frac{n(n-1)}{2}$ корреляций для модели Хестона для n активов.

Таким образом, для $i = 1, ..., n$, получаем следующую систему:

$$\begin{pmatrix} dX_i(t) \\ dv_i(t) \end{pmatrix} = \begin{pmatrix} r - \frac{1}{2}v_i(t) \\ \kappa_i(\bar{\theta}_i - v_i(t)) \end{pmatrix} dt \; + \; \begin{pmatrix} \sqrt{v_i(t)} & 0 \\ 0 & \gamma_i\sqrt{v_i(t)} \end{pmatrix} \begin{pmatrix} dW_i(t) \\ dW_{v_i}(t) \end{pmatrix} \quad (18)$$

где $W_i(t)$ - винеровский процесс, остальные обозначения как в (5). Кроме того, будет удобно следующее представление: применяем разложение Холецкого [22] для процессов $W_i(t)$ и $W_{v_i}(t)$, корреляция которых равна ρ_i, и получаем:

$$\begin{pmatrix} dX_i(t) \\ dv_i(t) \end{pmatrix} = \begin{pmatrix} r - \frac{1}{2}v_i(t) \\ \kappa_i(\bar{\theta}_i - v_i(t)) \end{pmatrix} dt +$$

$$+ \; \begin{pmatrix} \sqrt{v_i(t)} & 0 \\ 0 & \gamma_i\sqrt{v_i(t)} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \rho_i & \sqrt{1-\rho_i^2} \end{pmatrix} \begin{pmatrix} dW_i(t) \\ d\overline{W}_{v_i}(t) \end{pmatrix}$$

где параметры $(\kappa_i, \gamma_i, \bar{\theta}_i, v_{0i}, \rho_i)$ известны после калибровки одномерных моделей. Процессы $W_i(t)$ и $\overline{W}_{v_i}(t)$ независимы.

Однако в данной системе в общем случае присутствуют следующие корреляции:

1) ρ_{ij} - корреляции между броуновским движением акций i и j;

2) $\rho_{v_iv_j}$ - корреляции между броуновскими движениями волатильностей акций i и j;

3) $\rho_{X_iv_j}$ - корреляции между броуновским движением акции i и броуновским движением волатильности акции j, $i \neq j$.

Корреляции между акциями ρ_{ij} не могут быть устранены, так как все рынки взаимосвязаны. Если какой-то сектор растет или падает, то и входящие в него акции в среднем будут расти или падать в зависимости от своей силы, соответственно между этими акциями будет явно наблю-

даться положительная корреляция. Однако, акции из противоположного сектора могут показывать обратную динамику, соответственно между акциями из этого сектора и предыдущего будет наблюдаться отрицательная корреляция.

Корреляции между волатильностями $\rho_{v_i v_j}$ могут также делать свой вклад, так как, например, если набор акций становится в среднем более волатильным, то некоторые трейдеры могут переходить на торговлю другими акциями, что в итоге в среднем увеличит и их волатильность.

Для моделирования путей случайных процессов далее будет использоваться метод Монте-Карло MQE - многомерный квадратично- экспоненциальный метод (Multi-dimensional Quadratic Exponential (MQE) method) [26]. Метод MQE по сравнению с обычным методом Эйлера с отсечением отрицательных значений волатильности (Euler Full Truncation method) более устойчив, когда условия Феллера не выполняются, лучше работает для больших экспираций, и его время работы растет линейно с увеличением размерности модели. Имплеметация метода MQE для случая $\rho_{v_i v_j} \neq 0$ представляет собой нетривиальную задачу [26], так как распределения процессов дисперсий не являются нормальными и их аппроксимация методом QE - квадратично-экспоненциальный метод (Quadratic Exponential), является нелинейной, поэтому применение разложения Холецкого будет неточным. Метод MQE [26] можно расширить и на случай $\rho_{v_i v_j} \neq 0$ с помощью применения копул через метод NorTA [14], но это не входит в задачи данной работы. Таким образом, далее будем считать, что $\rho_{v_i v_j} = 0$. Кроме того, также будем считать, что $\rho_{X_i v_j} = 0,\ i \neq j$.

Перед тем как записать корреляционную структуру между процес-

сами, запишем систему (18) в следующем виде:

$$\begin{cases} dv_1(t) = \kappa_1(\bar{\theta}_1 - v_1(t))dt + \gamma_1\sqrt{v_1(t)}dW_{v_1}(t) \\ \dots \\ dv_n(t) = \kappa_n(\bar{\theta}_n - v_n(t))dt + \gamma_n\sqrt{v_n(t)}dW_{v_n}(t) \\ \\ dX_1(t) = (r - \frac{1}{2}v_1(t))dt + \sqrt{v_1(t)}W_1(t) \\ \dots \\ dX_n(t) = (r - \frac{1}{2}v_n(t))dt + \sqrt{v_n(t)}W_n(t) \end{cases} \tag{19}$$

Матрица корреляций $C \in \mathbb{R}^{2n\times 2n}$ 2n - мерного броуновского движения

$$\begin{pmatrix} dW_{v_1}(t) \\ \dots \\ dW_{v_n}(t) \\ dW_1(t) \\ \dots \\ dW_n(t) \end{pmatrix}$$

имеет следующий вид при сделанных предположениях:

$$C = \begin{pmatrix} E_n & C_{Xv} \\ C_{Xv}^T & C_X \end{pmatrix} \tag{20}$$

где E_n - единичная матрица размерности $n \times n$, C_X - корреляционная матрица между броуновскими движениями акций размерности $n \times n$; C_{Xv}^T матрица размерности $n \times n$ следующего вида:

$$\begin{pmatrix} \rho_1 & 0 & \dots & & 0 \\ 0 & \rho_2 & 0 & \dots & 0 \\ & & \dots & & \\ 0 & & \dots & 0 & \rho_n \end{pmatrix} \tag{21}$$

Корреляции ρ_i становятся известны после калибровок одномерных моделей Хестона.

Получить корреляции ρ_{ij} между акциями можно:

1) сделав калибровку к опционам, зависящим от группы этих акций;

Тут есть несколько подводных камней: они обладают низкой ликвидностью, может быть трудно получить доступ к информации о ценах на них и, кроме того, достаточно проблемно получить хорошую аппроксимацию по распределению выплат по ним.

2) воспользовавшись методом на основе исторических данных, предложенным в [13].

1.3.1 Метод калибровки корреляции

Необходимо найти корреляционную матрицу $\overline{C_X}$ между броуновскими движениями акций-такую, что корреляция между доходностями модельных логарифмических цен на акции была близка к историческим корреляциям этих доходностей, то есть:

$$\min_{C_X \in C_n} \left\| \mathbb{E}C^{\mathbb{P}}(C_X) - C_h^{\mathbb{P}} \right\|. \tag{22}$$

где C_n - множество всех корреляционных матриц размерности n, остальные матрицы определяются следующим образом:

$$C^{\mathbb{P}} = (\rho_{ij}^{\mathbb{P}})_{1 \le i,j \le n},$$

$$C_h^{\mathbb{P}} = (\bar{\rho}_{ij}^{\mathbb{P}})_{1 \le i,j \le n},$$

$$C_X = (\rho_{ij})_{1 \le i,j \le n}. \tag{23}$$

$$\bar{\rho}_{ij}^{\mathbb{P}} = \frac{cov(X_{hi}(t+\Delta) - X_{hi}(t), X_{hj}(t+\Delta) - X_{hj}(t))}{\sqrt{\mathbb{D}(X_{hi}(t+\Delta) - X_{hi}(t))}\sqrt{\mathbb{D}(X_{hj}(t+\Delta) - X_{hj}(t))}},$$

где $\bar{\rho}_{ij}^{\mathbb{P}}$ - исторические корреляции приращений доходностей логарифмированных цен на акции i и j, $X_{hi}(t)$ - наблюдаемое значение логарифмирован-

ной цены акции i в момент времени t, $\Delta > 0$.

$$\rho_{ij}^{\mathbb{P}} = \frac{cov(X_i(t+\Delta) - X_i(t), X_j(t+\Delta) - X_j(t))}{\sqrt{\mathbb{D}(X_i(t+\Delta) - X_i(t))}\sqrt{\mathbb{D}(X_j(t+\Delta) - X_j(t))}},$$

где $\rho_{ij}^{\mathbb{P}}$ корреляции приращения доходностей модельных логарифмированных цен на акции i и j в реальной мере $\mathbb{P}$, $X_j(t)$ - модельное значение логарифмированной цены акции j в момент времени t.

Метод [13] заключается в следующем. Предполагается такая же корреляционная структура, как в (20). Вначале происходит калибровка одномерных моделей (18). Далее происходит калибровка каждой корреляций между акциями по элементам матрицы C_X, то есть, решаются следующие задачи минимизации:

$$\min_{-1 \le \rho_{ij} \le 1} \left| \mathbb{E}\rho_{ij}^{\mathbb{P}}(\rho_{ij}) - \bar{\rho}_{ij}^{\mathbb{P}} \right|. \tag{24}$$

Чтобы решить задачу (24), в случае модели Хестона, можно воспользоваться фактом [13] , что $\mathbb{E}\rho_{ij}^{\mathbb{P}}(\rho_{ij})$ является строго возрастающей непрерывной функцией по ρ_{ij}, тогда применим метод бисекций. Однако на каждом шаге метода будут требоваться значения $\mathbb{E}\rho_{ij}^{\mathbb{P}}(\rho_{ij})$ и $\bar{\rho}_{ij}^{\mathbb{P}}$. Значение $\bar{\rho}_{ij}^{\mathbb{P}}$ можно получить, воспользовавшись оценкой:

$$\widehat{\rho}_{ij}^{\mathbb{P}} = \frac{\sum_{k=1}^{N}(X_{hi}^k - \bar{X}_{hi})(X_{hj}^k - \bar{X}_{hj})}{\sqrt{\sum_{k=1}^{N}(X_{hi}^k - \bar{X}_{hi})^2 \sum_{k=1}^{N}(X_{hj}^k - \bar{X}_{hj})^2}}, \tag{25}$$

$$X_{hi}^k = log\frac{S_{hi}^k}{S_{hi}^{k-1}},\ \bar{X}_{hi} = \frac{\sum_{k=1}^{N} X_{hi}^k}{N},$$

где S_{hi}^k - историческая цена акции i в k-ый день измерения; N - количество дней измерения.

Значение $\mathbb{E}\rho_{ij}^{\mathbb{P}}(\rho_{ij})$ можно определить с помощью вычисления матожидания методом Монте-Карло следующим способом. Воспользуемся ана-

логичной оценкой для $\rho_{ij}^{\mathbb{P}}(\rho_{ij})$, как для $\bar{\rho}_{ij}^{\mathbb{P}}$:

$$\rho_{ij}^{\mathbb{P}}(\rho_{ij}) = \frac{\sum_{k=1}^{N}(X_i^k - \bar{X}_i)(X_j^k - \bar{X}_j)}{\sqrt{\sum_{k=1}^{N}(X_i^k - \bar{X}_i)^2 \sum_{k=1}^{N}(X_j^k - \bar{X}_j)^2}}, \tag{26}$$

$$X_i^k = log\frac{S_i^k}{S_i^{k-1}},\ \bar{X}_i = \frac{\sum_{k=1}^{N} X_i^k}{N}.$$

где S_i^k - модельная цена акции i на k-м шаге; N - количество шагов.

Далее рассмотрим динамики $S_i(t)$ и $S_j(t)$ поподробнее. Имеем следующую динамику после разложения Холецкого, как в [13]:

$$\begin{pmatrix} dS_i(t) \\ dv_i(t) \\ dS_j(t) \\ dv_j(t) \end{pmatrix} = \begin{pmatrix} \mu_i S_i(t) \\ \kappa_i(\theta_i^{\mathbb{P}} - v_i(t)) \\ \mu_j S_j(t) \\ \kappa_j(\theta_j^{\mathbb{P}} - v_j(t)) \end{pmatrix} dt +$$

$$+ \begin{pmatrix} S_i(t)\sqrt{v_i(t)} & 0 & 0 & 0 \\ 0 & \gamma_i\sqrt{v_i(t)} & 0 & 0 \\ 0 & 0 & S_j(t)\sqrt{v_j(t)} & 0 \\ 0 & 0 & 0 & \gamma_i\sqrt{v_i(t)} \end{pmatrix} \times$$

$$\times \begin{pmatrix} 1 & 0 & 0 & 0 \\ \rho_i & \sqrt{1-\rho_i^2} & 0 & 0 \\ \rho_{ij} & 0 & \sqrt{1-\rho_{ij}^2} & 0 \\ \rho_j\rho_{ij} & 0 & \rho_j\sqrt{1-\rho_{ij}^2} & \sqrt{1-\rho_j^2} \end{pmatrix} \begin{pmatrix} d\bar{W}_i^{\mathbb{P}}(t) \\ d\bar{W}_{v_i}^{\mathbb{P}}(t) \\ d\bar{W}_j^{\mathbb{P}}(t) \\ d\bar{W}_{v_j}^{\mathbb{P}}(t) \end{pmatrix} \tag{27}$$

где $(\bar{W}_i^{\mathbb{P}}(t), \bar{W}_{v_i}^{\mathbb{P}}(t), \bar{W}_j^{\mathbb{P}}(t), \bar{W}_{v_j}^{\mathbb{P}}(t))$ - винеровский процесс с независимыми компонентами. Далее система (27) моделируется по методу Монте-Карло методом Эйлера с шагом $\frac{1}{250}$, при этом процессы v_i и v_j тоже моделируются методом Эйлера, но с отсечением отрицательных значений. Такое представление более удобно в виду того, что нет ограничений на экстремальные значений корреляций -1 и 1.

Необходимо промоделировать систему (27) $\bar{N}$ раз, получив $\bar{N}$ значений (26). Затем эти значения суммируются и делятся на $\bar{N}$. Таким образом

получается оценка для $\mathbb{E}\rho_{ij}^{\mathbb{P}}(\rho_{ij})$. В качестве $\bar{N}$ можно брать не очень большое значение в виду того, что при каждом моделировании пути получается уже усердненная корреляция вдоль пути.

Отметим, что откалиброванные одномерные модели Хестона необходимо перевести в реальную меру $\mathbb{P}$, то есть, нужно получить параметры μ_i и $\theta_i^{\mathbb{P}}$, так как они связаны в соотношении $\bar{\theta}_i := \theta_i^{\mathbb{P}} - \rho_i\gamma(\mu_i - r)/\kappa_i$, то нужно получить только один из них. Для этого воспользуемся оценкой, как в [13], для $\theta_i^{\mathbb{P}}$:

$$\theta_i^{\mathbb{P}} = \frac{1}{T}\sum_{k=1}^{N}(X_{hi}^k)^2 = \frac{1}{T}\sum_{k=1}^{N}(logS_{hi}^k - logS_{hi}^{k-1})^2,$$

где T - размерность временного промежутка в годах, на котором взяты исторические значения.

Таким образом, откалибровав поэлементно элементы матрицы C_X, получим матрицу $\widetilde{C}$, которая будет симметричной, но не обязательно положительно определенной. И можно применить метод регуляризации, как в ([19]), тогда она станет положительно определенной. Таким образом, получили только кандидата на решение задачи (22). После всей процедуры калибровки получим следующую корреляционную структуру:

$$C = \begin{pmatrix} E_n & C_{Xv} \\ C_{Xv}^T & \widetilde{C} \end{pmatrix}$$

У этого метода есть некоторые недостатки:

1) результирующая корреляционная матрица может существенно отличаться от той, на которой достигается минимум, так как поиск минимума проводится поэлементно. При этом результирующая матрица может не оказаться положительно определенной, тогда нужно будет использовать метод, предложенный в ([19]).

2) метод основан на калибровке к историческим данным, поэтому некоторые параметры могут быть ненаблюдаемы и эмпирические оценки могут не согласовываться с оценками в риск-нейтральной мере (при отсутствии арбитража) [9].

Так как в нашем случае нет возможности провести оценку в риск-

нейтральной мере, то в дальнейшем будет применяться метод на основе калибровки по историческим данным.

В случае модели Бейтса тоже можно попробовать применить изложенный метод, проделав вначале калибровку одномерных моделей Бейтса. Однако могут возникнуть трудности с решением задачи (24). Из-за наличия прыжков свойства функции $\mathbb{E}\rho_{ij}^{\mathbb{P}}(\rho_{ij})$ неизвестны, поэтому для поиска минимума нужно брать алгоритм глобальной оптимизации, например, генетический алгоритм, но это скажется на скорости вычислений, так как при подсчете значений функции $\mathbb{E}\rho_{ij}^{\mathbb{P}}(\rho_{ij})$ нужно моделировать $\bar{N}$ путей методом Монте Карло. Причем в силу наличия прыжков $\bar{N}$ придется брать большим, что увеличит объем вычислений. Есть еще трудности с переводом модели Бейтса в реальную меру $\mathbb{P}$, так как оценка в случае модели Хестона не работает из-за наличия Пуассоновского процесса.

В случае модели Блэка-Шоулза проблем никаких не возникает, так как в этом случае $\rho_{ij} = \rho_{ij}^{\mathbb{P}}$ ([26]).

1.3.2 Моделирование пуассоновских процессов в многомерной модели Бейтса

Многомерная модель Бейтса [15] имеет следующий вид:

$$\begin{cases} dv_1(t) = \kappa_1(\bar{\theta}_1 - v_1(t))dt + \gamma_1\sqrt{v_1(t)}dW_{v_1}(t) \\ \dots \\ dv_n(t) = \kappa_n(\bar{\theta}_n - v_n(t))dt + \gamma_n\sqrt{v_n(t)}dW_{v_n}(t) \\ dX_1(t) = (r - \frac{1}{2}v_1(t) - \xi_1\mathbb{E}[e^{J_1} - 1])dt + \sqrt{v_1(t)}W_1(t) + J_1 dX_{\mathcal{P}1}(t) \\ \dots \\ dX_n(t) = (r - \frac{1}{2}v_n(t) - \xi_n\mathbb{E}[e^{J_n} - 1])dt + \sqrt{v_n(t)}W_n(t) + J_n dX_{\mathcal{P}n}(t) \end{cases}$$

где $X_{\mathcal{P}1}(t), ..., X_{\mathcal{P}n}(t)$ - пуассоновские процессы с интенсивностями $\xi_1, ..., \xi_n$ соответственно, $J_1 \sim \mathcal{N}(\mu_{J_1}, \sigma_{J_1}^2), ..., J_n \sim \mathcal{N}(\mu_{J_n}, \sigma_{J_n}^2)$ - нормально распределённые амплитуды прыжков. В силу того, что прыжки цен на акции не являются независимыми, а могут быть подвержены влиянию рынка или сектора, то предлагается следующий подход к моделированию совместных

прыжков под влиянием сектора.

Пусть есть значения интенсивностей $\xi_1, ..., \xi_n$, полученные после калибровок одномерных моделей Бейтса. Если дериватив зависит от акций из одного сектора, то можно взять достаточно большой промежуток времени, посчитать, сколько прыжков $\bar{n}_i$ совершает акция и сколько прыжков n_i совершает акция из этого сектора одновременно с этим сектором. Тогда получаем, что в качестве вклада сектора в интенсивность прыжков ξ_i можно взять число $\xi_i \frac{n_i}{\bar{n}_i}$, и так для каждой акции в секторе. Далее понадобится факт, что можно представить пуассоновский процесс в виде суммы независимых пуассоновских процессов, просуммировав интенсивность которых получим интенсивность исходного пуассоновского процесса [20]. Так как акции должны "прыгать" в одно время, то можно сделать следующую итеративную процедуру разложения исходных пуассоновских процессов. Сделаем разложение интенсивностей, так что каждой интенсивности будет соответствовать независимый от остальных пуассоновский поток. Предлагается следующий итеративный процесс разложения интенсивностей:

1) Упорядочиваем по возрастанию элементы набора $(\frac{n_1}{\bar{n}_1}\xi_1, ..., \frac{n_n}{\bar{n}_n}\xi_n)$, и делаем переобозначение номеров и акций, так чтобы элементу с первым номером соответствовала акция, у которой соотношение $\frac{n_i}{\bar{n}_i}\xi_i$ было минимальным и т.д..

2) Далее делаем следующие разложения:

$$\xi_1 = \xi_1^* + \bar{\xi}_1$$

где $\xi_1^* = \frac{n_1}{\bar{n}_1}\xi_1$, а $\bar{\xi}_1$ - определяется уже, согласно определенным значениям.

$$\xi_2 = \xi_1^* + \xi_2^* + \bar{\xi}_2$$

где $\xi_2^* = \frac{n_2}{\bar{n}_2}\xi_2 - \xi_1^*$

$$\xi_3 = \xi_1^* + \xi_2^* + \xi_3^* + \bar{\xi}_3$$

где $\xi_3^* = \frac{n_3}{\bar{n}_3}\xi_3 - \xi_2^* - \xi_1^*$

и т.д.

$$\xi_n = \sum_{k=1}^{n-1} \xi_k^* + \bar{\xi}_n$$

Интенсивности $\xi_1^*, ..., \xi_{n-1}^*$ соответствуют независимым пуассоновским процессам $X_{\mathcal{P}1}^*(t), ..., X_{\mathcal{P}n-1}^*(t)$ с этими интенсивностями соответственно. Они описывают воздействие сектора на возникновение прыжков. Интенсивности $\bar{\xi}_1, ..., \bar{\xi}_n$ соответствуют независимым пуассоновским процессам $\bar{X}_{\mathcal{P}1}(t), ..., \bar{X}_{\mathcal{P}n}(t)$ с этими интенсивностями соответственно. Они уже описывают возникновение индивидуальных прыжков акций.

Чтобы вычислить значения $n_1, ..., n_n, \bar{n}_1, ..., \bar{n}_n$, нужно выбрать биржевой фонд (ETF - exchange-traded fund) на сектор, к которому относятся акции, затем по историческим данным на этот ETF и на акции посчитать возникновение прыжков.

У метода есть недостатки:

1) Возможен не полный учет вклада сектора в интенсивность прыжков акции, у которой будет самое высокое значение $\frac{n_n}{\bar{n}_n}\xi_n$;

2) Оценки вычисления вклада сектора в интенсивность $\frac{n_i}{\bar{n}_i}$ довольно грубые;

3) Вычисления зависят от выбора ETF на сектор.

Кроме того, при возникновении прыжка сектора большинство акций в этом секторе прыгнет в ту же сторону, но с разной силой. Поэтому амплитуды этих прыжков $J_1, ..., J_n$ будут иметь положительные корреляции, которые можно попробовать оценить через соотношения $\frac{n_1}{\bar{n}_1}, ..., \frac{n_n}{\bar{n}_n}$, выбрав, например, $\rho_{ij}^J = \min(\frac{n_i}{\bar{n}_i}, \frac{n_j}{\bar{n}_j})$. Так как, если сектор не сильно влияет на акцию i из этого сектора, то можно ожидать, что значение $\frac{n_i}{\bar{n}_i}$ будет сильнее отличаться от значения $\frac{n_j}{\bar{n}_j}$ для акции j, на которую сектор влияет сильнее, тем самым, их прыжки будут менее коррелированными.

Нужно отметить, что это довольно грубые оценки. Они получены по историческим данным, соответственно, в реальной мере. В данной работе действует дополнительное допущение о том, что характеристики амплитуд прыжков и их интенсивностей совпадают в риск-нейтральной и реальной мере. Это допущение было реализовано при переходе от процессов в риск-

нейтральной мере к процессам в реальной мере.

1.4 Метод MQE

Рассмотрим метод MQE, предложенный в работе [26], для моделирования путей многомерного процесса методом Монте-Карло. Но для начала кратко приведем метод QE [8] для моделирования (3). Этот метод необходим, так как многие компьютерные пакеты могут не включать достаточно высокоэффективный алгоритм моделирования нецентрального распределения χ^2, особенно когда условия Феллера не выполнены[15].

Метод основан на аппроксимации по двум первым моментам распределния (3) другими распределениями. Сперва вычисляются первый и второй моменты:

$$\begin{cases} \bar{m} = \mathbb{E}[v(t)|v(0)] = \bar{c}(t,0)(\delta + \bar{\kappa}(t,0)), \\ \bar{s}^2 = \mathbb{D}[v(t)|v(0)] = \bar{c}^2(t,0)(2\delta + 4\bar{\kappa}(t,0)). \end{cases}$$

Далее в зависимости от того, где лежит значение $\frac{\bar{s}^2}{\bar{m}^2}$ проводится аппроксимация либо нецентральным распределением χ^2 с одной степенью свободы, либо распределением с некоторой массой в нуле и экспоненциально распределенным концом, то есть, имеем следующие две аппроксимации:

$$v(t) \approx a(b + Z_v)^2 \tag{28}$$

где $Z_v \sim \mathcal{N}(0,1)$, $a = \frac{\bar{m}^2}{1+b^2}$ и $b^2 = 2\frac{\bar{m}^2}{\bar{s}^2} - 1 + \sqrt{2\frac{\bar{m}^2}{\bar{s}^2}}\sqrt{2\frac{\bar{m}^2}{\bar{s}^2} - 1} \geq 0$.

$$v(t) \approx \frac{1}{d} log(\frac{1-c}{1-u})\mathbb{I}_{u>c} \tag{29}$$

где $u \sim U[0,1]$, $c = \frac{\frac{\bar{s}^2}{\bar{m}^2}-1}{\frac{\bar{s}^2}{\bar{m}^2}+1}$ и $d = \frac{2}{\bar{m}\frac{\bar{s}^2}{\bar{m}^2}+1}$.

Первая аппроксимация хорошо определена при $\frac{\bar{s}^2}{\bar{m}^2} \leq 2$, а вторая при $\frac{\bar{s}^2}{\bar{m}^2} \geq 1$. Так как эти два интервала накладываются друг на друга, то тогда нужно выбрать некоторую разграничительную точку $y \in [1,2]$, и если $\frac{\bar{s}^2}{\bar{m}^2} \leq y$, то использовать аппроксимацию (28), а если $\frac{\bar{s}^2}{\bar{m}^2} > y$, то использовать

(29). Выбор y не сильно отражается на качестве аппроксимации метода [8], поэтому можно взять y=1.5, как середину указанного отрезка.

Теперь опишем метод MQE. Пусть у нас имеется система (19), с корреляционной структурой (20), (21). Делаем разложение Холецкого L корреляционной матрицы (20), тогда получим [26]:

$$L = \begin{pmatrix} E_n & 0_n \\ C_{Xv} & L^* \end{pmatrix}$$

где L^* - нижняя треугольная матрица размерности $n \times n$. Таким образом, можно разложить приращения винеровских процессов $dW_{v_1}(t), ..., dW_{v_n}(t)$, $dW_1(t), ..., dW_n(t)$ следующим образом:

$$\begin{pmatrix} dW_{v_1}(t) \\ ... \\ dW_{v_n}(t) \end{pmatrix} = \begin{pmatrix} d\bar{W}_1(t) \\ ... \\ d\bar{W}_n(t) \end{pmatrix}$$

$$\begin{pmatrix} dW_1(t) \\ ... \\ dW_n(t) \end{pmatrix} = C_{Xv} \begin{pmatrix} d\bar{W}_1(t) \\ ... \\ d\bar{W}_n(t) \end{pmatrix} + L^* \begin{pmatrix} d\bar{W}_{n+1}(t) \\ ... \\ d\bar{W}_{2n}(t) \end{pmatrix}$$

где $\bar{W}_1(t), ..., \bar{W}_{2n}(t)$ независимые винеровские процессы. Запишем представление $dW_i(t)$ в явной форме:

$$dW_i(t) = \rho_i d\bar{W}_i(t) + \sum_{j=1}^{i} L^*_{i,j} d\bar{W}_{n+j}(t)$$

Далее делаем замену в (19) и получаем:

$$dX_i(t) = (r - \frac{1}{2}v_i(t))dt + \sqrt{v_i(t)}\rho_i d\bar{W}_i(t) + \sum_{j=1}^{i} \sqrt{v_i(t)} L^*_{i,j} d\bar{W}_{n+j}(t).$$

Запишем в интегральной форме:

$$X_i(t+\delta) = X_i(t) + \int_t^{t+\delta} (r - \frac{1}{2}v_i(s))ds + \rho_i \int_t^{t+\delta} \sqrt{v_i(s)} d\bar{W}_i(s) +$$

$$+\sum_{j=1}^{i} L^*_{i,j} \int_t^{t+\delta} \sqrt{v_i(s)} d\bar{W}_{n+j}(s). \tag{30}$$

где $\delta > 0$. В начале метода MQE происходит моделирование следующих значений: $\left\{v_i(t), v_i(t+\delta), \int_t^{t+\delta} v_i(s)ds\right\}$, $i = 1, ..., n$. Значение $v_i(t+\delta)$ моделируется с помощью метода QE. Для моделирования $\int_t^{t+\delta} v_i(s)ds$ используется следующая аппроксимация:

$$\int_t^{t+\delta} v_i(s)ds \approx \delta(l_1 v_i(t) + l_2 v_i(t+\delta))$$

где $l_1 = l_2 = \frac{1}{2}$.

Далее запишем процесс для $v_i(t)$ в интегральной форме:

$$v_i(t+\delta) = v_i(t) + \kappa_i\bar{\theta}_i\delta - \kappa_i \int_t^{t+\delta} v_i(s))ds + \gamma_i \int_t^{t+\delta} \sqrt{v_i(s)} dW_{v_i}(s)$$

Тогда можем записать:

$$\int_t^{t+\delta} \sqrt{v_i(s)} d\bar{W}_i(s) = \int_t^{t+\delta} \sqrt{v_i(s)} dW_{v_i}(s) =$$

$$= \frac{1}{\gamma_i}(v_i(t+\delta) - v_i(t) - \kappa_i\bar{\theta}_i\delta + \kappa_i \int_t^{t+\delta} v_i(s)ds),\ i = 1, ..., n.$$

Чтобы промоделировать $\int_t^{t+\delta} \sqrt{v_i(s)} d\bar{W}_{n+j}(s)$, воспользуемся следующим представлением[26]:

$$\int_t^{t+\delta} \sqrt{v_i(s)} d\bar{W}_{n+j}(s) \overset{d}{=} Z_j \sqrt{\int_t^{t+\delta} v_i(s)ds},\ i, j = 1, ..., n. \tag{31}$$

где $\mathbf{Z} = (Z_1, ..., Z_n)^T$ - вектор с независимыми стандартно нормально распределенными компонентами. Итак, вычислив $\left\{v_i(t), v_i(t+\delta), \int_t^{t+\delta} v_i(s)ds\right\}$, $i = 1, ..., n$, и интегралы (31), можно вычисить значения $X_i(t+\delta)$, $i = 1, ..., n$. Для удобства имплементации запишем всю процедуру в матричной форме [26]. Предположим, что значения $\left\{v_i(t), v_i(t+\delta), \int_t^{t+\delta} v_i(s)ds\right\}$,

$i = 1, ..., n$ посчитаны, тогда введем следующие обозначения:

$$I_{ij} = \int_t^{t+\delta} \sqrt{v_i(s)} d\bar{W}_j(s),\ i, j = 1, ..., n,$$

$$s_i = \sqrt{\int_t^{t+\delta} v_i(s) ds},\ i = 1, ..., n,$$

$$f_i = \frac{1}{\gamma_i}(v_i(t+\delta) - v_i(t) - \kappa_i \bar{\theta}_i \delta + \kappa_i \int_t^{t+\delta} v_i(s) ds),\ i = 1, ..., n,$$

$$Z_j \sim \mathcal{N}(0, 1),\ j = 1, ..., n.$$

Тогда с учетом предыдущих соотношений получаем:

$$I_{i,i} = f_i,\ i = 1, ..., n,$$

$$I_{i,n+i} = s_i Z_j,\ i, j = 1, ..., n.$$

Последние два элемента в (30) можно представить в следующем виде:

$$\rho_i I_{i,i} + \sum_{j=1}^{i} L^*_{ij} I_{i,n+j} = \rho_i f_i + \sum_{j=1}^{i} L^*_{ij} s_i Z_j,\ i = 1, ..., n.$$

Тогда (30) можно представить в матричной форме:

$$\mathbf{X}(t+\delta) = \mathbf{X}(t) + \int_t^{t+\delta} (r - \frac{1}{2}\mathbf{V}(s)) ds + C_{Xv}\mathbf{f} + D_s L^* \mathbf{Z},$$

где

$$\mathbf{X}(t) = \begin{pmatrix} X_1(t) \\ ... \\ X_n(t) \end{pmatrix}, \mathbf{V}(t) = \begin{pmatrix} v_1(t) \\ ... \\ v_n(t) \end{pmatrix}, C_{Xv} = \begin{pmatrix} \rho_1 & 0 & ... & & 0 \\ 0 & \rho_2 & 0 & ... & 0 \\ & & ... & & \\ 0 & & ... & 0 & \rho_n \end{pmatrix},$$

$$\mathbf{f} = \begin{pmatrix} f_1 \\ ... \\ f_n \end{pmatrix}, \mathbf{Z} = \begin{pmatrix} Z_1 \\ ... \\ Z_n \end{pmatrix}, D_s = \begin{pmatrix} s_1 & 0 & ... & & 0 \\ 0 & s_2 & 0 & ... & 0 \\ & & ... & & \\ 0 & & ... & 0 & s_n \end{pmatrix}.$$

1.5 Оценивание структурных нот

Структурная нота представляет собой [30] долговую ценную бумагу, выпущенную финансовым учреждением. Ее доход может быть основан на индексах акций, отдельной акции, корзине акций, процентных ставках, товарах или иностранной валюте. Выплаты по структурной ноте происходят как у облигаций - в определенные даты, но так как она зависит от определенного базового актива, то выплаты в эти даты также будут зависеть от базового актива. В качестве базового актива рассматриваем группу акций. Тогда в дальнейшем рассматриваем выплаты по схеме, схожей с барьерным опционом: если минимальная стоимость акции из этой группы находится в определенном диапазоне, то выплата купонов осуществляется, если выше диапазона, то происходит выплата купона и затем происходил автокол, то есть, выплата номинальной стоимости, если же ниже диапазона, то выплата происходит позже, когда стоимость попадает обратно в диапазон.

Справедливая стоимость структурной ноты - это средние суммарные дисконтированные выплаты в будущем по этой структурной ноте. Основная задача состоит в том, чтобы определить справедливую стоимость структурной ноты. Если есть откалиброванная модель базового актива, то можно моделировать его пути с помощью методов Монте-Карло и считать на каждом пути дисконтированные выплаты по структурной ноте, а затем взять среднее значение.

Таким образом, получаем следующий алгоритм. Пусть имеется структурная нота с экспирацией через T лет, с барьером по выплатам $bar \in [0, 1]$ и с номинальной стоимость nom. Пусть даты $(t_1^*, ..., t_k^*)$ выплат расположены на отрезке $[0, T]$ следующим образом: $0 < t_1^* < ... < t_{k-1}^* < t_k^* = T$.

Пусть имеется откалиброванная модель $\mathbf{S}(t) = (S_1(t), ..., S_n(t))$ динамики группы из n акций. Промоделируем путь, в моменты времени $(t_1, ..., t_{\bar{N}})$

такие, что $(t_1^*, ..., t_k^*) \sqsubseteq (t_1, ..., t_{\bar{N}})$. Тогда получаем следующий набор значений: $\mathbf{S}(t_1), ..., \mathbf{S}(t_{\bar{N}})$. Тогда дисконтированная выплата $e^{-t_i^* r} C(t_i^*)$, где r - процентная ставка и $C(t_i^*)$ - выплата в дату t_i^*, в дату t_i^* будет осуществлена, если $S_{\bar{j}(i)}(t_i^*) > S_{\bar{j}(i)}(0) \times bar$, где

$$\bar{j}(i) = arg \min_{j(i)=\overline{1,n}} (\frac{S_j(t_i^*)}{S_j(0)}).$$

Если же $S_{\bar{j}(i)}(t_i^*) < S_{\bar{j}(i)}(0) \times bar$, то выплата $C(t_i^*)$ запоминается и выплачивается в следующую дату, когда неравенство $S_{\bar{j}(i)}(t_i^*) > S_{\bar{j}(i)}(0) \times bar$ выполнится. В дату экспирации происходит выплата номинальной стоимости и все выплаты, которые до этого не были выплачены, если $S_{\bar{j}(k)}(T) > S_{\bar{j}(k)}(0) \times bar$, если неравенство не выполняется, то выплачивается только $\frac{S_{\bar{j}(k)}(T)}{S_{\bar{j}(k)}(0)} \frac{1}{bar} \times nom$. Таким образом, получаем набор дисконтированных выплат вдоль пути. Суммируя эти выплаты вдоль пути, получим суммарные дисконтированные выплаты вдоль пути. Далее моделируем N путей и вычисляем суммарные дисконтированные выплаты на каждом пути. Таким образом, получаем набор $C_1, ..., C_N$, состоящий из N значений. Тогда оценка справедливой стоимости C структурной ноты будет равна:

$$C = \frac{\sum_{l=1}^{N} C_l}{N}.$$

Глава 2. Практическая часть

Описанную методологию применим к оцениванию справедливой стоимости структурных нот, зависящих от динамики акций. Соответственно, оцениваются структурные ноты, зависящие от одной, двух и трех акций. В качестве акций были выбраны три акции из сектора полупроводников: Nvidia (NVDA), Advanced Micro Devices (AMD), Qualcomm (QCOM), и три акции из разных секторов: UnitedHealth Group (UNH), Chevron (CVX), Boeing Company (BA). В скобочках указаны тикеры, по ним и будем далее обращаться.

Чтобы моделировать выплаты по структурным нотам, необходимо выбрать модели и откалибровать их. В качестве моделей были взяты модели Блэка-Шоулза (2), Хестона (5) и Бейтса (7).

В начале необходимо откалибровать одномерные модели, для этого нужно зафиксировать дату моделирования и получить данные по котировкам европейских опционов типа call и данные по процентной ставке в эту дату. Моделирование происходило в дату 28.04.2022, поэтому данные по опционам и по процентной ставке были взяты на сайте [28] на это число. Данные были взяты на цены американских опционов типа call, но в данной работе их цены приравниваются к ценам европейских опционов, так как обычно покупателю опциона выгоднее его перепродать, чем исполнять, так как в этом случае не потеряется его внешняя стоимость. В качестве процентной ставки была взята ставка SOFR на сайте [29], которая 28.04.2022 равнялась 0,28%.

Далее происходит калибровка одномерных моделей для каждой акции. В разложении (15) выберем $N = 500$, так как COS метод сходится экспоненциально [15] (при неразрыной функции плотности), то слишком большое N нет смысла выбирать, тем более, что далее будет использоваться генетический алгоритм для решения задачи (17), и в качестве отрезка [a, b] в (15) выберем отрезок (16), где возьмем $L = 12$. В качестве метода решения (17) используется метод дифференциальной эволюции [24], который по сути аналогичен генетическим алгоритмам, которые не требуют гладкости оптимизируемой функции. При калибровке параметров в моделях Хестона

и Бейтса используется следующая эвристика [15]: $\kappa = 0.5$, основанная на том, что параметры κ и γ оказывают один и тот же эффект на подразумеваемую волатильность, поэтому один из этих параметров имеет смысл зафиксировать, тем самым уменьшив размерность задачи.

Таким образом, получаем следующие параметры откалиброванных моделей:

	NVDA	AMD	QCOM	UNH	CVX	BA
σ	0.5316	0.5344	0.4253	0.3103	0.3156	0.4077

Таблица 1: Параметр в модели Блэка-Шоулза

	κ	γ	$\bar{v}$	v_0	ρ
NVDA	0.5	0.488	0.5	0.377	-0.99
AMD	0.5	0.5	0.294	0.455	-0.89
QCOM	0.5	0.772	0.427	0.273	-0.25
UNH	0.5	0.612	0.363	0.088	-0.69
CVX	0.5	0.369	0.079	0.152	-0.9
BA	0.5	0.26	0.223	0.204	-0.99

Таблица 2: Параметры в модели Хестона

	κ	γ	$\bar{v}$	v_0	ρ	ξ	μ_J	σ_J
NVDA	0.5	0.431	0.196	0.299	-0.84	0.204	-0.921	0.764
AMD	0.5	0.658	0.174	0.29	-0.41	0.484	-0.604	0.372
QCOM	0.5	0.795	0.177	0.251	-0.07	0.323	-0.444	0.05
UNH	0.5	0.36	0.101	0.069	-0.59	0.121	-1.114	0.042
CVX	0.5	0.215	0.068	0.129	-0.75	0.01	-0.189	0.054
BA	0.5	0.195	0.048	0.131	0.003	0.229	-0.792	0.3

Таблица 3: Параметры в модели Бейтса

Так как некоторые структурные ноты зависят от нескольких акций, то необходимо составить многомерные модели и откалибровать для каждой модели матрицу корреляций C_X, как в (23), между винеровскими процессами акций. Для этого нужно одномерные модели в риск-нейтральной

мере перевести в реальную меру, как было описано в разделе 1.3.1. Так как оценка параметров в реальной мере происходит на основе исторических данных на акции и чем больше данных тем точнее оценка, то в качестве промежутка, на котором рассматриваются исторические данные, был взят промежуток 30.04.2010 - 27.04.2022. Исторические данные за этот промежуток были взяты на сайте [28]. Калибровка параметров в реальной мере в случае модели Бейтса является нетривиальной задачей. Можно, например, воспользоваться алгоритмом, предложенным в [27]. Однако, после его реализации выясняется, что алгоритм получается очень требовательным к вычислительным мощностям (реализация алгоритма прикладывается в приложении), которых в нашем случае не так много, поэтому для простоты возьмем ту же оценку, что и для модели Хестона.

При подсчете ожидаемой корреляции между ценами двух акций, моделирование путей будет основываться на методе, предложенном в секции (1.3.2) для моделирования пуассоновских процессов совместных прыжков цен акций, так этот метод оказался более надежным в получении положительной определенности результирующей матрицы без использования регуляризации по сравнению с тем, где пути моделируются при полностью независимых прыжках. Если акции принадлежат одному сектору, то интенсивности прыжков считаются на основе подсчета совместных прыжков акций и сектора, если разным, то интенсивности прыжков считаются на основе подсчета совместных прыжков акций и всего рынка в целом. Так как в нашем случае были взяты акции из сектора полупроводников, то в качестве ETF на этот сектор был выбран Ishares semiconductor ETF (SOXX). Для всего рынка был выбран SPDR S&P 500 ETF TRUST (SPY). Подсчет прыжков осуществлялся на том же временном промежутке, на котором оценивались параметры в реальной мере.

Теперь можно приступить к моделированию справедливой стоимости структурных нот. Оцениваем следующие структурные ноты: зависящие только от динамики одной акции - NVDA, AMD, QCOM, UNH, CVX, BA, от двух акций - (NVDA, QCOM), (NVDA, AMD), (QCOM, AMD), (BA, CVX) и (BA, UNH), от трех акций - (NVDA, QCOM, AMD) и (UNH, CVX, BA).

Во всех случаях считаем, что номинальная стоимость структурной ноты равна 1000, купоны 4,5% выплачиваются ежеквартально, если цена акции с худшей динамикой относительно своей стоимости в начальный момент времени выше барьерного значения 0,65. Если цена акции с худшей динамикой окажется выше своего начального значения, то происходит автокол, при котором выплачивается номинальная стоимость и все невыплаченные купоны, если это происходит в первый квартал, то выплачиваются 2 купона по 4,5% вместо одного. Если в дату экспирации цена акции с худщей динамикой $\bar{S}$ относительно своей начальной стоимости $\bar{S}_0$ окажется ниже барьерного значения 0,65, то выплатится $\frac{\bar{S}}{\bar{S}_0}\frac{1}{0.65}1000$, если выше, то выплатится номинальная стоимость минус страховка в виде 1% от номинальной стоимости плюс все невыплаченные купоны.

Справедливую стоимость каждой структурной ноты моделируем для экспираций T = 1, 2, 3, 5, 10 лет. Также для каждой структурной ноты оцениваем вероятность p получения худшего сценария в момент экспирации (с выплатой $\frac{\bar{S}}{\bar{S}_0}\frac{1}{0.65}1000$) и доверительные интервалы, которые не отображаем, чтобы было нагляднее. Затем для каждой структурной ноты строятся графики, на которых отражается справедливая стоимость структурной ноты в зависимости от экспирации для каждой модели, и также строятся отдельно графики зависимости вероятности худшего сценария от экспирации.

Таким образом, получаем следующие графики - рисунки (4) - (9).

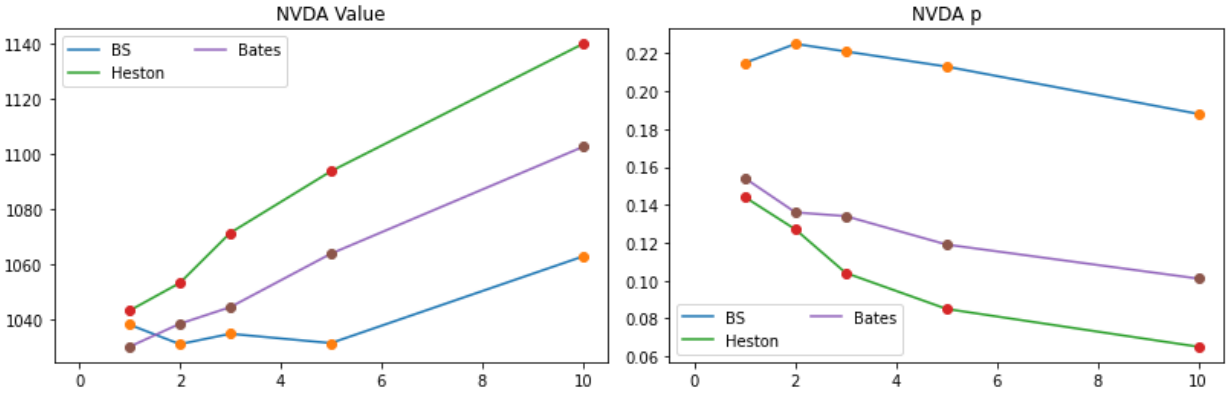

Рис. 4: Модельные значения для NVDA

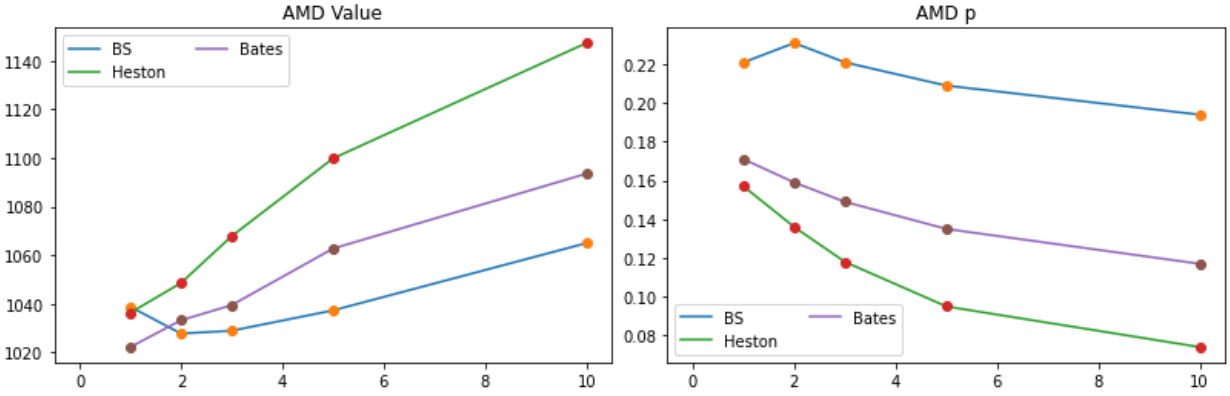

Рис. 5: Модельные значения для AMD

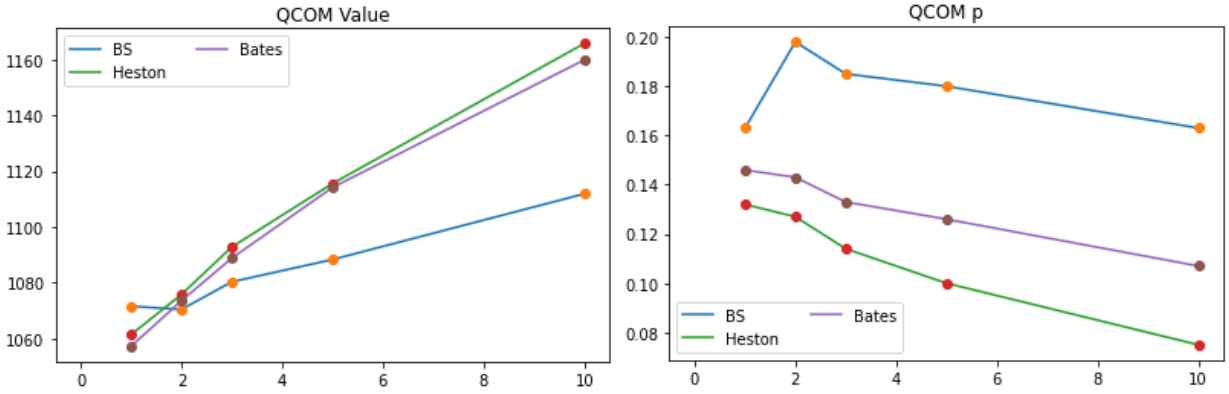

Рис. 6: Модельные значения для QCOM

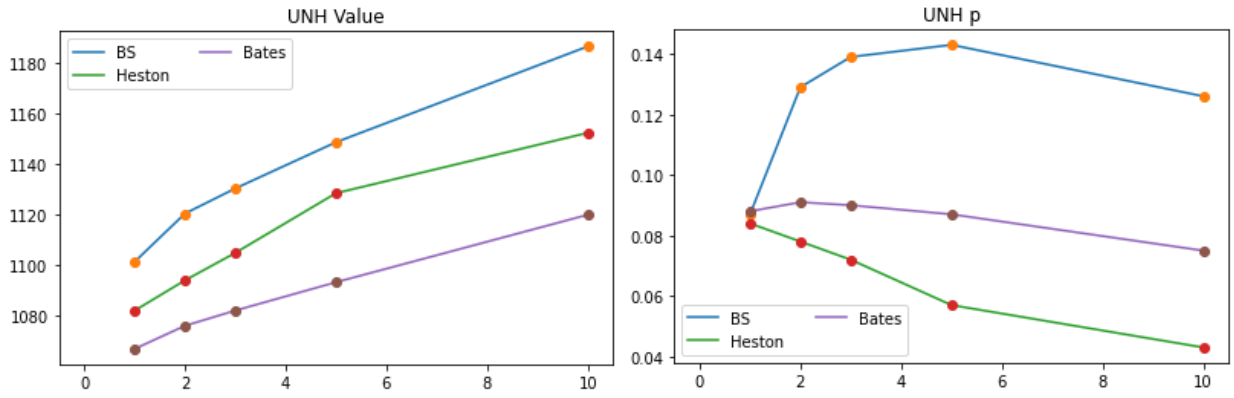

Рис. 7: Модельные значения для UNH

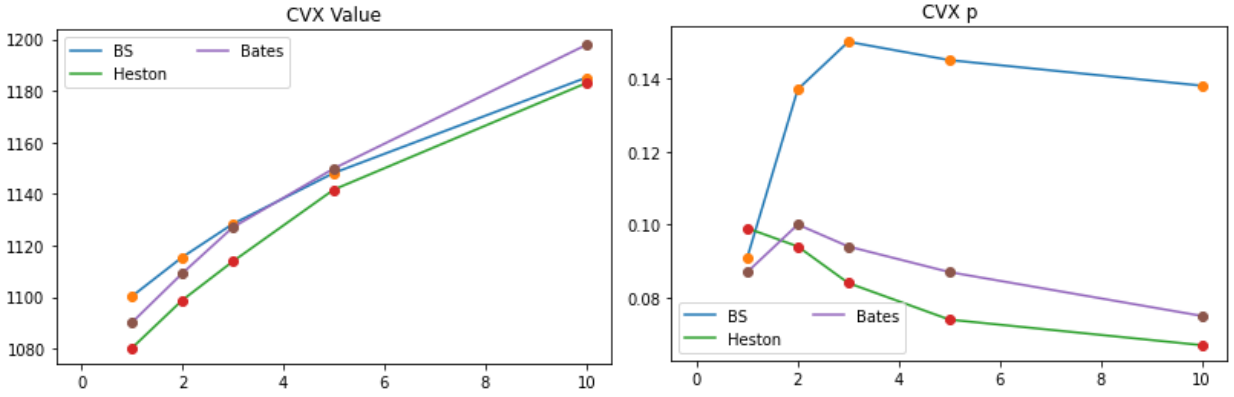

Рис. 8: Модельные значения для CVX

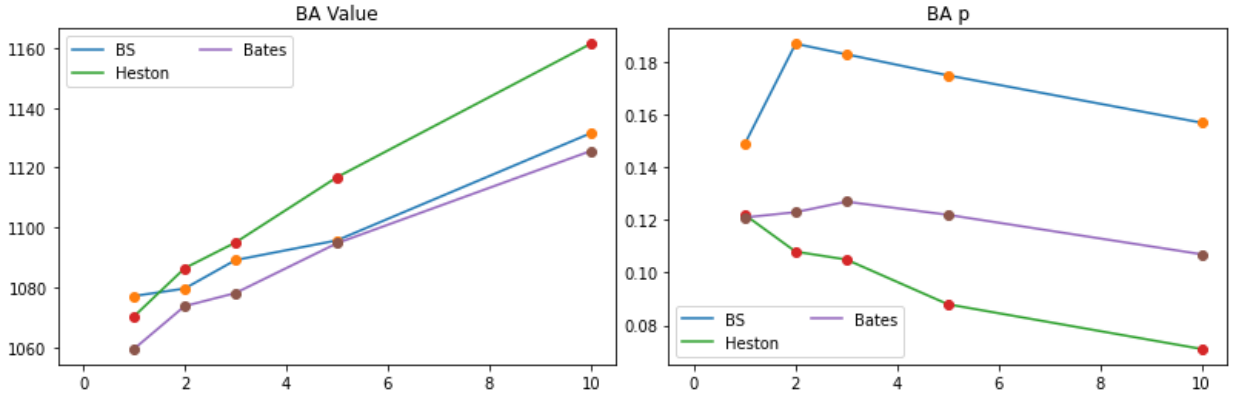

Рис. 9: Модельные значения для BA

По рисункам (4) - (9) можно сделать следующие выводы:

1) При увеличении экспирации структурной ноты ее стоимость растет, а вероятность плохого сценария падает. Это связано с тем, что по ней происходят выплаты, когда цена акции находится в диапазоне, и чем дольше она в нем находится, тем больше выплатится купонов, тем больше дисконтированных выплат и тем больше ее справедливая стоимость.

2) Для сектора полупроводников модель Хестона дает наиболее оптимистичную оценку, тогда как модель Блэка-Шоулза самую пессимистичную. Оценка для модели Бейтса лежит между оценками для моделей Хестона и Блэка-Шоулза.

Для структурных нот, зависящих от двух акций, теперь уже строятся по 4 графика на каждом рисунке (10) - (14): для моделей Блэка-Шоулза, Хестона, Бейтса и Бейтса с модификацией из секции (1.3.2).

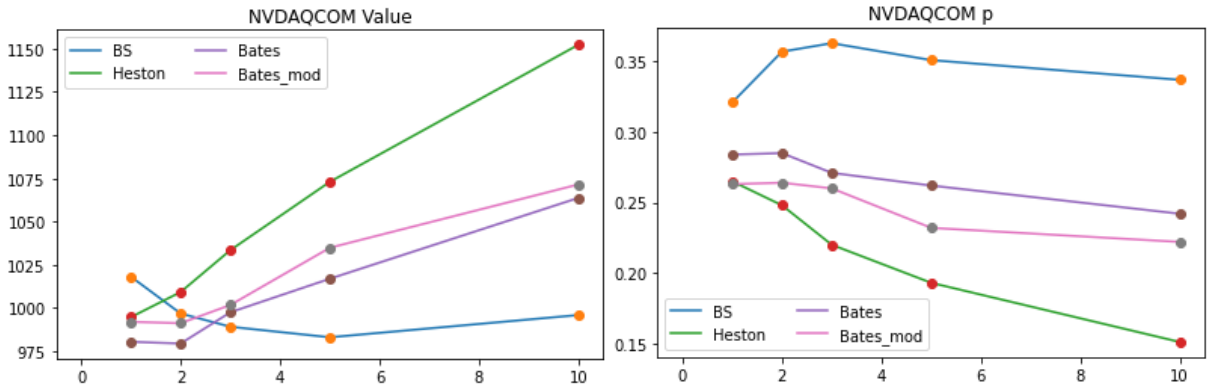

Рис. 10: Модельные значения для NVDA, QCOM

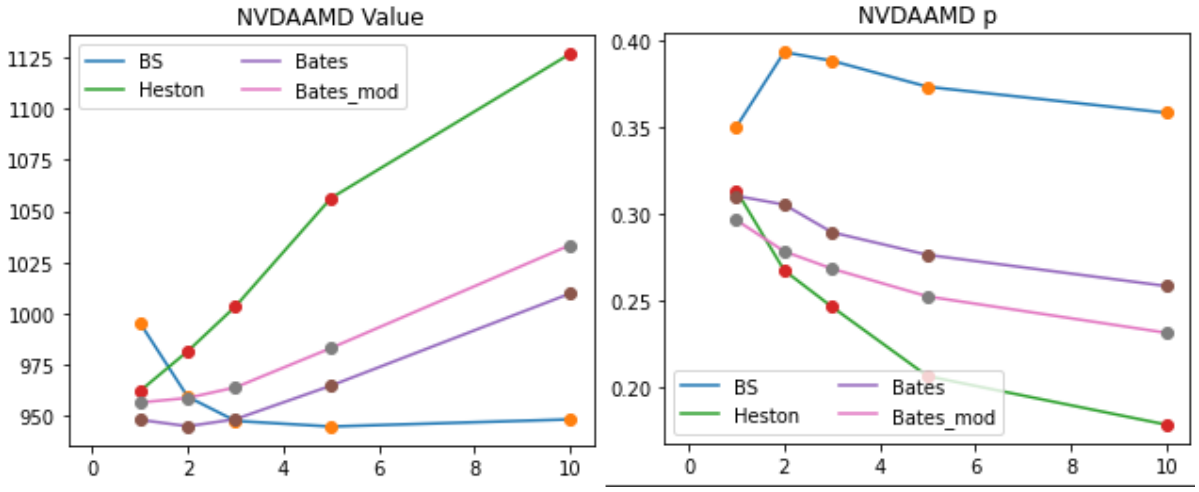

Рис. 11: Модельные значения для NVDA, AMD

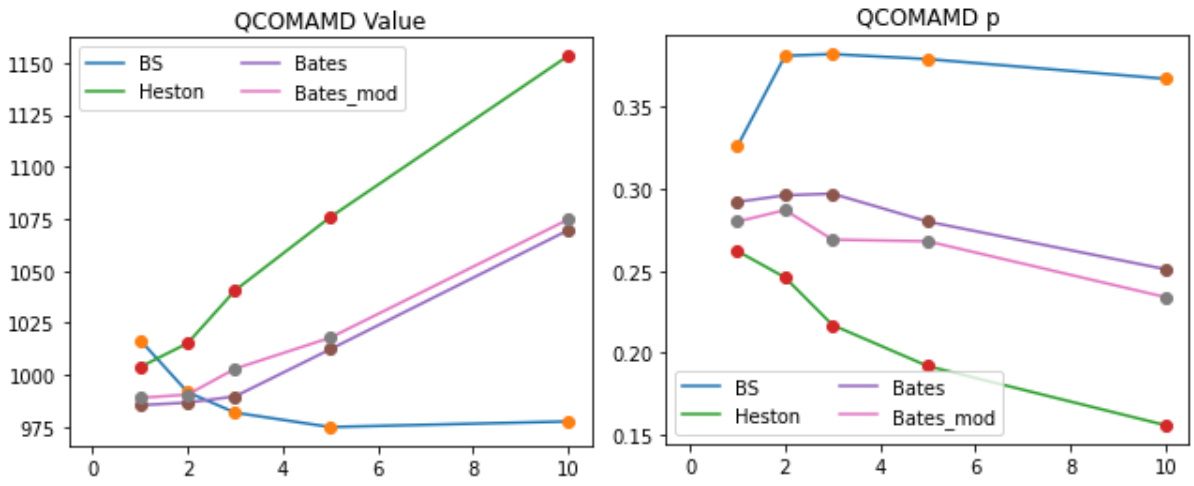

Рис. 12: Модельные значения для QCOM, AMD

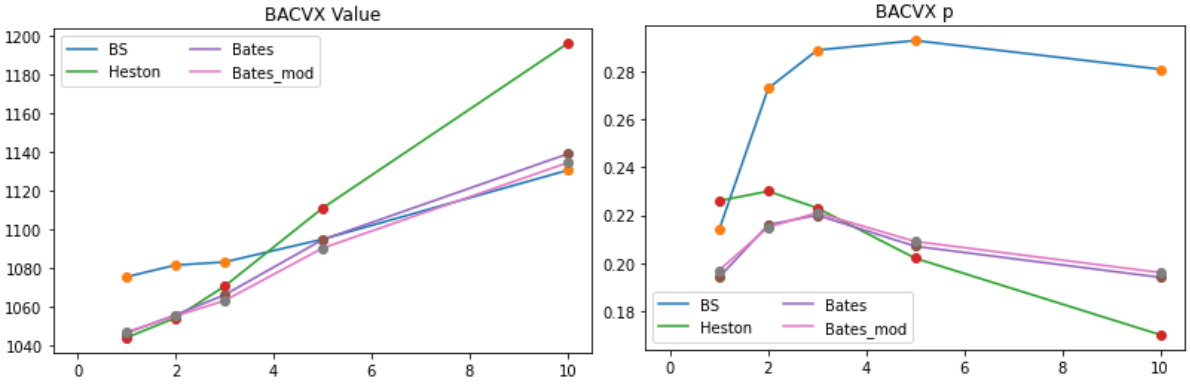

Рис. 13: Модельные значения для BA, CVX

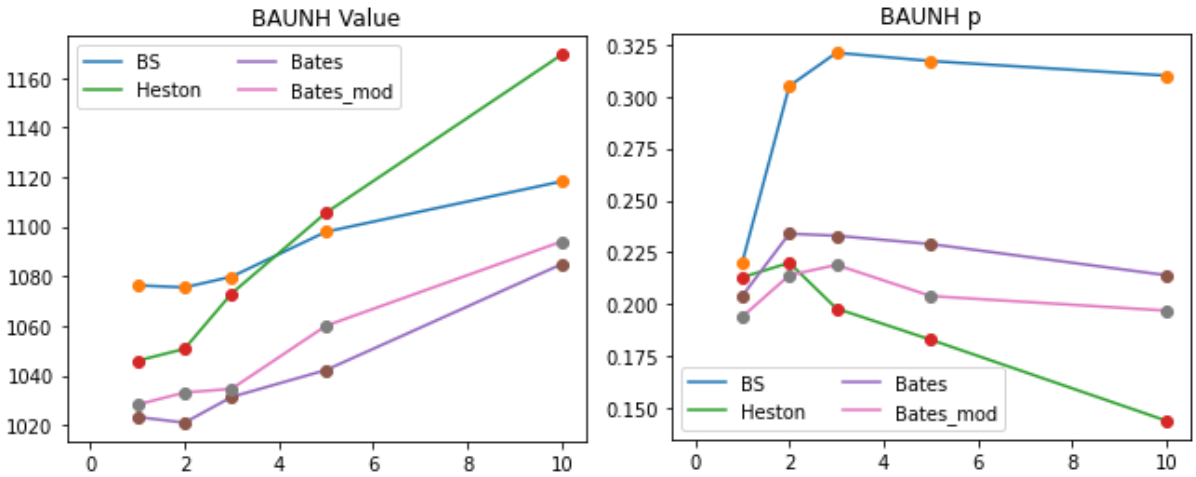

Рис. 14: Модельные значения для BA, UNH

По рисункам (10) - (14) можно сделать следующие выводы:

1) Как и ранее, при увеличении экспирации стоимость структурных нот растет. Однако теперь для экспираций меньше трех лет в случае структурной ноты на акции из одного сектора их справедливая стоимость по моделям Блэка-Шоулза, Бейтса, Бейтса с модификацией меньше номинальной стоимости, но при экспирациях выше трех лет, выравнивается.

2) Как и ранее, у модели Хестона самая оптимистичная оценка, тогда как у модели Блэка-Шоулза самая пессимистичная, при этом нарушается логика повышения стоимости структурной ноты при увеличении экспирации. Модель Бейтса с модификацией немного оптимистичней обычной модели Бейтса. Заметим, что теперь вероятность плохого сценария увеличилась примерно в два раза в виду того, что выплата купонов привязана к динамике наихудшей акции. Следовательно, среди двух акций возьмется наихудшая, поэтому риск плохого сценария возрастет в два раза.

Также построим графики оценочных стоимостей структурных нот, зависящих от трех акций - рисунки (15) - (16).

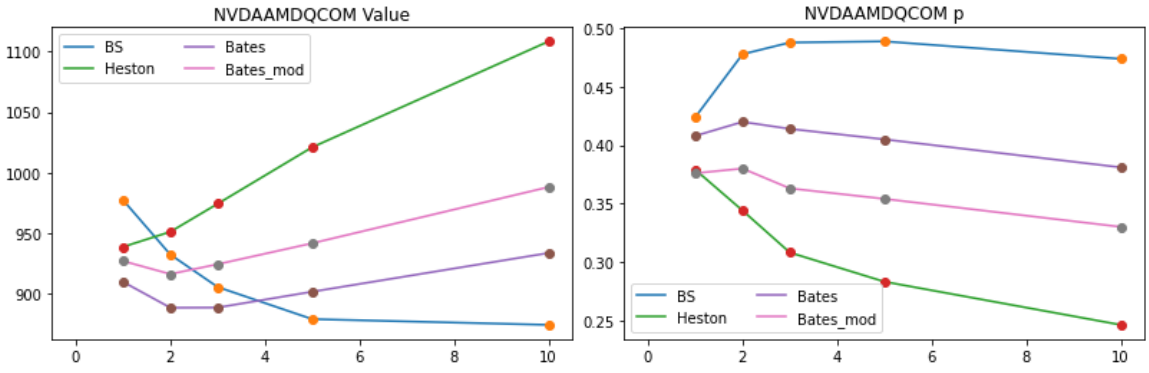

Рис. 15: Модельные значения для NVDA, AMD, QCOM

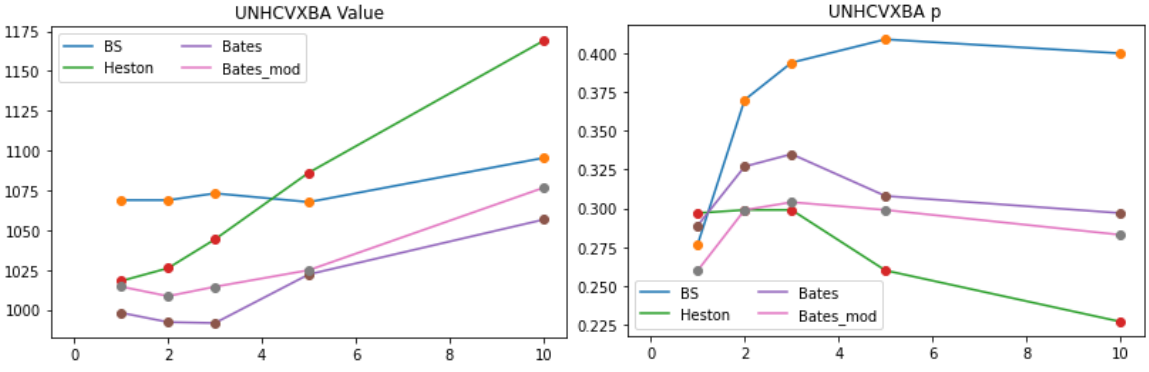

Рис. 16: Модельные значения для UNH, CVX, BA

Как и ранее, при увеличении экспирации стоимость структурных нот растет. Однако теперь для всех моделей в случае структурной ноты на акции из одного сектора со сроком экспирации меньше пяти лет ее стоимость становится меньше номинальной. Модель Блэка-Шоулза самая пессимистичная, при этом нарушается логика повышения стоимости структурной ноты при увеличении экспирации. Такой же эффект у моделей Бейтса с модификацией и без для экспираций меньше трех лет, но в случае модели Бейтса с модификацией этот эффект сильнее сглажен. Риск плохого сценария теперь увеличился примерно в три раза, в виду наличия трех акций и особенности моделирования выплат по структурной ноте. Во всех случаях модель Хестона показывает наиболее оптимистичную оценку.

Подводя итоги моделирования, отметим, что во всех случаях модель Хестона показывала наиболее оптимистичную динамику и при этом проявляла устойчивость к увеличению числа акций. Модель Блэка-Шоулза оказалась самой пессимистичной, при этом неустойчивой к увеличению числа акций. Модель Бейтса с модификацией показала более устойчивую и схожую с моделью Хестона динамику по сравнению с моделью Бейтса без этой модификации на малых экспирациях, при этом при калибровке корреляций она тоже проявила устойчивость относительно получения положительно определенного результата, поэтому при моделировании многомерных процессов использование модификации из (1.3.2) стоит иметь в виду. Во всех случаях стоимость структурной ноты росла при увеличении экспирации, а риск плохого сценария падал, при этом на дальних экспирациях оценочная справедливая стоимость оказывается выше номинальной.

Заключение

В ходе работы были выбраны стохастические модели и откалиброваны при помощи методов, описанных в [15], [26], [13], после чего были применены к оцениванию деривативов на примере структурных нот. После было проведено сравнение этих моделей, в ходе которого выяснилось, что наиболее неустойчивой моделью относительно повышения размерности модели оказалась модель Блэка-Шоулза, наиболее простая из рассмотренных, устойчивыми моделями являются модель Хестона и модель Бейтса с методом моделирования совместной динамики прыжков и амплитуд прыжков, разработанным в данной работе и описанным в секции (1.3.2).

В ходе сравнения моделей выяснилось, что оценочная справедливая стоимость структурной ноты имеет тенденцию к росту при увеличении экспирации, а риск наихудшего сценария по выплатам падает с течением времени, при этом при экспирациях больше пяти лет оценочная справедливая стоимость структурной ноты становится выше номинальной.

Список литературы

[1] Булинский А.В., Ширяев А.Н. Теория случайных процессов. М.: ФИЗМАТЛИТ,2005. — 400 с.

[2] Буре В.М., Парилина Е.М. Теория вероятности и математическая статистика. СПб.: Издательство «Лань», 2013. — 416 с.

[3] Колесников А.В. Лекции по теории вероятности

[4] Маталыцкий М. А., Хацкевич Г. А.. Теория вероятностей, математическая статистика и случайные процессы. Минск : Выш. шк., 2012. – 720 с.

[5] Шарп У., Александер Г., Бэйли Дж. Инвестиции: Пер. с англ. М.: ИНФАРА-М, 2001. - 1028 с.

[6] Ширяев А. Н. Основы стохастической финансовой математики : В 2-х т. Т. 1 : Факты, модели. М. : МЦНМО, 2016. - 440 с.

[7] Ширяев А. Н. Основы стохастической финансовой математики : В 2-х т. Т. 2 : Теория. М. : МЦНМО, 2016. - 464 с.

[8] Andersen L. Efficient Simulation of the Heston Stochastic Volatility Model. L., 2007

[9] Bakshi G., Cao C., Chen Z.Empirical Performance of Alternative Option Pricing Models // The Journal of Finance. 1997. Vol. 52. No 5. P. 2003-2049.

[10] Black F.,Scholes M. (1973). The Pricing of Options and Corporate Liabilities // Journal of Political Economy. 1973. Vol.81. N 3. Pp. 637–654.

[11] Boukai B. On the RND under Heston's stochastic volatility model. 2021 // https://doi.org/10.48550/arXiv.2101.03626

[12] Brandimarte P. Handbook in Monte Carlo Simulation: Applications in Financial Engineering, Risk Management, and Economics. N.Y., 2014. - 662 p.

[13] Dimitroff G., Lorenz S., Szimayer A. A Parsimonious Multi-Asset Heston Model: Calibration and Derivative Pricing.2009 // SSRN eLibrary.

[14] Ghosh S., Henderson S.G. Behavior of the NORTA method for correlated random vector generation as the dimension increases // ACM Transactions on Modeling and Computer SimulationVolume. 2003. Issue 13. 3July. Pp. 276–294

[15] Grzelak L.A. Mathematical Modeling and Computation in Finance. Amsterdam, 2020. - 556 p.

[16] Hagan P.S., Kumar D., Lesniewski A., Woodward D.E. Managing smile risk. Wilmott, 2002. - 108 p.

[17] Heston S.L. A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options // The Review of Financial Studies. 1993. Vol. 6. Issue 2. Pp. 327–343.

[18] Hull J. Options, Futures, and Other Derivatives. N.Y., 2021. - 882 p.

[19] Jackel P. Monte Carlo methods in finance. L., 2002. - 409 p.

[20] Kreinin A. Correlated Poisson Processes and Their Applications in Financial Modeling. // Financial Signal Processing and Machine Learning. April 2016

[21] Loerx A., Sachs E.W. Model Calibration in Option Pricing. 2012. Vol. 17. No 1 // https://doi.org/10.24200/squjs.vol17iss1pp84-102

[22] Note Sur Une Méthode de Résolution des équations Normales Provenant de L'Application de la MéThode des Moindres Carrés a un Système D'équations Linéaires en Nombre Inférieur a Celui des Inconnues // Application de la Méthode a la Résolution D'un Système Defini D'éQuations LinéAires. Bull. Géodésique. 1924. N 2. Pp. 67–77 // https://doi.org/10.1007/BF03031308

[23] Romo E., Ortiz-Gracia L. SWIFT calibration of the Heston model // Mathematics. 2021. Vol. 9. N 5. Pp. 529-539 // https://doi.org/10.3390/math9050529

[24] Storn R., Price K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces // Journal of Global Optimization. 1997. Vol. 11. Pp. 341–359

[25] Sydow L. and al. BENCHOP – The BENCHmarking project in option pricing. 2015. Pp. 2361-2379 // https://doi.org/10.1080/00207160.2015.1072172

[26] Wadman W.S. An advanced Monte Carlo method for the multi-asset Heston model. November 2010 // https://www.researchgate.net/publication/325952691

[27] Wilson D. E. A.. The Estimation of Stochastic Models in Finance with Volatility and Jump Intensity. A thesis presented to the University of Waterloo in fulfillment of the thesis requirement for the degree of Doctor of Philosophy in Statistics Waterloo. Ontario. Canada. 2018.

[28] https://finance.yahoo.com/

[29] https://fred.stlouisfed.org/series/SOFR

[30] https://www.investopedia.com/terms/s/structurednote.asp

Приложение

Функции для модели Блэка-Шоулза.

```
#Ввод необходимых библиотек
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as st
import enum
import scipy.optimize as optimize
import datetime
import statsmodels.api as sm
import pandas as pd
#!pip install yahoo_fin
#from yahoo_fin import options
import time
import pickle
from scipy.interpolate import CubicSpline
from tqdm import tqdm

    # Вводим мнимую единицу i
np.set_printoptions(precision = 5)
i = np.complex(0.0,1.0)

    # Вводим класс отражающий тип опциона

    class OptionType(enum.Enum):
CALL = 1.0
PUT = -1.0

    # Функция вычисления стоимости опциона
def CallPutOptionPriceCOSMthd(par, CP, S0, r, tau, K, N, L):
# par - параметры модели
# CP - тип опциона
```

```
# S0 - начальная цена базового актива
# r - процентная ставка (постоянная)
# tau - время до экспирации
# K - лист страйков
# N - число элементов в разложении
# L - определяет величину отсечения пространства (L=8 или L=10)
# меняем размер K - теперь K вектор
if K is not np.array:
K = np.array(K).reshape([len(K), 1])

        i = np.complex(0.0, 1.0)
x0 = np.log(S0 / K)

        # усеченное пространство
a = 0.0 - L * np.sqrt(tau)
b = 0.0 + L * np.sqrt(tau)

        # Суммирование по k = 0 до k = N-1
k = np.linspace(0, N-1, N).reshape([N, 1])
u = k * np.pi / (b - a);

        # определяем коэффициенты и характеристическую функцию
cf = ChFBSModel(r,tau, par)
H_k = CallPutCoefficients(CP,a,b,k)
mat = np.exp(i * np.outer((x0 - a) , u))
temp = cf(u) * H_k
temp[0] = 0.5 * temp[0]
# суммируем полученные коэффициент
ы val = np.exp(-r * tau) * K * np.real(mat.dot(temp))
return val

        # вычисление коэффициентов для соответствующего типа опционов
def CallPutCoefficients(CP,a,b,k):
```

```
if CP==OptionType.CALL:
c = 0.0
d = b
coef = Chi_Psi(a,b,c,d,k)
Chi_k = coef["chi"]
Psi_k = coef["psi"]
if a < b and b < 0.0:
H_k = np.zeros([len(k),1])
else:
H_k = 2.0 / (b - a) * (Chi_k - Psi_k)
elif CP==OptionType.PUT:
c = a
d = 0.0
coef = Chi_Psi(a,b,c,d,k)
Chi_k = coef["chi"]
Psi_k = coef["psi"]
H_k = 2.0 / (b - a) * (- Chi_k + Psi_k)

        return H_k


        # вспомогательная функция для вычисления коэффициентов в сумме
def Chi_Psi(a,b,c,d,k):
psi = np.sin(k * np.pi * (d - a) / (b - a)) - np.sin(k * np.pi * (c - a)/(b - a))
psi[1:] = psi[1:] * (b - a) / (k[1:] * np.pi)
psi[0] = d - c


        chi = 1.0 / (1.0 + np.power((k * np.pi / (b - a)) , 2.0))
expr1 = np.cos(k * np.pi * (d - a)/(b - a)) * np.exp(d) - np.cos(k * np.pi
* (c - a) / (b - a)) * np.exp(c)
expr2 = k * np.pi / (b - a) * np.sin(k * np.pi *
(d - a) / (b - a)) - k * np.pi / (b - a) * np.sin(k
* np.pi * (c - a) / (b - a)) * np.exp(c)
```

```
chi = chi * (expr1 + expr2)

        value = "chi":chi,"psi":psi
return value

        # вычисление характеристической функции в модели Блэка-Шоулса
def ChFBSModel(r,tau,par):

        sigma = par[0]

        i = np.complex(0.0,1.0)

        cf = lambda u: np.exp((r-(sigma ** 2) / 2) * i * u * tau - (sigma ** 2 )
* (u**2) * tau / 2)
return cf

        # моделирование двухмерной коррелированной модели Блэка-Шоулса
на основе моделирования методом Эйлера
# возвращает 2 путя, и время
def BS2path(par1, par2, NoOfSteps, T, mu, S_01, S_02, rho12):
# параметры (1=kappa, 2=gamma, 3=vbar, 4=v0, 5=rho)
# мгновенная кореляция между активами
Z1 = np.random.normal(0.0,1.0,[1, NoOfSteps])
Z2 = np.random.normal(0.0,1.0,[1, NoOfSteps])

        W1 = np.zeros([1, NoOfSteps + 1])
W2 = np.zeros([1, NoOfSteps + 1])

        X1 = np.zeros([1, NoOfSteps + 1])
X2 = np.zeros([1, NoOfSteps + 1])

        X1[:,0] = np.log(S_01)
X2[:,0] = np.log(S_02)
```

```
time = np.zeros([1, NoOfSteps+1])
dt = T / float(NoOfSteps)
for i in range(0,NoOfSteps):

    Z2[:,i] = rho12 * Z1[:,i] + np.sqrt(1.0-rho12**2) * Z2[:,i]

    W1[:,i+1] = W1[:,i] + np.power(dt, 0.5)*Z1[:,i]
W2[:,i+1] = W2[:,i] + np.power(dt, 0.5)*Z2[:,i]

    X1[:,i+1] = X1[:,i] + (mu[0] - 0.5 * par1 * par1) * dt + par1 * (W1[:,i+1]-
W1[:,i])
X2[:,i+1] = X2[:,i] + (mu[1] - 0.5 * par2 * par2) * dt +par2 * (W2[:,i+1]-W2[:,i])

    time[0][i+1] = time[0][i] + dt
return np.exp(X1), np.exp(X2), time

    # функция для подсчета выборочной корреляции между двумя приращениями логарифмических цен двух путей
def EmpCorr(path1, path2):
return np.corrcoef([np.log(path1[j + 1]/path1[j]) for j in range(len(path1) - 1)],
[np.log(path2[j + 1]/path2[j]) for j in range(len(path2) - 1)])[0][1]

    # функция для подсчета ожидаемой выборочной корреляции
def E_rho_emp(N, par1, par2, NoOfSteps, T, mu, S_01, S_02, rho12):
s = 0
for i in range(N):
a = BS2path(par1, par2, NoOfSteps, T, mu, S_01, S_02, rho12)
s += EmpCorr(a[0][0], a[1][0])
return s / N

    # калибровка(на основе метода бисекций) для вычисления мгновенной корреляции между двумя активами при данной исторической корреляции и откалиброванными параметрами модели
```

```
def Bis_opt(N, par1, par2, NoOfSteps, T, mu, S_01, S_02, rho_emp, eps):
# N - количество слагаемых для вычисления матожидания
# par1, par2 - откалиброванные параметры модели для 1 и 2 актива соответственно
# NoOfSteps - количество шагов в моделировании путей
# S_01, S_02 - начальные цены активов
# rho_emp - историческая корреляция
# eps - допустимая погрешность
rho_l = -1
rho_u = 1
E_l = E_rho_emp(N, par1, par2, NoOfSteps, T, mu, S_01, S_02, rho_l) - rho_emp
rho_emp_new = 0.5 * (rho_l + rho_u)
E_delta = E_rho_emp(N, par1, par2, NoOfSteps, T, mu, S_01,
S_02, rho_emp_new) - rho_emp
if (rho_emp > 0.99 and rho_emp <= 1) or (rho_emp < -0.99 and rho_emp >= -1.0):
eps = 1e-2

        while abs(E_delta) > eps:
if E_l * E_delta < 0:
rho_u = rho_emp_new
rho_emp_new = 0.5 * (rho_l + rho_u)
E_delta = E_rho_emp(N, par1, par2, NoOfSteps, T, mu, S_01, S_02,
rho_emp_new) - rho_emp

        else:
rho_l = rho_emp_new
E_l = E_delta
rho_emp_new = 0.5 * (rho_l + rho_u)
E_delta = E_rho_emp(N, par1, par2, NoOfSteps, T, mu,
S_01, S_02, rho_emp_new) - rho_emp
```

return rho_emp_new

#Проверка, что все элементы вектора положительные. Нужно для проверки положительной определенности матрицы.
def check_pos_def(vec):
for i in vec:
if i <= 0:
return False
return True

# нахождение индекса элемента массива, у которого округленное значение совпадает с elem
def findind(arr, elem, par = 4):
for i, j in enumerate(arr):
if round(j, par) == elem:
return i
return False

разложение Холецкого, предполагается, что на вход подается положительно определенная симметричная матрица. После выполнения процедуры регуляризации
#в python все равно может оказаться, что под корнем будет очень маленькое отрицательное значение, тогда оно заменяется на очень маленькое положительное значение
def chol(M):
L = np.zeros([len(M[0]), len(M[0])])
for j in range(len(M[0])):
for i in range(j, len(M[0])):
if i == 0 and j == 0:
L[i, j] = np.sqrt(M[i, j])
if i == j and i != 0 and j != 0:
if M[i, j] - np.sum(np.array([L[i, k] ** 2 for k in range(j)])) <= 0:
L[i, j] = 10e-5

```
else:
L[i, j] = np.sqrt(M[i, j] - np.sum(np.array([L[i, k] ** 2 for k in range(j)])))
else:
L[i, j] = (M[i, j] - np.sum(np.array([L[j, k] * L[i, k ] for k in range(j)])))/L[j, j]
return L


    # Процедура регуляризации
def Janckel_reg(M):
w, v = np.linalg.eig(M)


    if check_pos_def(w) == True:
return chol(M)
else:
Lambda = np.zeros([len(w), len(w)])
sp = np.zeros([len(w)])
for i in range(len(w)):
if w[i] >= 0:
Lambda[i][i] = w[i]
sp[i] = w[i]
T = np.zeros([len(w), len(w)])
for i in range(len(w)):
T[i][i] = 1/(np.power(v[i], 2).dot(sp))


    B = np.dot(np.dot(np.sqrt(T), v), np.sqrt(Lambda))
Cnew = np.dot(B, np.transpose(B))
w, v = np.linalg.eig(Cnew)
for i in range(len(Cnew)):
Cnew[i, i] = 1
return chol(Cnew)


    # генерация многомерного пути динамики акций в модели Блэка-Шоулза.
def multi_asset_path(par_model, corr_matrix, parS0, T, r, NoOfSteps):
```

```
# на вход:
# par_model - лист параметров моделей, в случае модели Блэка-Шоулза одномерный
# Матрица посчитанных мгновенных корреляций между броуновскими движениями активов
# parS0 - одномерный лист начальных стоимостей акций
# T - время экспирации
# r - процентная ставка
# NoOfSteps - количество шагов

        X = np.zeros([len(parS0), NoOfSteps+1])
# проводим регуляризацию, если матрица неположительно определена, на выходе разложение Холецкого
L = Janckel_reg(corr_matrix)
sigma = np.zeros([1, len(parS0)])
Z = np.random.normal(0.0,1.0,[len(parS0), NoOfSteps])
W = np.zeros([len(parS0), NoOfSteps + 1])
for i in range(len(parS0)):
X[i, 0] = np.log(parS0[i])
sigma[0, i] = par_model[i]

        time = np.zeros([NoOfSteps+1])
dt = T / float(NoOfSteps)
for i in range(0, NoOfSteps):
time[i+1] = time[i] + dt
W[:,i+1] = W[:,i] + np.power(dt, 0.5)*Z[:,i]
X[:, i+1] = X[:, i] + (r - sigma * sigma/2) * dt +sigma * np.dot(L, (W[:,i+1]-W[:,i])).reshape([1, len(parS0)])[0]

        return np.exp(X), time

        # моделирование стоимости структурной ноты
# возвращает оценку справедливой стоимости, оценку вероятности плохого
```

сценария по выплатам, левую и правую границы доверительного интервала с уровнем доверия 95%
# В модели Бейтса с модификацией будет еще один дополнительный параметр intens
# intens - лист оцененных влияний сектора на интенсивности прыжков
def autocallbondprice(par_model, rho_emp_matr, par_S_0, T, r, NoOfSteps, N, dates, coupon, barier, autocall, nom, ins):
# par_model - лист откалиброванных параметров модели
# rho_emp_matr - матрица откалиброванных мгновенных корреляций
# par_S_0 - лист начальных стоимостей активов
# T - экспирация структурнйо ноты
# r - процентная ставка
# NoOfSteps - количество шагов моделирования многомерного пути
# N - количество путей
# dates - даты выплат купонов
# coupon - значение купона
# barier - барьер для выплат
# autocall - значение при котором произойдет автокол (может быть как $>$ 1, так и $<$ 1)
# nom - номинальная стоимость структурной ноты
# ins - страховка

price = []
count = 0
for i in range(N):
C = 0
memory = 0
paths = multi_asset_path(par_model, rho_emp_matr,
par_S_0, T, r, NoOfSteps)
for k in range(len(par_S_0)):
# нормируем каждый путь актива в многомерном пути на начальное значение этого путя для удобства дальнейшего сравнения с барьерным значением

```
paths[0][k, :] = paths[0][k, :] / par_S_0[k]
for l, j in enumerate(dates):
# находим минимальное значение отношения в дату выплат
d = min(paths[0][:, findind(paths[1], j)])
if d < barier and l == len(dates) - 1:
C += np.exp(- r * j)*(d * nom / barier)
count += 1
elif barier <= d <= autocall and l == len(dates) - 1:
memory += 1
C += np.exp(- r * j)*(memory * nom * coupon / 100 + (1 - ins) * nom)
elif d > autocall and l == 0:
C = np.exp(- r * j)*(2 * nom * coupon / 100 + nom)
break
elif d > autocall:
memory += 1
C += np.exp(- r * j)*(memory * nom * coupon / 100 + nom)
break
elif d < barier:
memory += 1
elif barier <= d <= autocall:
memory += 1
C += np.exp(- r * j)*(memory * nom * coupon / 100)
memory = 0
price.append(C)
price = np.array(price)
z = 1.96
m = np.mean(price)
s = np.std(price, ddof=1)
return m , round(count / N, 3), m - z * s / np.sqrt(N) , m + z * s / np.sqrt(N)

    # калибровка модели Блэка-Шоулза
def Calibration_BS(CP, S0, r, N, L, bounds, optimizer, market, expiration,
strikes):
```

```
# CP - тип опциона
# S0 - начальная цена базового актива
# r - процентная ставка (постоянная)
# N - число элементов в разложении
# L - определяет величину отсечения пространства (L=8 или L=10)
# bounds - границы для параметров
# optimizer - тип оптимизации : в данной работе применяется дифференциальная эволюция
# market - данные, по которым калибруют (даны в виде подразумеваемых волатильностей)
# expiration - вектор времен до экспирации
# strikes - вектор страйков

    # функция ошибок
def Err_fun(x):
# внутри суммирование идет по страйкам при фиксированной экспирации, а далее идет суммирвоание по экспирациям
return np.sum(np.array([np.sum((CallPutOptionPriceCOSMthd(x, CP, S0, r, expiration[i], strikes, N, L).reshape([1,len(strikes)])
- np.array(market[i]).reshape([1,len(strikes)])) ** 2) for i in range(len(expiration))]))
# оптимизация при заданных границах
res = optimizer(Err_fun, bounds)
return res.x, res.fun

    # вычисление по введенным тикерам эмпирические корреляции приращения логарифмических цен на активы из списка тикеров (предполагается, что исторические данные уже сохранены)
def empcorrmatr(tickers):
num_tickers = {}
num = []
for ind, ticker in enumerate(tickers):
num_tickers.update({f'{ind}': ticker})
num.append(ind)
```

```
empcor = np.zeros([len(num), len(num)])
for i in range(len(num) - 1):
b_file = open(f"{num_tickers[f'{i}']}HD.pkl "rb")
data = pickle.load(b_file)
b_file.close()
path1 = data[num_tickers[f'{i}']]
for j in range(i + 1, len(num)):
b_file = open(f"{num_tickers[f'{j}']}HD.pkl "rb")
data = pickle.load(b_file)
path2 = data[num_tickers[f'{j}']]
b_file.close()
empcor[i, j] = EmpCorr(path1, path2)
empcor[j, i] = empcor[i, j]
for i in range(len(num)):
empcor[i, i] = 1
return empcor
```

```
    # оценка параметра mu по историческим данным в модели Блэка-Шоулза
# возвращает оценку mu, стоимость актива в начале интервала наблюдения, длину интервала наблюдения в годах
def par_mu_est(ticker):
b_file = open(f"{ticker}HD.pkl "rb")
data = pickle.load(b_file)
b_file.close()
v = np.array(data[ticker])
S0 = v[0]
return (np.log(v[-1]) - np.log(S0)) / len(v), S0 , data['Time range']
```

```
    # подготовительная фаза для калибровки корреляций
# в начале идет вычисление параметров в реальной мере
# потом вычисление матрицы эмпирических корреляций по тикерам
# возвращает откалиброванную матрицу корреляций (не обязательно по-
```

```
ложительно определенную) и матрицу эмпирических корреляций
def corr_matr_X(tickers, N):
mu = []
S_0 = []

    for ticker in tickers:
k = par_mu_est(ticker)
S_0.append(k[1])
mu.append(k[0])
T = k[2]

    Par = []
# параметры (1=kappa, 2=gamma, 3=vbar, 4=v0, 5=rho)
for ind, ticker in enumerate(tickers):
b_file = open(f"{ticker}.pkl "rb")
data = pickle.load(b_file) b_file.close()
Par.append(data['Parameters BS'])

    empcorr = empcorrmatr(tickers)
NoOfSteps = int(T * 252)
return corr_calibration(Par, S_0, N, NoOfSteps, T, mu, empcorr) , empcorr

    # калибровка матрицы корреляций
# на входе матрица исторических корреляций между активами
# на выходе откалиброванная матрица мгновенных корреляций

    def corr_calibration(par_model, par_S_0,
N, NoOfSteps, T, mu, rho_emp_matr):
# par_model - матрица, строка i которой состоит из откалиброванных параметров модели Хестона для актива i
# par_S_0 - вектор начальных стоимостей активов
# rho_emp_matr - матрица исторических корреляций
```

corr_Xi_Xj = np.zeros([len(par_S_0), len(par_S_0)])
eps = 10e-3
# вычисляем матрицу мгновенных корреляций между активами
for i in range(len(par_S_0)):
corr_Xi_Xj[i][i] = 1
for j in range(i + 1, len(par_S_0)):
# применение метода бисекций
corr_Xi_Xj[i][j] = Bis_opt(N, par_model[i], par_model[j], NoOfSteps, T,
np.array([mu[i], mu[j]]), par_S_0[i], par_S_0[j], rho_emp_matr[i][j], eps)
corr_Xi_Xj[j][i] = corr_Xi_Xj[i][j]
return corr_Xi_Xj

# по введенным тикерам возвращает параметры откалиброванных моделей и начальную стоимость в нужном формате, чтобы потом их отправлять на вход функции autocallbondprice
def par_data_final(tickers):
Par = []
S0 = []
for ticker in tickers:
b_file = open(f"{ticker}.pkl "rb")
data = pickle.load(b_file) b_file.close()
Par.append(data['Parameters BS'][0])
b_file = open(f"{ticker}HD.pkl "rb")
data = pickle.load(b_file) S0.append(np.array(data[ticker])[-1:])
b_file.close()
return Par, S0

Функции для модели Хестона.

Часть функций уже определена в предыдущем разделе. Здесь введем дополнительные.

# вычисление характеристической функции в модели Хестона
def ChFHestonModel(r,tau,par):

```
        kappa = par[0]
gamma = par[1]
vbar = par[2]
v0 = par[3]
rho = par[4]

        i = np.complex(0.0,1.0)
D1 = lambda u: np.sqrt(np.power(kappa-gamma*rho*i*u,2)
+(u*u+i*u)*gamma*gamma)
g = lambda u: (kappa-gamma*rho*i*u-D1(u))/(kappa-gamma*rho*i*u+D1(u))
C = lambda u: (1.0-np.exp(-D1(u)*tau))/(gamma *
gamma*(1.0-g(u)*D1(u)*tau))
*(kappa-gamma*rho*i*u-D1(u))
        # тут нет множителя -r*tau, так как дисконтирование происходит в функции для COS метода

        A = lambda u: r * i*u *tau + kappa*vbar*tau/gamma/gamma *(kappa-gamma*rho*i*u-D1(u))
- 2*kappa*vbar/gamma/gamma*np.log((1-g(u)*np.exp(-D1(u)*tau))/(1-g(u)))

        # характеристическая функция в модели Хестона

        cf = lambda u: np.exp(A(u) + C(u)*v0)
return cf

        # моделирование двухмерной коррелированной модели Хестона на основе моделирования методом Эйлера
# возвращает 2 путя
def Heston2path(par1, par2, NoOfSteps, T, mu, S_01, S_02, rho12):
# параметры (1=kappa, 2=gamma, 3=vbar, 4=v0, 5=rho)
# мгновенная кореляция между активами
Z1 = np.random.normal(0.0,1.0,[1, NoOfSteps])
Z2 = np.random.normal(0.0,1.0,[1, NoOfSteps])
```

```
Z3 = np.random.normal(0.0,1.0,[1, NoOfSteps])
Z4 = np.random.normal(0.0,1.0,[1, NoOfSteps])
W1 = np.zeros([1, NoOfSteps + 1])
W2 = np.zeros([1, NoOfSteps + 1])
W3 = np.zeros([1, NoOfSteps + 1])
W4 = np.zeros([1, NoOfSteps + 1])
V1 = np.zeros([1, NoOfSteps + 1])
V2 = np.zeros([1, NoOfSteps + 1])
X1 = np.zeros([1, NoOfSteps + 1])
X2 = np.zeros([1, NoOfSteps + 1])
V1[:,0] = par1[3]
V2[:,0] = par2[3]
X1[:,0] = np.log(S_01)
X2[:,0] = np.log(S_02)
time = np.zeros([1, NoOfSteps+1])
dt = T / float(NoOfSteps)
for i in range(0,NoOfSteps):

    Z2[:,i] = par1[4] * Z1[:,i] + np.sqrt(1.0-par1[4]**2) * Z2[:,i]
Z4[:,i] = par2[4] * rho12 * Z1[:,i] + par2[4] * np.sqrt(1.0-rho12**2) * Z3[:,i] +
np.sqrt(1.0-par2[4] ** 2) * Z4[:,i]
Z3[:,i] = rho12 * Z1[:,i] + np.sqrt(1.0-rho12**2) * Z3[:,i]

    W1[:,i+1] = W1[:,i] + np.power(dt, 0.5)*Z1[:,i]
W2[:,i+1] = W2[:,i] + np.power(dt, 0.5)*Z2[:,i]
W3[:,i+1] = W3[:,i] + np.power(dt, 0.5)*Z3[:,i]
W4[:,i+1] = W4[:,i] + np.power(dt, 0.5)*Z4[:,i]

    V1[:,i+1] = V1[:,i] + par1[0] * (par1[2] - V1[:,i]) * dt +
par1[1] * np.sqrt(V1[:,i]) * (W2[:,i+1]-W2[:,i])
V1[:,i+1] = np.maximum(V1[:,i+1],0.0)
X1[:,i+1] = X1[:,i] + (mu[0] - 0.5 * V1[:,i]) * dt + np.sqrt(V1[:,i]) * (W1[:,i+1]-
W1[:,i])
```

```
V2[:,i+1] = V2[:,i] + par2[0] * (par2[2] - V2[:,i]) * dt + par2[1] * np.sqrt(V2[:,i])
* (W4[:,i+1]-W4[:,i])
V2[:,i+1] = np.maximum(V2[:,i+1],0.0)
X2[:,i+1] = X2[:,i] + (mu[1] - 0.5 * V2[:,i]) * dt + np.sqrt(V2[:,i]) * (W3[:,i+1]-
W3[:,i])

    time[0][i+1] = time[0][i] + dt
return np.exp(X1), np.exp(X2), time

    # функция для подсчета ожидаемой выборочной корреляции в случае модели Хестона
def E_rho_emp(N, par1, par2, NoOfSteps, T, mu, S_01, S_02, rho12):
s = 0
for i in range(N):
a = Heston2path(par1, par2, NoOfSteps, T, mu, S_01, S_02, rho12)
s += EmpCorr(np.array(a[0][0]), np.array(a[1][0]))
return s / N

    # вычисление разложения Холецкого для общей корреляционной структуры
# на входе параметры моделей в виде двумерного листа и матрица корреляций между броуновскими движениями активов
def chol_matr(par_model, corr_matr):
C = np.zeros([len(par_model) * 2, len(par_model) * 2])
# конструируем общую корреялционную матрицу между броуновскими движениями в системе
for i in range(len(par_model)):
C[i, i] = 1
C[i, i + len(par_model)] = par_model[i][4]
C[i + len(par_model), i ] = C[i, i + len(par_model)]
for k1, i in enumerate(range(len(par_model), len(par_model) * 2)):
C[i, i] = 1
for k2, j in enumerate(range(len(par_model), len(par_model) * 2)):
```

```
        C[i, j] = corr_matr[k1][k2]
C[j, i] = C[i, j]

        w, v = np.linalg.eig(C)

        # проверяем положительную определенность, если не выполняется, то используем метод регуляризации Джэкеля
if check_pos_def(w) == True:

        return np.linalg.cholesky(C)
else:
Lambda = np.zeros([len(w), len(w)])
sp = np.zeros([len(w)])
for i in range(len(w)):
if w[i] >= 0:
Lambda[i][i] = w[i]
sp[i] = w[i]
T = np.zeros([len(w), len(w)])
for i in range(len(w)):
T[i][i] = 1/(np.power(v[i], 2).dot(sp))

        B = np.dot(np.dot(np.sqrt(T), v), np.sqrt(Lambda))
Cnew = np.dot(B, np.transpose(B))
w, v = np.linalg.eig(Cnew)
for i in range(len(par_model) *2):
Cnew[i, i] = 1
return chol(Cnew)

        # вычисление параметров для QE схемы
def CIRCDF(kappa, gamma, vbar, s, t, v_s):
delta = 4.0 *kappa*vbar/gamma/gamma
c= 1.0/(4.0*kappa)*gamma*gamma*(1.0-np.exp(-kappa*(t-s)))
```

```
kappaBar = 4.0*kappa*v_s*np.exp(-kappa*(t-s))/(gamma
*gamma*(1.0-np.exp(-kappa*(t-s))))
cdf =lambda x: st.ncx2.cdf(x/c,delta,kappaBar)
return cdf

        def CIRMean(kappa, gamma, vbar, s, t, v_s):
delta = 4.0 *kappa*vbar/gamma/gamma
c= 1.0/(4.0*kappa)*gamma*gamma*(1.0-np.exp(- kappa * (t-s)))
kappaBar = 4.0 * kappa * v_s * np.exp(-kappa * (t-s)) / (gamma
* gamma * (1.0 - np.exp(-kappa*(t-s))))
return c * (delta + kappaBar)

        def CIRVar(kappa, gamma, vbar, s, t, v_s):
delta = 4.0 *kappa*vbar/gamma/gamma
c= 1.0/(4.0*kappa)*gamma*gamma*(1.0-np.exp(-kappa*(t-s)))
kappaBar = 4.0*kappa*v_s*np.exp(-kappa*(t-s))/(gamma*gamma*(1.0-np.exp(-
kappa*(t-s))))
VarV = c*c*(2.0*delta+4.0*kappaBar)
return VarV

        # QE схема для моделирования v(t)|v(s)
def QE_scheme(kappa, gamma, vbar, s, t, v_s, NoOfSamples):
m = CIRMean(kappa, gamma, vbar, s, t, v_s)
s2 = CIRVar(kappa, gamma, vbar, s, t, v_s)
if m < 0 or s2 < 0:
print('параметры', kappa, gamma, vbar, s, t, v_s)
aStar = 1.5 # параметр разделения
if (s2/ (m * m) < aStar):

        # a и b - первый подход
b1 = 2.0 * m * m / s2 - 1.0 + np.sqrt(2.0 * m * m / s2)*np.sqrt((2.0 * m * m
/ s2) - 1.0)
b = np.sqrt(b1)
```

```
a = m / (1.0 + b1)
Z = np.random.normal(0.0, 1.0, [NoOfSamples, 1])

        if NoOfSamples > 1:
Z = (Z - np.mean(Z)) / np.std(Z)

        A = a * np.power(b + Z, 2.0)

        else:

        # c и d - второй подход
c = ((s2 / (m*m)) - 1.0) / ((s2 / (m*m)) + 1.0)
d = (1.0 - c) / m
U = np.random.uniform(0.0, 1.0, [NoOfSamples, 1])
A = 1.0 / d * np.log((1.0-c)/(1.0-U))
A[U < c] = 0.0

        return A

        # Моделирование многомерного пути в модели Хестона методом MQE

        def multi_asset_path(par_model, corr_matrix, parS0, T, r, NoOfSteps):
# параметры (1=kappa, 2=gamma, 3=vbar, 4=v0, 5=rho)
# par_model - двумерный лист параметров
# corr_matrix - откалиброванная мтариц корреляции между броуновскими движениями активов
# parS0 - лист начальных стоимостей активов
# T - дата экспирации
# r - процентная ставка
# NoOfSteps - количество шагов

        V = np.zeros([len(parS0), NoOfSteps+1])
X = np.zeros([len(parS0), NoOfSteps+1])
```

```
Vint = np.zeros([len(parS0), NoOfSteps+1])
f = np.zeros([len(parS0), 1])
s = np.zeros([len(parS0), 1])
L = chol_matr(par_model, corr_matrix)
rho_m = np.zeros([len(parS0), len(parS0)])

        for i in range(len(parS0)):
rho_m[i, i] = par_model[i][4]

        Ds = np.zeros([len(parS0), len(parS0)])
for i in range(len(parS0)):
V[i, 0] = par_model[i][3]
X[i, 0] = np.log(parS0[i])
L_star = np.zeros([len(parS0), len(parS0)])
for k1, i in enumerate(range(len(par_model), len(par_model) * 2)):
for k2, j in enumerate(range(len(par_model), len(par_model) * 2)):
L_star[k1, k2] = L[i, j]

        time = np.zeros([NoOfSteps+1])
dt = T / float(NoOfSteps)
s for i in range(0,NoOfSteps):
time[i+1] = time[i] + dt
Z = np.random.normal(0.0, 1.0, [len(parS0), 1])
for j in range(len(parS0)):

        V[j,i+1] = QE_scheme(par_model[j][0], par_model[j][1], par_model[j][2],
time[i], time[i+1], V[j, i], 1)[0][0]
Vint[j, i] = dt * (V[j,i+1] + V[j,i])/2
f[j] = (V[j,i+1] - V[j,i] - par_model[j][0] * par_model[j][2] * dt + par_model[j][0]
* Vint[j, i]) / par_model[j][1]
Ds[j, j] = np.sqrt(Vint[j, i])

        X[:, i+1] = X[:, i] + (r - Vint[:, i]/2) * dt + np.dot(rho_m , f).reshape([1,
```

```
len(parS0)])[0]
+ np.dot(Ds, np.dot(L_star, Z)).reshape([1, len(parS0)])[0] return np.exp(X), time

    # Возвращает оцененный параметр vbar, цену начала периода!!!! и длину интервала
def par_teta_est(ticker):
b_file = open(f"{ticker}HD.pkl "rb")
data = pickle.load(b_file)
b_file.close()
v = np.array(data[ticker])
S0 = v[0]
return np.sum(np.array([np.log(v[j + 1]/v[j]) for j in range(len(v) - 1)]) ** 2) / data['Time range'], S0 , data['Time range']
```

Функции для модели Бэйтса.

Часть функций как и в предыдущих разделах. Остальные:

```
    # вычисление характеристической функции в модели Бейтса
def ChFBatesModel(r,tau,par):

    kappa = par[0]
gamma = par[1]
vbar = par[2]
v0 = par[3]
rho = par[4]
xip = par[5]
muj = par[6]
sigmaj = par[7]

    i = np.complex(0.0,1.0)
D1 = lambda u: np.sqrt(np.power(kappa-gamma*rho*i*u,2)
+(u*u+i*u)*gamma*gamma)
g = lambda u: (kappa-gamma*rho*i*u-D1(u))/
```

```
(kappa-gamma*rho*i*u+D1(u))
C = lambda u: (1.0-np.exp(-D1(u)*tau))/
(gamma*gamma*(1.0-g(u)*D1(u)*tau))
*(kappa-gamma*rho*i*u-D1(u))

    # тут нет множителя -r*tau, так как дисконтирование происходит в функции для COS метода

    Aheston = lambda u: r * i*u *tau + kappa*vbar*tau/gamma/gamma
*(kappa-gamma*rho*i*u-D1(u))
- 2*kappa*vbar/gamma/gamma*np.log((1-g(u)*np.exp(-D1(u)*tau))/(1-g(u)))

    A = lambda u: Aheston(u) - xip * i * u * tau *(np.exp(muj+
0.5*sigmaj*sigmaj) - 1.0) +
xip * tau * (np.exp(i*u*muj - 0.5 * sigmaj * sigmaj * u * u) - 1.0)

    cf = lambda u: np.exp(A(u) + C(u)*v0)
return cf

    # Моделирование многомерного пути активов в модели Бейтса без модификации
def multi_asset_path(par_model, corr_matrix, parS0, T, r, NoOfSteps):
# параметры (1=kappa, 2=gamma, 3=vbar, 4=v0, 5=rho, 6=xip, 7=muj, 8=sigmaj)
dt = T / float(NoOfSteps)
V = np.zeros([len(parS0), NoOfSteps+1])
X = np.zeros([len(parS0), NoOfSteps+1])
xiEeJ = []
for i in range(len(parS0)):
xiEeJ.append(par_model[i][5] * (np.exp(par_model[i][6]
+ 0.5*par_model[i][7]*par_model[i][7]) - 1))

    Pois = np.array([np.random.poisson(par_model[i][5]*dt,[1,NoOfSteps +
```

```
1])[0] for i in range(len(parS0))])
J = np.array([np.random.normal(par_model[i][6], par_model[i][7],[1 ,NoOfSteps
+ 1])[0] for i in range(len(parS0))])
xiEeJ = np.array(xiEeJ)
Vint = np.zeros([len(parS0), NoOfSteps+1])
f = np.zeros([len(parS0), 1])
s = np.zeros([len(parS0), 1])
L = chol_matr(par_model, corr_matrix)

        rho_m = np.zeros([len(parS0), len(parS0)])

        for i in range(len(parS0)):
rho_m[i, i] = par_model[i][4]

        Ds = np.zeros([len(parS0), len(parS0)])
for i in range(len(parS0)):
V[i, 0] = par_model[i][3]
X[i, 0] = np.log(parS0[i])
L_star = np.zeros([len(parS0), len(parS0)])
for k1, i in enumerate(range(len(par_model), len(par_model) * 2)):
for k2, j in enumerate(range(len(par_model), len(par_model) * 2)):
L_star[k1, k2] = L[i, j]
time = np.zeros([NoOfSteps+1])

        for i in range(0,NoOfSteps):
time[i+1] = time[i] + dt
Z = np.random.normal(0.0, 1.0, [len(parS0), 1])
for j in range(len(parS0)):

        V[j,i+1] = QE_scheme(par_model[j][0], par_model[j][1], par_model[j][2],
time[i], time[i+1], V[j, i], 1)[0][0]
Vint[j, i] = dt * (V[j,i+1] + V[j,i])/2
f[j] = (V[j,i+1] - V[j,i] - par_model[j][0] * par_model[j][2] * dt + par_model[j][0]
```

```
* Vint[j, i]) / par_model[j][1]
Ds[j, j] = np.sqrt(Vint[j, i])
        X[:, i+1] = X[:, i] + (r - Vint[:, i]/2 - xiEeJ) * dt + np.dot(rho_m ,
f).reshape([1, len(parS0)])[0] +
np.dot(Ds, np.dot(L_star, Z)).reshape([1, len(parS0)])[0] + J[:, i] * Pois[:, i]
return np.exp(X), time


        # вычисление интенсивностей по введенным тикерам, и границам
промежутка
# на выходе матрица совместных прыжков за указанный промежуток
!pip install yfinance
import yfinance as yf
def intensities(tickers, t1, t2):
data = []
for i in tickers:
p1 = datetime.datetime(t1[2], t1[1], t1[0], 23, 59)
p2 = datetime.datetime(t2[2], t2[1], t2[0], 23, 59)
period1 = int(time.mktime(p1.timetuple()))
period2 = int(time.mktime(p2.timetuple()))
df = yf.download(i, p1 , p2)
data.append(np.array(df[['High', 'Low']]))


        jum = np.zeros([len(data), len(data)])


        for i in range(len(data[0]) - 1):
for j in range(len(data) - 1):
for k in range(j + 1, len(data)):
if data[j][i, 0] < data[j][i+1, 1] and data[k][i, 0] < data[k][i+1, 1]:
jum[j][k] += 1
elif data[j][i, 1] > data[j][i+1, 0] and data[k][i, 1] > data[k][i+1, 0]:
jum[j][k] += 1


        if data[j][i, 0] < data[j][i+1, 1] or data[j][i, 1] > data[j][i+1, 0]:
```

```
jum[j][j] += 1
if data[len(data) - 1][i, 0] < data[len(data) - 1][i+1, 1] or data[len(data) - 1][i,
1] > data[len(data) - 1][i+1, 0]:
jum[len(data) - 1][len(data) - 1] += 1

    voc = {}
for l, m in enumerate(tickers):
voc.update({m : jum[l]})

    b_file = open("Jumps.pkl "wb")
pickle.dump(voc, b_file)
b_file.close()
return jum

    # функция калибровки параметра theta в реальной мере в модели Бейтса
# на входе параметры модели Бейтса в риск-нейтральной мере, y - лист дневного приращения логарифмических цен на актив
# к сожалению после попыток оптимизации кода значения функции вычисляются очень долго
def RWCBates(par, y, r):

    def fun(x):
kappa = par[0]
omega = par[1]
rho = par[4]
lamb = par[5]
alpha = par[6]
beta = par[7]
eta = x
mu = r + kappa * (eta - par[2])/(rho * omega)
tau = 1/252
i = np.complex(0.0,1.0)
```

```
a0 = 2 * kappa * eta / (omega * omega)
b0 = (omega * omega) / (2 * kappa)
def kG(u):
return kappa - i * u * rho * omega
def psiX(u):
return lamb * (np.exp(i * beta * u - (1/2) * np.power(alpha, 2) * np.power(u,
2)) - 1) -
i * u * lamb * (np.exp(beta + (1/2) * np.power(alpha, 2)) - 1)
def psi(u):
return -0.5 * (u * u + i * u)
def gamma(u):
return np.sqrt(np.power(kG(u), 2) - 2 * (omega * omega) * psi(u))
def A(u):
return (kG(u) - gamma(u))/(kG(u) + gamma(u))
def C(u):
return i * u * mu * tau + (kappa * eta / (omega * omega)) * (kG(u) - gamma(u))
* tau -
(2 * kappa * eta / (omega * omega)) * np.log((A(u) * np.exp(- gamma(u) *
tau ) - 1)/(A(u) - 1)) + psiX(u) * tau
def D(u):
return ((kG(u) - gamma(u))/ (omega * omega)) * ((1-np.exp(-gamma(u) *
tau))/(1 -
A(u) * np.exp(-gamma(u) * tau)))
def F0(u, a, b):
return np.exp(C(u)) / np.power(1 - b * D(u), a)
def f(u, a, b):
return C(u) - a * np.log(1 - b * D(u))
def H(u):
return (kG(u) + gamma(u)) / (1 - A(u) * np.exp(-gamma(u) * tau))
def K(u):
return (omega * omega) * (1 - np.exp(-gamma(u) * tau)) / H(u)
def Cv(u):
return ((2 * kappa * eta / (omega * omega)) * K(u) * i)
```

```
def Cvv(u):
return -1 * (2 * kappa * eta / (omega * omega)) * np.power(K(u), 2)
def R(u):
return 4 * np.exp(-gamma(u) * tau) * np.power(gamma(u)/H(u) , 2)
def Dv(u):
return R(u) * i
def Dvv(u):
return -2 * K(u) * R(u)
def fv(u, a, b):
return Cv(u) + (a * b * Dv(u))/(1 - b * D(u))
def fvv(u, a, b):
return Cvv(u) + (a * b * Dv(u) + a * b * b * (np.power(Dv(u), 2) - D(u) *
Dvv(u))) / np.power( 1 - b * D(u), 2)
def F(u,a, b):
return np.exp(f(u, a, b))
def Fv(u, a, b):
return fv(u, a, b) * F(u, a, b)
def Fvv(u,a, b):
return (fvv(u, a, b) + np.power(fv(u, a, b), 2)) * F(u, a, b)

    def find_min(x1, x2):
for e1, e2 in enumerate(x1):
if e2 > x2:
return e1 - 3, e1 + 3

    h = []
N = np.power(2, 12)
du = 0.25
dx = 8 * np.pi / N
b = np.pi * 4
x = np.array([ -b + (t - 1) * dx for t in range(1 , N + 1)])
w = np.array([1.0 for t in range(N)])
w[0] = 1/2
```

```
w[N-1] = 1/2

        u = [(j -1 )* du for j in range(1, N+1)]

        py = []

        for n in tqdm(range(len(y) - 1)):
h0 = []
h1 = []
h2 = []
h3 = []
bou = find_min(x, y[n + 1])

        if n == 0:

        F0s = [np.exp(i* b * u[j -1] ) * F0(u[j -1], a0, b0) * w[j -1] * du for j in
range(1, N +1)]

        for k in range(bou[0], bou[1]):

        h0.append(np.real(np.sum([np.exp(-i * 2 * np.pi * (k - 1) * (j - 1)/N ) *
F0s[j -1] for j in range(1, N +1)])))

        h0 = np.array(h0)

        S0 = CubicSpline(x[bou[0] : bou[1]], h0)
py.append(S0(y[n + 1]))
else:

        Fs = [np.exp(i* b * u[j -1] ) * F(u[j -1], a0, b0) * w[j -1] * du for j in
range(1, N +1)]
fvs = [fv(u[j -1], a0, b0) for j in range(1, N +1)]
```

```
        Fvs = [fvs[j-1] * Fs[j -1] for j in range(1, N +1)]
Fvvs = [(fvv(u[j -1], a0, b0) + np.power(fvs[j-1], 2)) * Fs[j -1] for j in range(1, N +1)]

        for k in range(bou[0], bou[1]):

        h1.append(np.real(np.sum([np.exp(-i * 2 * np.pi * (k - 1) * (j - 1)/N ) * Fs[j-1] for j in range(1, N +1)]))/np.pi)
h2.append(np.imag(np.sum([np.exp(-i * 2 * np.pi * (k - 1) * (j - 1)/N ) * Fvs[j-1] for j in range(1, N +1)]))/np.pi)
h3.append(-np.real(np.sum([np.exp(-i * 2 * np.pi * (k - 1) * (j - 1)/N ) * Fvvs[j-1] for j in range(1, N +1)]))/np.pi)
h1 = np.array(h1)
S1 = CubicSpline(x[bou[0] : bou[1]], h1)
py.append(S1(y[n + 1]))
h2 = np.array(h2)
S2 = CubicSpline(x[bou[0] : bou[1]], h2)
m1 = S2(y[n + 1])/py[-1]
h3 = np.array(h3)
S3 = CubicSpline(x[bou[0] : bou[1]], h3)
m2 = S3(y[n + 1])/py[-1]
b0 = (m2 - np.power(m1, 2))/m1
a0 = m1 /b0

        return -np.sum(np.log(np.array(py)))

        res = optimize.differential_evolution(fun, bounds=[(0.01, 1)])
return res.x[0]
# оценка параметра theta в модели Бейтса в случае, если калибровка была бы быстрее
def par_teta_est2(ticker, r):
b_file = open(f"{ticker}HD.pkl "rb")
data = pickle.load(b_file)
```

```
b_file.close()
v = np.array(data[ticker])
y = np.array([np.log(v[j + 1]/v[j]) for j in range(len(v) - 1)])

    S0 = v[0]
b_file = open(f"{ticker}.pkl "rb")
data = pickle.load(b_file)
b_file.close() print(data['Parameters Bates'])
teta = RWCBates(data['Parameters Bates'] , y, r)
return teta, S0 , data['Time range']

    # генерация путей двухмерной модели Бейтса с модификацией введенной в работе
# на вход дополнительно подаются оценки вклада сектора в интенсивность прыжков
# !!!! на входе подаются уже отсортированные параметры
def Bates2pathmod(par1, par2, NoOfSteps, T, mu, S_01, S_02, rho12, intens):
# параметры (1=kappa, 2=gamma, 3=vbar, 4=v0, 5=rho, 6=xip, 7=muj, 8=sigmaj)

    dt = T / float(NoOfSteps)
Z1 = np.random.normal(0.0,1.0,[1, NoOfSteps])
Z2 = np.random.normal(0.0,1.0,[1, NoOfSteps])
Z3 = np.random.normal(0.0,1.0,[1, NoOfSteps])
Z4 = np.random.normal(0.0,1.0,[1, NoOfSteps])
xi_star = []
xi_bar = []
xi_star.append(par1[5] * intens[0])
xi_bar.append(par1[5] - xi_star[0])
for i in range(1, 1):
xi_star.append(par2[5] * intens[i] - sum(xi_star))
xi_bar.append(par2[5] - sum(xi_star))
```

```
        xi_bar.append(par2[5] - sum(xi_star))

        Pois1 = np.array([np.random.poisson(xi_star[i] *dt,[1,NoOfSteps + 1])[0]
for i in range(1)])
Pois2 = np.array([np.random.poisson(xi_bar[i]*dt,[1,NoOfSteps + 1])[0] for i
in range(2)])
Pois = np.array([Pois2[i] for i in range(2)])
for i in range(2):
for j in range(i + 1):
if j == 1:
break
else:
Pois[i] += Pois1[j]

        Cj = np.zeros([2, 2])
for i in range(1):
for j in range(i + 1, 2):
Cj[i, j] = min(intens[i], intens[j])
Cj[j, i] = Cj[i, j]
Cj[i, i] = 1

        Cj[1, 1] = 1
Cj = Janckel_reg(Cj)
Cj[0, 1] = Cj[0, 1] * par1[7]* par2[7]
Cj[1, 0] = Cj[0, 1]
Cj[0, 0] = par1[7]* par1[7]
Cj[1, 1] = par2[7] * par2[7]

        Lj = chol(Cj)

        mu1 = par1[6]
mu2 = par2[6]
# Моделирование пуассоновских процессов и их амплитуд
```

```
        W1 = np.zeros([1, NoOfSteps + 1])
W2 = np.zeros([1, NoOfSteps + 1])
W3 = np.zeros([1, NoOfSteps + 1])
W4 = np.zeros([1, NoOfSteps + 1])
V1 = np.zeros([1, NoOfSteps + 1])
V2 = np.zeros([1, NoOfSteps + 1])
X1 = np.zeros([1, NoOfSteps + 1])
X2 = np.zeros([1, NoOfSteps + 1])
V1[:,0] = par1[3]
V2[:,0] = par2[3]
X1[:,0] = np.log(S_01)
X2[:,0] = np.log(S_02)
time = np.zeros([1, NoOfSteps+1])
EeJ1 = np.exp(par1[6] + 0.5*par1[7]*par1[7])
EeJ2 = np.exp(par2[6] + 0.5*par2[7]*par2[7])

        for i in range(0,NoOfSteps):

        Z2[:,i] = par1[4] * Z1[:,i] + np.sqrt(1.0-par1[4]**2) * Z2[:,i]
Z4[:,i] = par2[4] * rho12 * Z1[:,i] + par2[4] * np.sqrt(1.0-rho12**2) * Z3[:,i] +
np.sqrt(1.0-par2[4] ** 2) * Z4[:,i]
Z3[:,i] = rho12 * Z1[:,i] + np.sqrt(1.0-rho12**2) * Z3[:,i]

        W1[:,i+1] = W1[:,i] + np.power(dt, 0.5)*Z1[:,i]
W2[:,i+1] = W2[:,i] + np.power(dt, 0.5)*Z2[:,i]
W3[:,i+1] = W3[:,i] + np.power(dt, 0.5)*Z3[:,i]
W4[:,i+1] = W4[:,i] + np.power(dt, 0.5)*Z4[:,i]
Zj1 = np.random.normal(0.0, 1.0, [1, 1])
Zj2 = np.random.normal(0.0, 1.0, [1, 1])
V1[:,i+1] = V1[:,i] + par1[0] * (par1[2] - V1[:,i]) * dt + par1[1] * np.sqrt(V1[:,i])
* (W2[:,i+1]-W2[:,i])
V1[:,i+1] = np.maximum(V1[:,i+1],0.0)
```

```
X1[:,i+1] = X1[:,i] + (mu[0] - 0.5 * V1[:,i] - par1[5]*(EeJ1-1)) * dt +
np.sqrt(V1[:,i]) * (W1[:,i+1]-W1[:,i]) + (mu1 + Zj1 * Lj[0][0]) * Pois[0][i]
V2[:,i+1] = V2[:,i] + par2[0] * (par2[2] - V2[:,i]) * dt + par2[1] * np.sqrt(V2[:,i])
* (W4[:,i+1]-W4[:,i])
V2[:,i+1] = np.maximum(V2[:,i+1],0.0)
X2[:,i+1] = X2[:,i] + (mu[1] - 0.5 * V2[:,i] - par2[5]*(EeJ2-1)) * dt +
np.sqrt(V2[:,i]) * (W3[:,i+1]-W3[:,i]) + (mu2 + Zj1 * Lj[1][0] + Zj2 * Lj[1][1])
* Pois[1][i]

    time[0][i+1] = time[0][i] + dt
return np.exp(X1), np.exp(X2), time

    def corrBatescalibration(N, par1, par2, NoOfSteps, T, mu, S_01, S_02,
rho_emp, bounds, intens):
def err_fun(x):
return (E_rho_emp(N, par1, par2, NoOfSteps, T, mu, S_01, S_02, x, intens)
- rho_emp) ** 2

    x = np.linspace(rho_emp - 0.1, 1, 60)
k = 0
zf = 10
# с учетом того, что данные по эмпирическим корреляциям есть, то было
решено взять максимально простой метод поиск минимума (по сути пере-
бор) и не брать метод дифференциальной эволюции,
#так как важно получить на выходе положительньо определенную матрицу
и не потратить много времени
# решением задачи минимизации оказывается первое число, на котором
ошибка оказывается меньше 0.001,
# то есть расхождение ожидаемой корреляции от эмпирической не больше
0,03
for j, i in enumerate(x):
y = err_fun(x[j])
if y <= 0.001:
```

```
k = j
zf = y
break
if zf > y:
k = j
zf = y
return x[k]
```

```
    # функция для подсчета ожидаемой выборочной корреляции в модели Бейтса
# генерация путей происходит с использование модификации
def E_rho_emp(N, par1, par2, NoOfSteps, T, mu, S_01, S_02, rho12, intens):
s = 0
for i in range(N): a = Bates2pathmod(par1, par2, NoOfSteps, T, mu, S_01, S_02, rho12, intens)
s += EmpCorr(a[0][0], a[1][0])
return s / N
```

```
    # сортировка тикеров
# На выходе лист отсортированных тикером и соответствующий лист вкладов сектора в интенсивность прыжков
def sort_tickers(tickers):
voc = {}
b_file = open("Intensities.pkl "rb")
jum = pickle.load(b_file)
b_file.close()
for i in tickers:
b_file = open(f"{i}.pkl "rb")
dt = pickle.load(b_file)
b_file.close()
voc.update({i: dt['Parameters Bates'][5] * jum[i]}) sorted_values = sorted(voc.values()
final_int = []
sorted_tickers = []
```

```
for i in sorted_values:
for k in voc.keys():
if voc[k] == i:
sorted_tickers.append(k)
final_int.append(jum[k])
break

    return sorted_tickers, final_int
```

Функция для моделирования путей многомерной модели Бейтса с модификацией.

```
    # Моделирование многомерного пути модели Бейтса с использованием модификации, введенной в работе
# параметры на входет уже отсортированы
def multi_asset_path(par_model, corr_matrix, parS0, T, r, NoOfSteps, intens):
# параметры (1=kappa, 2=gamma, 3=vbar, 4=v0, 5=rho, 6=xip, 7=muj, 8=sigmaj)
dt = T / float(NoOfSteps)
V = np.zeros([len(parS0), NoOfSteps+1])
X = np.zeros([len(parS0), NoOfSteps+1])
xiEeJ = []
for i in range(len(parS0)):
xiEeJ.append(par_model[i][5] * (np.exp(par_model[i][6]
+ 0.5*par_model[i][7]*par_model[i][7]) - 1))

    xi_star = []
xi_bar = []
xi_star.append(par_model[0][5] * intens[0])
xi_bar.append(par_model[0][5] - xi_star[0])
for i in range(1, len(parS0) - 1):
xi_star.append(par_model[i][5] * intens[i] - sum(xi_star))
xi_bar.append(par_model[i][5] - sum(xi_star))
```

```
        xi_bar.append(par_model[len(parS0) - 1][5] - sum(xi_star))

        Pois1 = np.array([np.random.poisson(xi_star[i] *dt,[1,NoOfSteps + 1])[0]
for i in range(len(parS0) - 1)])
Pois2 = np.array([np.random.poisson(xi_bar[i]*dt,[1,NoOfSteps + 1])[0] for i
in range(len(parS0))])
Pois = np.array([Pois2[i] for i in range(len(parS0))])
for i in range(len(parS0)):
for j in range(i + 1):
if j == len(parS0) - 1:
break
else:
Pois[i] += Pois1[j]

        Cj = np.zeros([len(parS0), len(parS0)])
for i in range(len(parS0) - 1):
for j in range(i + 1, len(parS0)):
Cj[i, j] = min(intens[i], intens[j])
Cj[j, i] = Cj[i, j]
Cj[i, i] = 1

        Cj[len(parS0) - 1, len(parS0) - 1] = 1
Cj = Janckel_reg(Cj)

        for i in range(len(parS0) - 1):
for j in range(i + 1, len(parS0)):
Cj[i, j] = Cj[i, j] * par_model[i][7]* par_model[j][7]
Cj[j, i] = Cj[i, j]
Cj[i, i] = par_model[i][7]* par_model[i][7]

        Cj[len(parS0) - 1, len(parS0) - 1] = par_model[len(parS0)
- 1][7] * par_model[len(parS0) - 1][7]
```

```
Lj = chol(Cj)

        muJ = np.array([par_model[i][6] for i in range(len(parS0))])

        J = np.array([np.random.normal(par_model[i][6],
par_model[i][7], [1 ,NoOfSteps +
1])[0] for i in range(len(parS0))])
xiEeJ = np.array(xiEeJ)
Vint = np.zeros([len(parS0), NoOfSteps+1])
f = np.zeros([len(parS0), 1])
s = np.zeros([len(parS0), 1])
L = chol_matr(par_model, corr_matrix)

        rho_m = np.zeros([len(parS0), len(parS0)])

        for i in range(len(parS0)):
rho_m[i, i] = par_model[i][4]

        Ds = np.zeros([len(parS0), len(parS0)])
for i in range(len(parS0)):
V[i, 0] = par_model[i][3]
X[i, 0] = np.log(parS0[i])
L_star = np.zeros([len(parS0), len(parS0)])
for k1, i in enumerate(range(len(par_model), len(par_model) * 2)):
for k2, j in enumerate(range(len(par_model), len(par_model) * 2)):
L_star[k1, k2] = L[i, j]
time = np.zeros([NoOfSteps+1])

        for i in range(0,NoOfSteps):
time[i+1] = time[i] + dt
Z = np.random.normal(0.0, 1.0, [len(parS0), 1])
Zj = np.random.normal(0.0, 1.0, [len(parS0), 1])
for j in range(len(parS0)):
```

V[j,i+1] = QE_scheme(par_model[j][0], par_model[j][1], par_model[j][2], time[i], time[i+1], V[j, i], 1)[0][0]
Vint[j, i] = dt * (V[j,i+1] + V[j,i])/2
f[j] = (V[j,i+1] - V[j,i] - par_model[j][0] * par_model[j][2] * dt + par_model[j][0] * Vint[j, i]) / par_model[j][1]
Ds[j, j] = np.sqrt(Vint[j, i])

X[:, i+1] = X[:, i] + (r - Vint[:, i]/2 - xiEeJ) * dt + np.dot(rho_m , f).reshape([1, len(parS0)])[0] +
np.dot(Ds, np.dot(L_star, Z)).reshape([1, len(parS0)])[0] + (muJ +np.dot(Lj, Zj).reshape([1, len(parS0)])[0]) * Pois[:, i]
return np.exp(X), time