Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Магістерська робота

освітній ступінь – магістр

на тему: **«Триноміальні моделі фінансового ринку: чутливість моделі і оцінювання американських опціонів»**

Виконала: студентка 2-го року навчання

освітньо-наукової програми «Системний аналіз»,

спеціальності 124 Системний аналіз

Паук Вікторія Михайлівна

Керівник: Щестюк Н. Ю.,

кандидат фіз.-мат. наук, доцент

Рецензент_____

(прізвище та ініціали)

Кваліфікаційна робота захищена з оцінкою _____

Секретар ЕК _____

«____» _____ 20____ р.

Київ – 2022

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри математики,

Доцент, д. ф.-м. н. Б.В. Олійник

_____

(підпис)

“____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на магістерську роботу

студентці Паук Вікторії Михайлівні

факультету інформатики МП 2 курсу

Тема: Триноміальні моделі фінансового ринку: чутливість моделі і оцінювання американських опціонів

1 Ідентифікація біноміальної моделі

2 Ідентифікація триноміальної моделі

3 Дослідження моделі на чутливість

4 Динамічне дельта хеджування

Дата видачі „____”_____ 2021 р.

Керівник _____

(підпис)

Завдання отримала _____

(підпис)

Календарний план виконання роботи:

№ п/п	Назва етапу магістерської роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на магістерську роботу.	04.10.2021	
2.	Огляд технічної літератури за темою роботи.	23.11.2021	
3.	Програмування практичної частини	17.04.2022	
4.	Написання пояснювальної роботи.	17.05.2022	
6.	Створення слайдів для доповіді та написання доповіді.	8.06.2022	
8.	Коригування роботи.	10.06.2022	
8.	Остаточне оформлення	15.06.2022	

	пояснювальної роботи та слайдів.		
10.	Здача магістерської роботи	30.06.2022	

Студент _____

Керівник _____

“_____”_____

Анотація

Робота присвячена застосуванню теорії триноміальної моделі на реальних даних, визначенню чутливості моделі та виконанню дельта хеджування.

В роботі було розглянено дані акцій для NVIDIA за період з 16 червня 2021 по 15 червня 2022. Отримані дані дозволили оцінити точність в оцінювані американських опціонів біноміальної та триноміальної моделі, порахувати чутливість за допомогою греків та виконати дельта хеджування для триноміальної моделі.

Зміст

ВСТУП

Протягом останніх десятиріч все більше і більше вчених залучаються до дослідження методів ціноутворення за допомогою дерев. Одна з найбільш відомих моделей - модель Блека-Шоулза - що має досить глибоке математичне підгрунтя. Наступна модель, яка вже стала базовою моделю фінансового ринку - модель Кокса-Роса-Рубінштейна або біноміальну модель опціонного ціноутворення. Дана модель була запропонована у 1979 році і суть її полягає в тому, що будується біноміальне дерево, яке описує рух ціни акцій з деякою ймовірністю, і на основі побудованого дерева знаходиться ціна опціону. Біноміальна модель набула популярність на фінансових ринкахв тому числі через простоту для розуміння, маючи нескладні математичні формули та економічну важливість. Проте відсутність можливості збереження ціни через певний інтервал часу може призвести до великих помилок в обрахунках складних варіантів.

Враховуючи недоліки біноміальної моделі у 1989 Фелім Бойль запропонував вдосконалити її додавши можливість відсутності руху ціни акції, окрім руху вгору і вниз [4]. Модель з трьома розвитками значення цін називається триноміальною.

Метою роботи є побудова триноміального дерева, знаходження справедливої ціни американського опціону та побудова дельта хеджування.

Робота складається з двох розділів.

В першому розділі ідентифіковано біноміальну та триноміальну модель та знайдено основні параметри, потрібні для побудови дерева, а також розглянуто алгоритм знаходження цін європейського та американського опціонів.

В другому розділі застосовано теорію триноміальних моделей на реальних даних, обраховано коефіцієнти дельта, гамма та виконано динамічне дельта хеджування.

1. Біноміальна та триноміальна моделі фінансового ринку

1.1 Біноміальна модель

Основна і найпоширеніша модель фінансового ринку є біноміальна. Для побудови даної моделі і визначення ціни опціону потрібно побудувати біноміальне дерево.

Біноміальне дерево - діаграма, що відображає можливі варіанти зміни цін протягом дії опціону. На кожному відрізку часу ціна акції може з зросати або спадати з деякою ймовірністю.

Біноміальне дерево задається наступними параметрами:

S_0- ціна акції в початковий момент,

u- коефіцієнт підвищення ціни,

d- коефіцієнт пониження ціни,

p- ймовірність підвищення стокової ціни,

r– безризикова ставка, яка постійно наростає,

σ – волатильність,

T – термін погашення опціону,

N — загальна кількість кроків дерева.

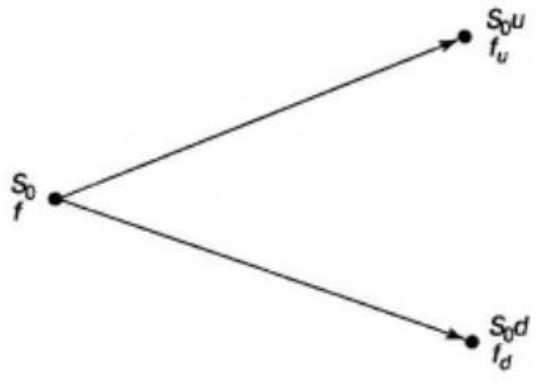

Рисунок 1.1

За умовою симетричності дерева $\mathrm{u} * \mathrm{d} = 1$.

Щоб дерево було нейтральним для ризику, середнє значення та дисперсія на кожному етапі часу мають бути асимптотично правильними [6].

Математичне очікування ціни акції:

$$E(S_T) = e^{r\Delta t} S_o \tag{1.1.1}$$

Щоб співставити очікувану ціну акції та ціну акції визначену за допомогою біноміального дерева запишемо рівняння:

$$S_0 up + S_0 d(1-p) = S_0 e^{r\Delta t} \tag{1.1.2}$$

$$up + d(1-p) = e^{r\Delta t} \tag{1.1.3}$$

Отже, ймовірність, з якою ціна акції може підвищитись задається наступною формулою:

$$\mathrm{p} = \frac{\mathrm{e}^{\mathrm{r}\Delta\mathrm{t}} - \mathrm{d}}{\mathrm{u} - \mathrm{d}} \tag{1.1.4}$$

Щоб співставити волатильність цін акцій з параметрами дерева запишемо рівняння:

$$\mathrm{pu}^2 + (1-\mathrm{p})\mathrm{d}^2 - \mathrm{e}^{2\mathrm{r}\Delta\mathrm{t}} = \sigma^2 \Delta\mathrm{t} \tag{1.1.5}$$

Підставимо формулу ймовірності (1.1.4) в (1.1.5):

$$\frac{\mathrm{e}^{\mathrm{r}\Delta\mathrm{t}} - \mathrm{d}}{\mathrm{u} - \mathrm{d}} u^2 + d^2 - \frac{\mathrm{e}^{\mathrm{r}\Delta\mathrm{t}} - \mathrm{d}}{\mathrm{u} - \mathrm{d}} \mathrm{d}^2 - e^{2r\Delta t} = \sigma^2 \Delta\mathrm{t} \tag{1.1.6}$$

$$(\mathrm{e}^{\mathrm{r}\Delta\mathrm{t}} - \mathrm{d})(\mathrm{u} + \mathrm{d}) + \mathrm{d}^2 - e^{2r\Delta t} = \sigma^2 \Delta\mathrm{t} \tag{1.1.7}$$

$$\mathrm{e}^{\mathrm{r}\Delta\mathrm{t}}(\mathrm{u} + \mathrm{d}) - \mathrm{ud} - e^{2r\Delta t} = \sigma^2 \Delta\mathrm{t} \tag{1.1.8}$$

$$\mathrm{e}^{\mathrm{r}\Delta\mathrm{t}}(\mathrm{u} + \mathrm{d}) = 1 + \sigma^2 \Delta\mathrm{t} + e^{2r\Delta t} \tag{1.1.9}$$

Застосуємо розклад в ряд Тейлора для $\mathrm{e}^{\sigma\sqrt{\Delta\mathrm{t}}}$ [7].

$$\mathrm{e}^{\sigma\sqrt{\Delta\mathrm{t}}} = 1 + \sigma\sqrt{\Delta\mathrm{t}} + \frac{\sigma^2}{2}\Delta\mathrm{t} + \ldots \tag{1.1.10}$$

Скориставшись формулою (1.1.9) та розкладом (1.1.10) отримаємо значення для параметрів u,d:

$$u = e^{\sigma\sqrt{\Delta t}} \quad (1.1.11)$$

$$d = e^{-\sigma\sqrt{\Delta t}} \quad (1.1.12)$$

Опишемо ітераційний метод обчислення справедливої ціни опціону для біноміальної моделі фінансового ринку:

1. Побудова біноміального дерева цін акцій
2. Знаходження значення платіжної функції для останнього часового проміжку, використовуючи дерево побудоване раніше та наступні функції:

   max { (S − K) , 0 } - для call опціону

   max { (K − S) , 0 }- для put опціону
3. Європейський опціон може бути виконаний лише на дату закінчення терміну дії опціону, тобто в один заздалегідь визначений момент часу. Для знаходження справедливої ціни будується платіжне дерево на основі знайденої функції за формулою (1.1.13). Значення ціни опціону відповідає значенню в вершині дерева на першому часовому проміжку.

$$f = max(e^{-r\Delta t}(pf_u + (1-p)f_d)) \quad (1.1.13)$$

Американський опціон може бути виконаний в будь-який час до закінчення терміну дії і обчислюється за формулою (1.1.14).

$$f = max(payoff, e^{-r\Delta t}(pf_u + (1-p)f_d)) \quad (1.1.14)$$

1.2 Визначення триноміальної моделі

Триноміальна модель є модифікацією біноміальної моделі запропонованою ірландським економістом Фелімом Бойлем у 1989 році.

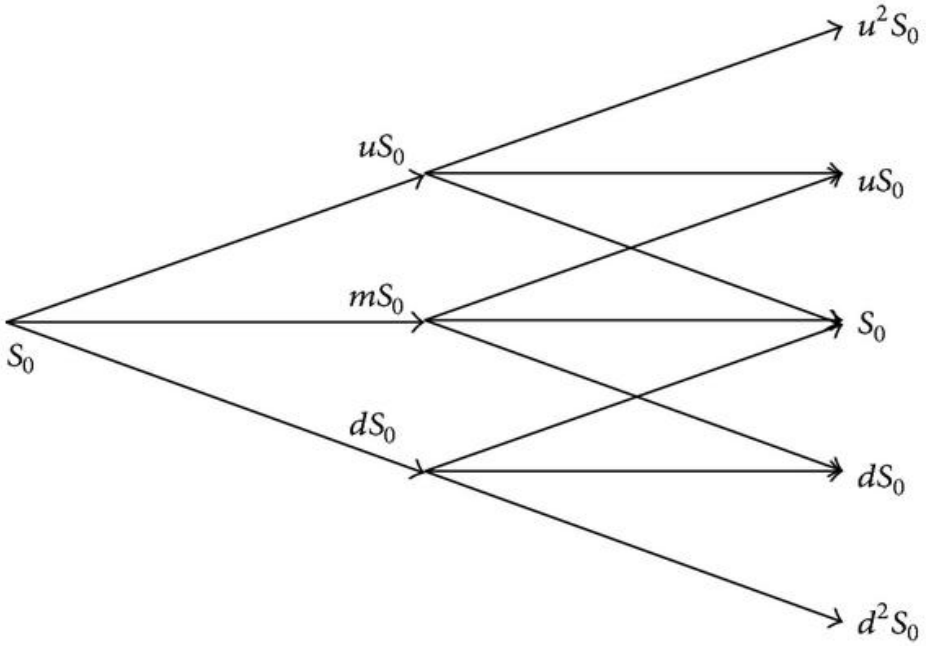

Рисунок 1.2.1

В кожен визначений момент часу ціна може підвищитись, понизитись чи залишитись такою ж. Відповідно, маємо наступні параметри:

u- коефіцієнт підвищення ціни

m- коефіцієнт середньої ціни

d-коефіцієнт пониження ціни

p_u- ймовірність підвищення стокової ціни

p_d- ймовірність пониження стокової ціни

Протягом останніх 20 років було запропоновано декілька модифікацій для визначення параметрів побудови моделі, але ми розглянемо класичне визначення Феліма Бойля.

Застосувавши припущення, що $m = 1$ зменшуємо кількість параметрів до чотирьох. Значення u та d за припущенням Бойля

дорівнюють значенням для біноміального дерева, але з деяким коефіцієнтом λ , що має бути більше одиниці [5]:

$$\mathrm{u} = \mathrm{e}^{\lambda\,\sigma\sqrt{\Delta \mathrm{t}}} \tag{1.2.1}$$

$$\mathrm{d} = \mathrm{e}^{-\lambda\,\sigma\sqrt{\Delta \mathrm{t}}} \tag{1.2.2}$$

$$\mathrm{m} = 1 \tag{1.2.3}$$

Найкращі результати були отримані при $\lambda \approx 1.2$ [3].

Очікуване значення дохідності та варіації визначене наступним чином:

$$E(S_T) = e^{r\Delta t} S_o \tag{1.2.4}$$

$$Var(S_T) = {S_0}^2 \mathrm{e}^{\Delta t \sigma^2} \tag{1.2.5}$$

В ризик-нейтральний світі очікувана дохідність всіх активів дорівнює безризиковій відсотковій ставці (це означає, що до всіх очікуваних виграшів застосовується ставка дисконту) [2].

З іншого боку, запишемо очікувану вартість та варіацію для дискретного випадку:

$$E(S_T) = p_u S_0 u + p_m S_0 + p_d S_0 d \tag{1.2.6}$$

$$Var(S_T) = {S_0}^2 u^2 p_u + {S_0}^2 p_m + {S_0}^2 d^2 p_d - (S_o e^{r\Delta t})^2 \tag{1.2.7}$$

Прирівняємо два значення для математичного очікування та варіації утворимо два рівняння системи. Також використовуючи властивість, що ймовірності протилежних подій в сумі дорівнюють одиниці, запишемо третє рівняння. Отже, отримали систему з трьох рівнянь і трьох невідомих:

$$\begin{cases} p_u + p_m + p_d = 1 \\ S_0 p_u + S_0 p_m + S_0 p_d = S_0 e^{r\Delta t} \\ {S_0}^2 u^2 p_u + {S_0}^2 p_m + {S_0}^2 d^2 p_d - (S_0 e^{r\Delta t})^2 = S_0 e^{\sigma^2 \Delta t} \end{cases} \tag{1.2.8}$$

Поділивши праву і ліву частину рівнянь на S_0: $S_0 > 0$ отримуємо:

$$\begin{cases} p_u + p_m + p_d = 1 \\ p_u + p_m + p_d = e^{r\Delta t} \\ u^2 p_u + p_m + d^2 p_d - (e^{r\Delta t})^2 = e^{\sigma^2 \Delta t} \end{cases} \tag{1.2.9}$$

Виразимо p_m через p_d та p_u та знаходимо ці значення.

$$\begin{cases} p_m = 1 - p_u - p_d \\ p_u(u-1) + p_d(d-1) = e^{r\Delta t} - 1 \\ p_u(u^2-1) + p_d(d^2-1) = e^{2r\Delta t}e^{\sigma^2\Delta t} - 1 \end{cases} \quad (1.2.10)$$

$$\begin{cases} p_m = 1 - p_u - p_d \\ p_u(u-1) = e^{r\Delta t} - 1 - p_d(d-1) \\ (u+1)(e^{r\Delta t} - 1 - p_d d + p_d) + p_d(d^2-1) = e^{2r\Delta t}e^{\sigma^2\Delta t} - 1 \end{cases} \quad (1.2.11)$$

Отже, отримали наступні значення ймовірностей для триноміальної моделі:

$$\begin{cases} p_m = 1 - p_u - p_d, \\ p_u = \frac{e^{2r\Delta t}e^{\sigma^2\Delta t} - e^{2r\Delta t}(d+1) + d}{(u-d)(u-1)}, \\ p_d = \frac{e^{2r\Delta t}e^{\sigma^2\Delta t} - e^{2r\Delta t}(u+1) + u}{(d-u)(d-1)}. \end{cases} \quad (1.2.12)$$

Щоб визначити ціну опціону на основі триноміального дерева застосовується наступний алгоритм:

1. Побудова біноміального дерева цін акцій
2. Знаходження значення платіжної функції для останнього часового проміжку, використовуючи дерево побудоване раніше та наступні функції:

   max { (S − K) , 0 } - для call опціону

   max { (K − S) , 0 }- для put опціону
3. Для знаходження справедливої ціни європейського опціону будується платіжне дерево на основі знайденої функції за формулою (1.2.13). Значення ціни опціону відповідає значенню в вершині дерева на першому часовому проміжку.

$$f = e^{-r\Delta t}(p_u f_u + p_m f_m + p_d f_d) \quad (1.2.13)$$

Американський опціон може бути виконаний в будь-який час до закінчення терміну дії і обчислюється за формулою (1.1.14).

2. Практичне дослідження триноміальної моделі

2.1 Визначення ціни американського опціону

Для побудови раніше описаних моделей та їх порівняння було обрано дані NVIDIA за період з з 16 червня 2021 по 15 червня 2022. Візуалізацію даних за цей період зроблена на рисунку 3.1.

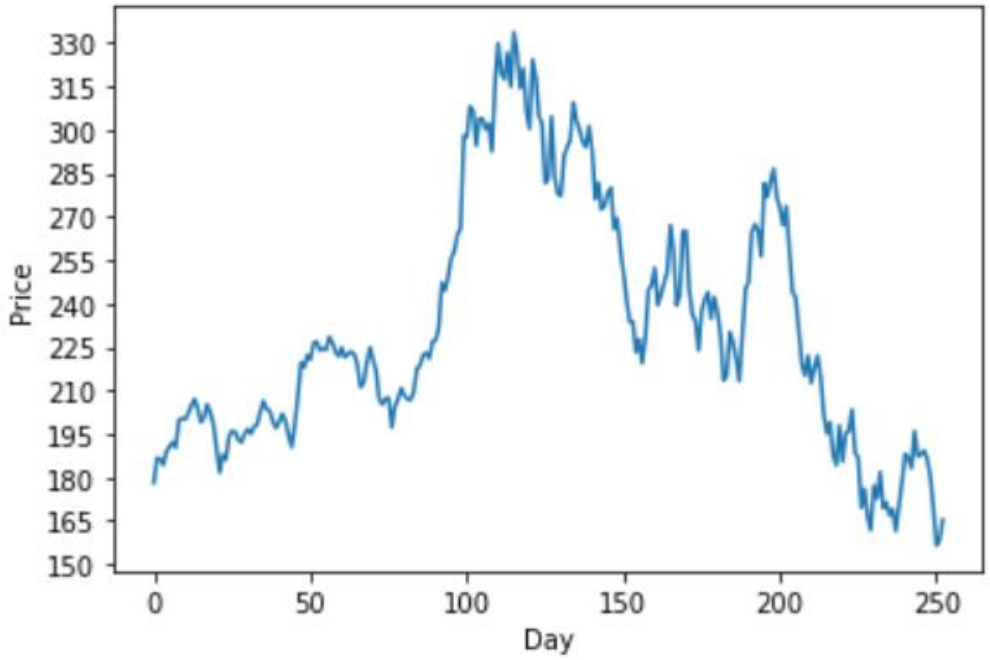

Рисунок 2.1.1

Для останньої ціни обраних даних побудуємо біноміальне дерево на 132 дні. Параметри побудови:

S = 165.27, K = 170, N = 5, $\sigma = 0.325779$, r = 0.052769.

					280.004244
				251.982568	226.765186
			226.765186	204.071456	183.648821
		204.071456	183.648821	165.27	148.730456
	183.648821	165.27	148.730456	133.846122	120.451351
165.27	148.730456	133.846122	120.451351	108.397073	97.549139

Для цих же параметрів побудуємо платіжне дерево та знайдемо вартість американського опціону put.

```
                                                          0.0
                                                0.0       0.0
                                      0.0       0.0       0.0
                           2.001432   4.024257  8.091528  16.269544
                  6.817466 11.707162  19.516796 31.153878  44.548649
14.164072  21.664775  31.858593  44.548649  56.602927  67.450861
```

Рисунок 2.1.2

Вартість опціону - 14.164.

Побудуємо триноміальне дерево для таких же параметрів, як для біноміального дерева:

```
                                                  280.004244
                                       251.982568 251.982568
                             226.765186 226.765186 226.765186
                   204.071456 204.071456 204.071456 204.071456
         183.648821 183.648821 183.648821 183.648821 183.648821
165.27       165.27     165.27     165.27     165.27     165.27
         148.730456 148.730456 148.730456 148.730456 148.730456
                   133.846122 133.846122 133.846122 133.846122
                              120.451351 120.451351 120.451351
                                         108.397073 108.397073
                                                     97.549139
```

Рисунок 2.1.3

Платіжне дерево для обрахунку американського put опціону:

```
                                                          0.0
                                                0.0       0.0
                                      0.0       0.0       0.0
                           2.084947   0.0       0.0       0.0
                  6.981596 4.028215   4.13544   0.0       0.0
14.362794  11.718301  11.870099  8.096205  8.202544  0.0
                  21.820163  19.527007  19.624014  16.269544  16.269544
                             31.932495  31.153878  31.153878  31.153878
                                        44.548649  44.548649  44.548649
                                                   56.602927  56.602927
                                                              67.450861
```

Рисунок 2.1.4

Параметри побудови:

S = 165.27, K = 165, N = 5, $\sigma = 0.325779$, r=0.052769.

Days to expiration date	Binomial call	Trinomial call	Real call option	Binomial put	Trinomial put	Real put option
2	1.932620	2.018657	4.150000	1.593532	1.681726	3.350000
7	3.494316	3.669989	2.920000	2.982633	3.171629	10.950000
12	4.559833	4.795040	5.060000	3.911879	4.145831	8.100000
32	7.521203	7.915548	13.760000	6.357282	6.700173	11.620000
72	11.559971	12.157478	19.720000	9.369492	9.835010	18.130000
132	16.100714	16.907633	28.110000	12.383632	12.948291	17.150000
252	23.145830	24.230220	53.950000	16.422157	17.083352	33.350000
504	34.466464	35.804885	53.980000	21.624010	22.343263	37.690000

Рисунок 2.1.5

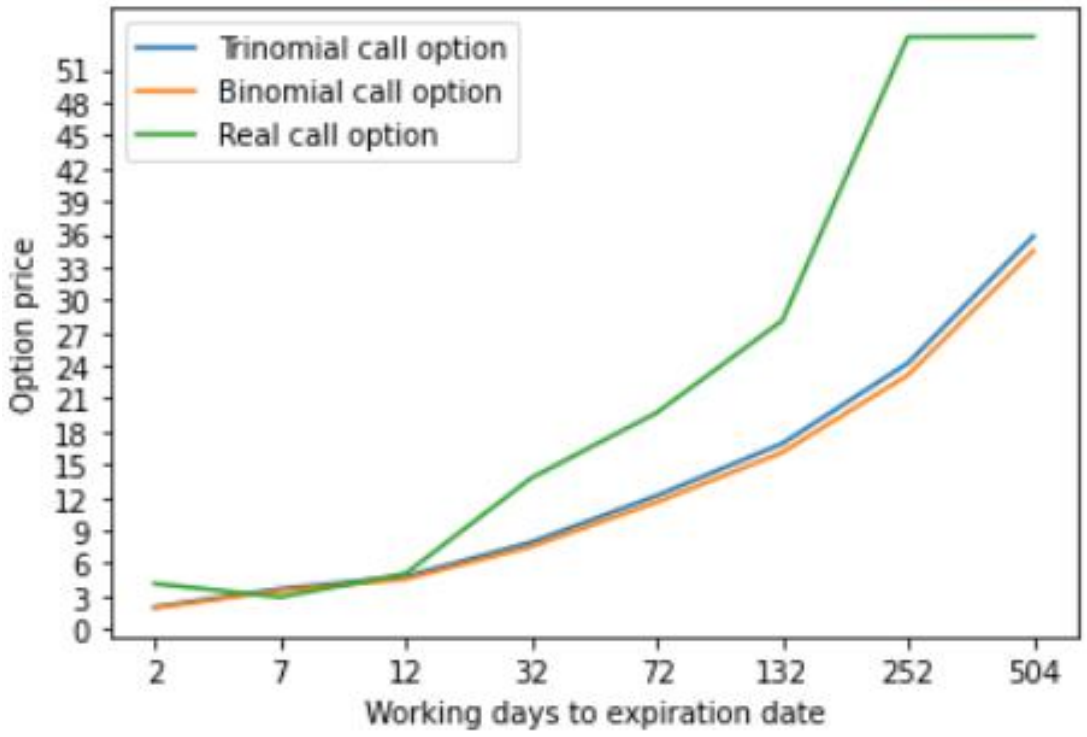

Рисунок 2.1.6

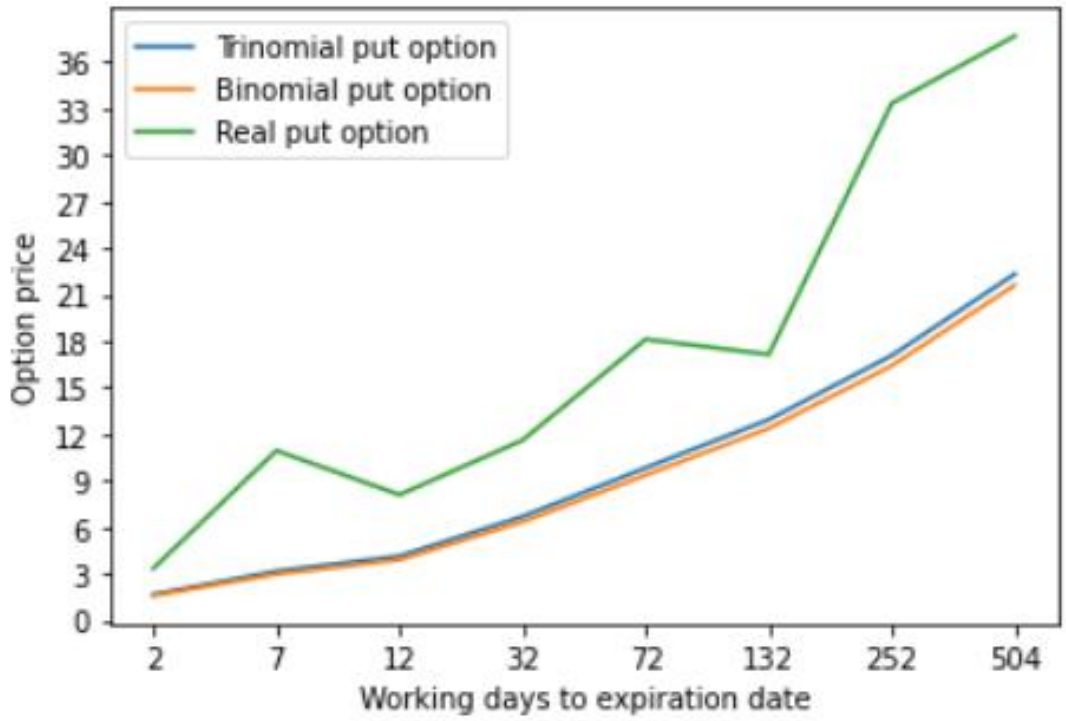

Рисунок 2.1.7

Проаналізувавши результати обчислень біноміальної і триноміальної моделей, а також порівнявши їх з реальними цінами опціонів можна припустити, що триноміальна модель дає значення ближче до справедливих, хоча обидві розглянуті моделі сильно занижають значення опціонів. Для оцінки результатів обрахуємо відносну похибку.

Error,binomial call	Error,binomial put	Error,trinomial call	Error,trinomial put
0.534308	0.524319	0.513577	0.497992
0.196683	0.727613	0.256846	0.710354
0.098847	0.517052	0.052364	0.488169
0.453401	0.452902	0.424742	0.423393
0.413795	0.483205	0.383495	0.457528
0.427225	0.277922	0.398519	0.244998
0.570976	0.507582	0.550876	0.487756
0.361496	0.426267	0.336701	0.407183

Рисунок 2.1.8

Отже, відносна похибка має різні значення в залежності від кількості днів до закінчення дії опціону. Причому, видно, що майже для всіх американських опціонів триноміальна модель похибка нижча.

Option type	RMSE
Binomial call	14.075259
Binomial put	9.720683
Trinomial call	12.617542
Trinomial put	8.753989

Рисунок 2.1.9

Значення обрахунку RMSE підтвердили припущеня, зроблені раніше, що триноміальна модель точніше оцінює значення опцінів.

2.2 Аналіз чутливості

Так звані «греки» — це різні параметри хеджування, які можна обчислити на основі ціни опціону. Греки використовуються для усунення ризику. Кожному греку дається різна назва залежно від того, яке похідне береться. Є багато греків, але ми розглянемо два основні - дельта і гамма.

Дельта представляє швидкість зміни нашої ціни опціону відносно нашого активу S.

$$\Delta = \frac{\partial C}{\partial S} \qquad (2.2.1)$$

Гама представляє швидкість зміни дельти відносно базового активу.

$$\gamma = \frac{\partial^2 C}{\partial S^2} \qquad (2.2.2)$$

Для обчислення потрібно дискретизувати часткову похідну, і ми отримаємо дельта. Для знаходження гамма використовується похідна першого порядку та другого порядку [1].

$$\frac{\partial C}{\partial S} = \frac{C(S(0)+\Delta S)-C(S(0))}{\Delta S} = \frac{C(S(0)+\Delta S)-C(S(0)-\Delta S)}{2\Delta S} \qquad (2.2.3)$$

$$\frac{\partial^2 C}{\partial S^2} = \frac{C(S(0)+\Delta S)-2C(S(0))+C(S(0)-\Delta S)}{\Delta S^2} \qquad (2.2.4)$$

Обчислимо показники дельта та гамма для триноміальної моделі та побудуємо дерева греків. Параметри побудови:

S = 165.27, K = 170, N = 5, $\sigma = 0.325779$, r = 0.052769.

Дерево дельта для американського put опціону:

```
                                                -0.009456
                                      -0.018858 -0.018858
                            -0.076019 -0.076019 -0.076019
                  -0.136026 -0.136026 -0.136026 -0.136026
          -0.274448 -0.274448 -0.274448 -0.274448 -0.274448
-0.427674 -0.427674 -0.427674 -0.427674 -0.427674 -0.427674
          -0.605261 -0.605261 -0.605261 -0.605261 -0.605261
                     -0.81304  -0.81304  -0.81304 -0.813040
                            -0.934792 -0.934792 -0.934792
                                  -1.0 -1.000000
                                       -1.000000
```

Рисунок 2.2.1

Дерево гамма для американського put опціону:

```
                                                  0.000675
                                       -0.000029 -0.000029
                             0.005057  0.005057  0.005057
                   0.000113  0.000113  0.000113  0.000113
         0.014771  0.014771  0.014771  0.014771  0.014771
0.0018     0.0018    0.0018    0.0018    0.0018  0.001800
         0.021095  0.021095  0.021095  0.021095  0.021095
                   0.007009  0.007009  0.007009  0.007009
                             0.010827  0.010827  0.010827
                                   0.0  0.000000
                                        0.000000
```

Рисунок 2.2.2

Дерева дельта і гамма для американського call опціону:

```
                                                  0.990544
                                        0.982125  0.982125
                              0.925164  0.925164  0.925164
                    0.871533  0.871533  0.871533  0.871533
          0.734461  0.734461  0.734461  0.734461  0.734461
0.598043  0.598043  0.598043  0.598043  0.598043  0.598043
          0.433329  0.433329  0.433329  0.433329  0.433329
                    0.260114  0.260114  0.260114  0.260114
                              0.161285  0.161285  0.161285
                                        0.051485  0.051485
                                                  0.027800

                                                   0.000675
                                       -0.000107 -0.000107
                              0.005147  0.005147  0.005147
                   -0.000653 -0.000653 -0.000653 -0.000653
          0.015677  0.015677  0.015677  0.015677  0.015677
-0.001491 -0.001491 -0.001491 -0.001491 -0.001491 -0.001491
          0.023861  0.023861  0.023861  0.023861  0.023861
                   -0.001513 -0.001513 -0.001513 -0.001513
                              0.018147  0.018147  0.018147
                                        -0.000575 -0.000575
                                                  0.005517
```

Рисунок 2.2.3

Порівнявши показники дельта можна побачити, що показники стають точнішими зі збільшенням часу до виконання опціону.

Delta, trinomial call	Delta, trinomial put	Real delta,call	Real delta,put
0.510259	-0.491832	0.143	-0.859
0.516450	-0.488549	0.318	-0.681
0.522824	-0.483305	0.359	-0.640
0.543234	-0.465604	0.435	-0.564
0.564784	-0.449239	0.488	-0.513
0.593285	-0.424829	0.532	-0.471
0.626958	-0.403759	0.581	-0.435
0.675212	-0.368996	0.635	-0.387

Рисунок 2.2.4

2.3 Динамічне дельта хеджування триноміального дерева

Дельта хеджування - стратегія, яка передбачає відновлення балансу позицій хеджування в міру зміни ринкових умов. Іншими словами, стратегія, яка прагне застрахувати вартість портфеля за допомогою опціону пут[8].

Для триноміальної моделі було зроблено дельта хеджування. Параметри моделі: K=160, Sigma=0.32577

	Stock price	Delta	Shares	Shares cost	Cumulative cost
0	178.1025	0.5791	57.9143	10314.6747	10314.6747
1	190.5725	0.7309	15.1776	2892.4364	20629.3494
2	200.0250	0.8334	10.2440	2049.0561	30944.0241
3	199.0275	0.8230	-1.0351	-206.0137	41258.6987
4	189.6625	0.7205	-10.2490	-1943.8470	51573.3734
5	195.9400	0.7903	6.9783	1367.3278	61888.0481
6	196.6200	0.7976	0.7292	143.3683	72202.7228
7	206.3700	0.8969	9.9265	2048.5393	82517.3975
8	199.0500	0.8232	-7.3616	-1465.3181	92832.0722
9	197.9800	0.8120	-1.1217	-222.0695	103146.7469
10	220.6800	1.0000	18.7974	4148.2185	113461.4216
11	223.9600	1.0000	-0.0000	-0.0000	123776.0962

Рисунок 2.3.1

За період 11 тижнів куплено 119 767 акцій, продано 19 767 акцій, володіємо сумою $223 960, потрачено $123 776, профіт - $100 184.

Висновки

Для реальних даних було ідентифіковано і побудовано триноміальну модель з якої видно, що вона містить реальні значення цін акцій. Триноміальна модель на досліджуваних даних дає результати кращі, аніж біноміальна, оскільки значення похибки менше, що підтверджує припущення про вищу точність моделі.

Методом ітерацій було знайдено ціни американського пут опціона для 8 дат і порівняно з ринковими цінами.

Обчислена чутливість моделі за допомогою греків дельта та гамма.

Проведено дельта хеджування за період 11 тижнів з якого видно, що прибуток склав $100 184.

Список використаної літератури

1. Clifford P. Pricing Options Using Trinomial Trees [Електронний ресурс] / P. Clifford, Yan Wang, O. Zaboronski. – 2009. – Режим доступу до ресурсу: https://warwick.ac.uk/fac/sci/maths/people/staff/oleg_zaboronski/fm/trinomial_tree_2009.pdf.

2. John C. Hull. Options, Futures and Other Derivatives / John C. Hull., 2002. – 892 с.

3. TRINOMIAL OR BINOMIAL: ACCELERATING AMERICAN PUT OPTION PRICE ON TREES [Електронний ресурс] / JIUN HONG CHAN, MARK JOSHI, ROBERT TANG, CHAO YANG – Режим доступу до ресурсу: https://fbe.unimelb.edu.au/__data/assets/pdf_file/0008/2591891/175.pdf

4. The Trinomial Asset Pricing Model [Електронний ресурс] // Chalmers. – 2016. – Режим доступу до ресурсу: https://publications.lib.chalmers.se/records/fulltext/238499/238499.pdf.

5. Boyle P. Option Valuation Using a Three-Jump Process. International Options Journal 3, 1986.

6. Hu Xiaoping, Guo Jiafeng, Du Tao, Cui Lihua, Cao Jie, "Pricing Options Based on Trinomial Markov Tree", Discrete Dynamics in Nature and Society, vol. 2014, ArticleID 624360, 7 pages, 2014. – Режим доступу до ресурсу: https://doi.org/10.1155/2014/624360

7. Lattice tree methods [Електронний ресурс] – Режим доступу до ресурсу: https://www.math.hkust.edu.hk/~maykwok/courses/Adv_Num_Met/Topic1.pdf.

8. https://www.nasdaq.com/glossary/d/delta-hedge

Додатки

#BinomialModel.py

```
import math
import numpy as np
import pandas as pd
from scipy import stats

class BinomialOption(object):
    def __init__(self, S0, K, r, T, N, sigma, pu=0.5, mu=0, is_call=True,
is_am=False):
        self.S0 = S0
        self.K = K
        self.T = T
        self.N = max(1,N)
        self.STs = []
        self.POs=[]

        self.is_call = is_call
        self.is_am = is_am
        self.dt=self.T/float(self.N)

        self.sigma = sigma
        self.r=r if mu==0 else  mu+self.sigma*self.sigma/2
        self.u = math.exp(self.sigma*math.sqrt(self.dt))
        self.d = 1/self.u
        self.pu = (math.exp(self.r*self.dt)-self.d)/(self.u-self.d)
        self.pd = 1-self.pu


    def check_early_exercise(self, payoffs, node):
        if self.is_call:
            return np.maximum(payoffs, self.STs[node] - self.K)
        else:
            return np.maximum(payoffs, self.K - self.STs[node])

    def init_payoffs_tree(self):
        if self.is_call:
            return np.maximum(0, self.STs[self.N]-self.K)
        else:
            return np.maximum(0, self.K-self.STs[self.N])
```

```
    def init_stock_price_tree(self):
        self.STs = [np.array([self.S0])]

        for i in range(self.N):
            prev_branches = self.STs[-1]
            st = np.concatenate(
                (prev_branches*self.u,
                 [prev_branches[-1]*self.d]))
            self.STs.append(st)



    def traverse_tree(self, payoffs):
        for i in reversed(range(self.N)):
            self.POs.insert(0,payoffs)
            payoffs = (payoffs[:-1]*self.pu +
                       payoffs[1:]*self.pd)*math.exp(-self.r*self.dt)

            if self.is_am:
                payoffs = self.check_early_exercise(payoffs,i)
        self.POs.insert(0,payoffs)
        return payoffs

    def print_tree(self):
        result=pd.DataFrame(self.STs).fillna('').T
        for k in range(len(result.columns)-1):
            result.loc[:,:k]=result.loc[:,:k].shift(1,fill_value='')
        print(result.to_string(header=True,index=False))

    def print_payoffs(self):
        result=pd.DataFrame(self.POs).fillna('').T
        for k in range(len(result.columns)-1):
            result.loc[:,:k]=result.loc[:,:k].shift(1,fill_value='')
        print(result.to_string(header=True,index=False))

    def delta(self,S):
        d1 = (np.log(S/self.K) + (self.r + self.sigma**2/2)*self.dt) /
(self.sigma*np.sqrt(self.dt))
        if self.is_call:
            return stats.norm.cdf(d1)
        return stats.norm.cdf(d1)-1

    def gamma(self,S):
```

```
        d1 = (np.log(S/self.K) + (self.r + self.sigma**2/2)*self.dt) /
(self.sigma*np.sqrt(self.dt))
        d2 = d1 - self.sigma * np.sqrt(self.dt)
        return self.K*np.exp(-
self.r*self.dt)/S/S/self.sigma/math.sqrt(2*math.pi*self.dt)*np.exp(-d2*d2/2)

    def price(self,print_tree=False,print_payoffs=False):
        self.init_stock_price_tree()
        payoffs = self.init_payoffs_tree()
        payoffs = self.traverse_tree(payoffs)
        if print_tree:
            self.print_tree()
            print()
        if print_payoffs:
            self.print_payoffs()
            print()
        return payoffs[0]



#TrinomialModel.py

import math
import numpy as np
import pandas as pd


class TrinomialOption(object):
    def __init__(self, S0, K, r, T, N, sigma, mu=0, is_call=True, is_am=False,
lmbd=1.2):
        self.S0 = S0
        self.K = K
        self.T = T
        self.N = max(1,N)
        self.STs = []
        self.POs=[]
        self.deltas=[]
        self.result_tree=[]

        self.is_call = is_call
        self.is_am = is_am
        self.dt=self.T/float(self.N)
```

```
        self.sigma = sigma

        self.r=r if mu==0 else  mu+self.sigma*self.sigma/2
        self.u = math.exp(self.sigma*math.sqrt(lmbd*self.dt))
        self.d = 1/self.u
        self.m=1

        self._M=math.exp(self.r*self.dt)
        self._V=math.exp(self.sigma*self.sigma*self.dt)
        self.pu=(self._M*self._M*self._V-self._M*(self.d+1)+self.d)/((self.u-
self.d)*(self.u-1))
        self.pd=(self._M*self._M*self._V-self._M*(self.u+1)+self.u)/((self.d-
self.u)*(self.d-1))
        self.pm = 1 - self.pu - self.pd


    def check_early_exercise(self, payoffs, node):
        if self.is_call:
            return np.maximum(payoffs, self.STs[node] - self.K)
        else:
            return np.maximum(payoffs, self.K - self.STs[node])

    def init_payoffs_tree(self):
        if self.is_call:
            return np.maximum(0, self.STs[self.N]-self.K)
        else:
            return np.maximum(0, self.K-self.STs[self.N])

    def init_stock_price_tree(self):
        self.STs = [np.array([self.S0])]

        for i in range(self.N):
            prev_branches = self.STs[-1]
            self.ST = np.concatenate(
                (prev_branches*self.u,
                 [prev_branches[-1]*self.m, prev_branches[-1]*self.d]))
            self.STs.append(self.ST)

    def traverse_tree(self, payoffs):
        for i in reversed(range(self.N)):
            self.POs.insert(0,payoffs)
            payoffs = (payoffs[:-2] * self.pu +
                       payoffs[1:-1] * self.pm +
```

```
                payoffs[2:] * self.pd) * math.exp(-(self.r)*self.dt)

            if self.is_am:
                payoffs = self.check_early_exercise(payoffs,i)
        self.POs.insert(0,payoffs)
        return payoffs

    def print_tree(self):
        self.result_tree=pd.DataFrame(self.STs).fillna('').T
        for k in range(len(self.result_tree.columns)-1):
            self.result_tree.loc[:,:k]=self.result_tree.loc[:,:k].shift(1,fill_value='')
        res=self.result_tree
        print(res.to_string(header=True,index=False))

    def print_payoffs(self):
        result=pd.DataFrame(self.POs).fillna('').T
        for k in range(len(result.columns)-1):
            result.loc[:,:k]=result.loc[:,:k].shift(1,fill_value='')
        print(result.to_string(header=True,index=False))

    def delta(self,S):
        dS=S/10
        opt1=TrinomialOption(S+dS, self.K, self.r, self.T, self.N, self.sigma,0,
self.is_call, self.is_am)
        opt2=TrinomialOption(S-dS, self.K, self.r, self.T, self.N, self.sigma,0,
self.is_call, self.is_am)
        return (opt1.price()-opt2.price())/dS/2

    def gamma(self,S):
        dS=S/10
        c_S0=TrinomialOption(S, self.K, self.r, self.T, self.N, self.sigma,0,
self.is_call, self.is_am).price()
        c_positive_dS=TrinomialOption(S+dS, self.K, self.r, self.T, self.N,
self.sigma,0, self.is_call, self.is_am).price()
        c_negative_dS=TrinomialOption(S-dS, self.K, self.r, self.T, self.N,
self.sigma,0, self.is_call, self.is_am).price()
        return (c_positive_dS-2*c_S0+c_negative_dS)/(dS*dS)


    def print_deltas(self):
        deltas_tree=pd.DataFrame(self.STs).fillna(0).T.apply(lambda x:
self.delta(x))
        for k in range(len(deltas_tree.columns)-1):
```

```
            deltas_tree.loc[:,:k]=deltas_tree.loc[:,:k].shift(1,fill_value='')
        deltas_tree.fillna('',inplace=True)
        print(deltas_tree.to_string(header=True,index=False))

    def print_gammas(self):
        gammas_tree=pd.DataFrame(self.STs).fillna(0).T.apply(lambda x:
self.gamma(x))
        for k in range(len(gammas_tree.columns)-1):
            gammas_tree.loc[:,:k]=gammas_tree.loc[:,:k].shift(1,fill_value='')
        gammas_tree.fillna('',inplace=True)
        print(gammas_tree.to_string(header=True,index=False))

    def delta_hedging(self,n,S_list,K,rate,T,N,sigma,is_call,is_am):
        delta_list=[]
        option=TrinomialOption(S_list[0], K, r=rate, T=T, N=1,sigma=sigma,
mu=0, is_call=is_call, is_am=is_am)
        delta_list.append(option.delta(S_list[0]))
        shares=[]
        shares.append(delta_list[0]*100)
        shares_cost=[]
        shares_cost.append(shares[0]*S_list[0])
        cum_cost=[]
        cum_cost.append(shares_cost[0])

        interest_cost=[]
        delta_rate=rate*7/365
        interest_cost.append(shares_cost[0]*delta_rate)
        for i in range(1,n):
            option=TrinomialOption(S_list[i], K, r=rate, T=T, N=1,sigma=sigma,
mu=0, is_call=is_call, is_am=is_am)
            delta_list.append(option.delta(S_list[i]))
            shares.append((delta_list[i]-delta_list[i-1])*100)
            shares_cost.append(shares[-1]*float(S_list[i]))
            cum_cost.append(cum_cost[-1]+shares_cost[0])
            interest_cost.append(interest_cost[-1]+shares_cost[-1]*delta_rate)
        delta_set=pd.DataFrame({"Stock
price":S_list[:n],"Delta":delta_list,"Shares":shares,"Shares
cost":shares_cost,"Cumulative cost":cum_cost})#,"Interest cost":interest_cost})
        return delta_set

    def price(self,print_tree=False,print_payoffs=False):
        self.init_stock_price_tree()
        payoffs = self.init_payoffs_tree()
```

```
        payoffs = self.traverse_tree(payoffs)

        if print_tree:
            self.print_tree()
            print()
        if print_payoffs:
            self.print_payoffs()
            print()
        #self.print_deltas()
        return payoffs[0]


#ExperimentalResults.py
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from TrinomialModel import TrinomialOption as TO
from BinomialModel import BinomialOption as BO



data=pd.read_csv("HistoricalData_NVIDIA.csv",parse_dates=["Date"])
data=data.reindex(index=data.index[::-1])
data["Close"]=data["Close/Last"].str[1:].astype("float")
data=pd.DataFrame(data["Close"].values.tolist())
x=np.log(data/data.shift(1))[1:]
data.plot(xlabel="Day",ylabel="Price",legend=False,yticks=range(150,331,15))
mu=x.mean()
dt=1/252
s=np.var(x)/dt
r=mu+s*s/2

data.tail(20)
S=data[0].iloc[-1]
print(S)
strike=170


option=BO(S,strike, r=r, T=132/252, N=5,sigma=s, mu=0, is_call=False,
is_am=True)
print("Option price = "+str(option.price(True,True)))
```

```
option=TO(S,strike, r=r, T=132/252, N=5,sigma=s, mu=0, is_call=True,
is_am=True)
print("Option price = "+str(option.price(True,True)))
option.print_deltas()
option.print_gammas()


S=data[0].iloc[-1]
strike=165

dates=[2,7,12,32,72,132,252,504]
nasdaq_call=[4.15,2.92,5.06,13.76,19.72,28.11,53.95,53.98]
nasdaq_put=[3.35,10.95,8.10,11.62,18.13,17.15,33.35,37.69]
n=2

bin_call=[BO(S,strike, r=r, T=i/252, N=n,sigma=s, mu=0, is_call=True,
is_am=True) for i in dates]
b_call=[i.price(False,False) for i in bin_call]

tr_call=[TO(S,strike, r=r, T=i/252, N=2*n,sigma=s, mu=0, is_call=True,
is_am=True) for i in dates]
tr_option_call=[i.price(False,False) for i in tr_call]

bin_put=[BO(S,strike, r=r, T=i/252, N=n,sigma=s, mu=0, is_call=False,
is_am=True) for i in dates]
b_put=[i.price(False,False) for i in bin_put]

tr_put=[TO(S,strike, r=r, T=i/252, N=2*n,sigma=s, mu=0, is_call=False,
is_am=True) for i in dates]
tr_option_put=[i.price(False,False) for i in tr_put]

results=pd.DataFrame({"Days to expiration date":dates,"Binomial call":
b_call,"Trinomial call": tr_option_call,"Real call option":nasdaq_call,
                      "Binomial put": b_put,"Trinomial put": tr_option_put,"Real put
option":nasdaq_put})


results.style.applymap(lambda x:"background-color: #ffff99",subset=['Real call
option', 'Real put option'])


tr_delta_call=[i.delta(S) for i in tr_call]
```

```
tr_delta_put=[i.delta(S) for i in tr_put]

delta_nasdaq_call=[0.143,0.318,0.359,0.435,0.488,0.532,0.581,0.635]
delta_nasdaq_put=[-0.859,-0.681,-0.64,-0.564,-0.513,-0.471,-0.435,-0.387]
deltas=pd.DataFrame({"Delta, trinomial call":tr_delta_call,"Delta, trinomial
put":tr_delta_put,"Real delta,call":delta_nasdaq_call,"Real
delta,put":delta_nasdaq_put})


x_axis=list(map(str, results["Days to expiration date"]))
plt.plot(x_axis, results["Trinomial call"],label='Trinomial call option')
plt.plot(x_axis, results["Binomial call"],label='Binomial call option')
plt.plot(x_axis, results["Real call option"],label="Real call option")
plt.legend()
plt.xlabel("Working days to expiration date")
plt.ylabel("Option price")
plt.yticks(np.arange(0, max(nasdaq_call), 3))
plt.show()




plt.plot(x_axis, results["Trinomial put"],label='Trinomial put option')
plt.plot(x_axis, results["Binomial put"],label='Binomial put option')
plt.plot(x_axis, results["Real put option"],label="Real put option")
plt.legend()
plt.xlabel("Working days to expiration date")
plt.ylabel("Option price")
plt.yticks(np.arange(0, max(nasdaq_put), 3))
plt.show()


errors_res=pd.DataFrame()
errors_res["Error,binomial call"]=abs(results["Binomial call"]-results["Real call
option"])/results["Real call option"]
errors_res["Error,binomial put"]=abs(results["Binomial put"]-results["Real put
option"])/results["Real put option"]
errors_res["Error,trinomial call"]=abs(results["Trinomial call"]-results["Real
call option"])/results["Real call option"]
errors_res["Error,trinomial put"]=abs(results["Trinomial put"]-results["Real put
option"])/results["Real put option"]
errors_res
```

```
opt_type=["Binomial call","Binomial put","Trinomial call","Trinomial put"]
rmse=[]
rmse.append((sum((results["Binomial call"]-results["Real call
option"])**2)/len(results["Binomial call"]))**0.5)
rmse.append((sum((results["Binomial put"]-results["Real put
option"])**2)/len(results["Binomial put"]))**0.5)
rmse.append((sum((results["Trinomial call"]-results["Real call
option"])**2)/len(results["Trinomial call"]))**0.5)
rmse.append((sum((results["Trinomial put"]-results["Real put
option"])**2)/len(results["Trinomial put"]))**0.5)
rmse=pd.DataFrame({"Option type": opt_type, "RMSE": rmse})
rmse


errors_res.describe()

S_list=data[0].values[::5]
dt=1/252
mu=x.mean()
s=np.var(x)
sigma=s/dt
r=mu+sigma*sigma/2
option=TO(S,170, r=r, T=7/252, N=3,sigma=s, mu=0, is_call=False,
is_am=True)
res_heg=option.delta_hedging(20,S_list,160, r, 0.3846, 1,sigma, is_call=True,
is_am=True).iloc[:12,:]
print(res_heg["Shares"].agg([(lambda col : col[col < 0].sum()),
                    (lambda col : col[col > 0].sum())
                    ]))
res_heg.style.format('{:.4f}')
```