Оглавление

[Введение 2](#_Toc27559472)

[Теоретическая часть 4](#_Toc27559473)

[Практическая часть 7](#_Toc27559474)

[Заключение 15](#_Toc27559475)

[Список использованной литературы 16](#_Toc27559476)

[Приложение 1 17](#_Toc27559477)

# Введение

Динамические процессы, происходящие в экономических системах, чаще всего проявляются в виде ряда последовательно расположенных в хронологическом порядке значений того или иного показателя, который в своих изменениях отражает ход развития изучаемого явления в экономике. Такие данные относятся к недетерминированным, т.е. случайным процессам.Случайный процесс невозможно описать во всех деталях, невозможно с заданной точностью предсказать его значения в будущий момент времени.

При обработке и анализе экономических данных стараются выделить зависимости или взаимную корреляцию с экономическими индексами и акциями, а также моделирование данных с помощью математических функций на определенном промежутке. Временной ряд экономических показателей можно разложить на четыре структурно образующих элемента: тренд, сезонная компонента, циклическая компонента и случайная компонента. Под трендом понимается устойчивое систематическое изменение процесса в течение продолжительного времени. Тренд, сезонная и циклическая компоненты называются регулярными, или систематическими, компонентами временного ряда.

В данной работе предлагается проанализировать экономический временной ряд фондового рынка Российской Федерации. В качестве экономического временного ряда выбран индекс РТС. Для моделирования данных предлагается использовать геометрическое броуновское движение (GBM). GBM применяется в целях моделирования ценообразования на финансовых рынках и используется преимущественно в моделях ценообразования опционов, так как GBM может принимать любые положительные значения. GBM является разумным приближением к реальной динамике цен акций, не учитывающем, однако, редкие события - выбросы.

В результате анализа экономического ряда предлагается оценить взаимную корреляцию индекса РТС с акциями российских компаний с аналогичный период. Предполагается, что данные будут иметь схожие результаты.

Цель работы: обработать и проанализировать экономический временной ряд фондового рынка России.

Задачи:

1. Выбрать, подготовить и визуализировать данные;
2. Выделить тренды;
3. Смоделировать данные с помощью геометрического броуновского движения;
4. Посчитать статистические характеристики для данных;
5. Оценить полученные результаты;
6. Оценить взаимную корреляцию между исходными данными и смоделированными.

# Теоретическая часть

Для анализа были выбраны данные, представляющие российский фондовый рынок за 3 года (с 12.12. 2016 по 10.12.2019): индекс РТС, акции Сбербанка, Газпрома и ВТБ.

Индекс РТС (RTS) - старейший фондовый индекс России, расчет которого начался 1 сентября 1995 года со значения в 100 пунктов. Представляет собой ценовой взвешенный по рыночной капитализации (free-float) композитный индекс российского фондового рынка, включающий наиболее ликвидные акции крупнейших и динамично развивающихся российских компаний. Расчет индекса производится на основе цен акций, выраженных в долларах США.

Публичное акционерное общество Сбербанк — российский финансовый конгломерат, крупнейший транснациональный и универсальный банк России, Центральной и Восточной Европы. Контролируется Центральным банком Российской Федерации, которому принадлежит 50 % уставного капитала плюс одна голосующая акция.

ПАО «Газпром» — российская транснациональная энергетическая компания, более 50 % акций которой принадлежит государству. Является холдинговой компанией Группы «Газпром». Непосредственно ПАО «Газпром» осуществляет только продажу природного газа и сдаёт в аренду свою газотранспортную систему. Основные направления деятельности — геологоразведка, добыча, транспортировка, хранение, переработка и реализация газа, газового конденсата и нефти, реализация газа в качестве моторного топлива, а также производство и сбыт тепло- и электроэнергии.

Банк ВТБ (ПАО) — советский и российский универсальный коммерческий банк c государственным участием (60,9 % принадлежит государству). Второй по величине активов банк страны и первый по размеру уставного капитала. Главный офис банка находится в Москве, зарегистрирован банк в Санкт-Петербурге.

Для моделирования процессов использовалось геометрическое броуновское движение:

|  |  |
| --- | --- |
|  | (1) |

где:

|  |  |
| --- | --- |
|  | (2) |
|  | (3) |
|  | (4) |

В ходе работы были получены и проанализированы следующие статистические характеристики:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Среднее значение |  | (5) |
| 2. | Дисперсия |  | (6) |
| 3. | Стандартное отклонение |  | (7) |
| 4. | Коэффициент асимметрии |  | (8) |
| 5. | Асимметрия |  | (9) |
| 6. | Центральный момент четвертого порядка |  | (10) |
| 7. | Куртозис |  | (11) |
| 8. | Минимальное значение |  | (12) |
| 9. | Максимальное значение |  | (13) |

А также рассчитана взаимная корреляция:

|  |  |
| --- | --- |
|  | (14) |

# Практическая часть

Для выполнения поставленной задачи было написано приложение на языке Python. Были использованы следующие библиотеки: tkinter, matplotlib, NumPy, math и csv. Приложение имеет объектно-ориентированную архитектуру, представленную следующими классами: model, analysis и MainWindow.

Класс model отвечает за расчёт тренда. Схема класса представлена на рисунке 1.

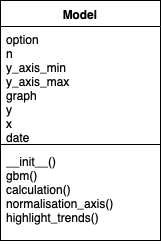


Рисунок 1 – Схема класса Model

Класс analysis реализует расчет статистик трендов. Схема класса представлена на рисунке 2.

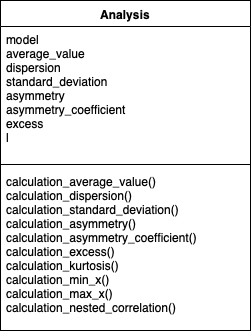


Рисунок 2 – Схема класса Analysis

Класс MainWindow реализует интерфейс пользователя и отрисовку данных. Схема класса представлена на рисунке 3.

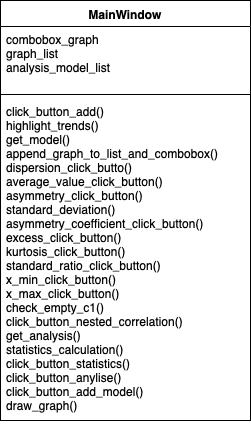


Рисунок 3 – Схема класса MainWindow

Входные данные по индексу и акциями представлены были представлены в формате csv, для работы были импортированы в приложения. Полученные графики представлены на рисунках 4-7.

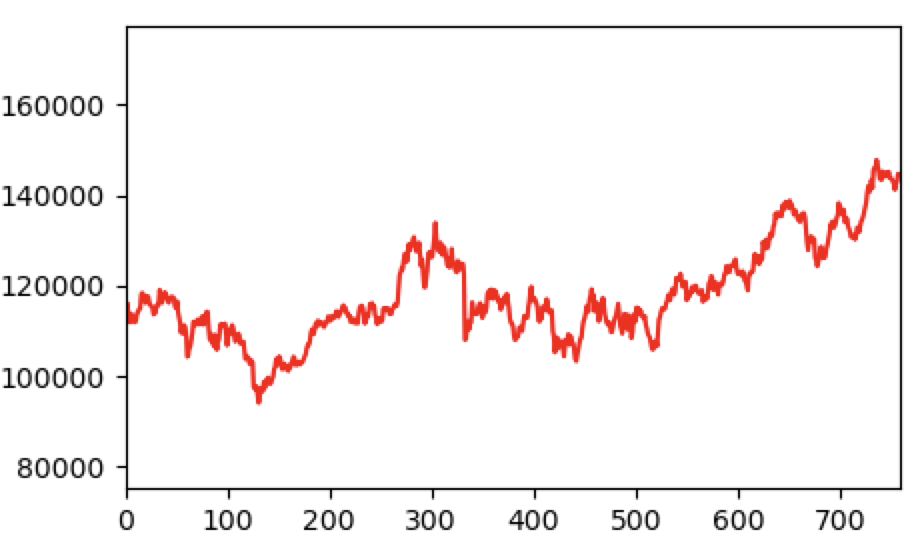


Рисунок 4 – Индекс РТС

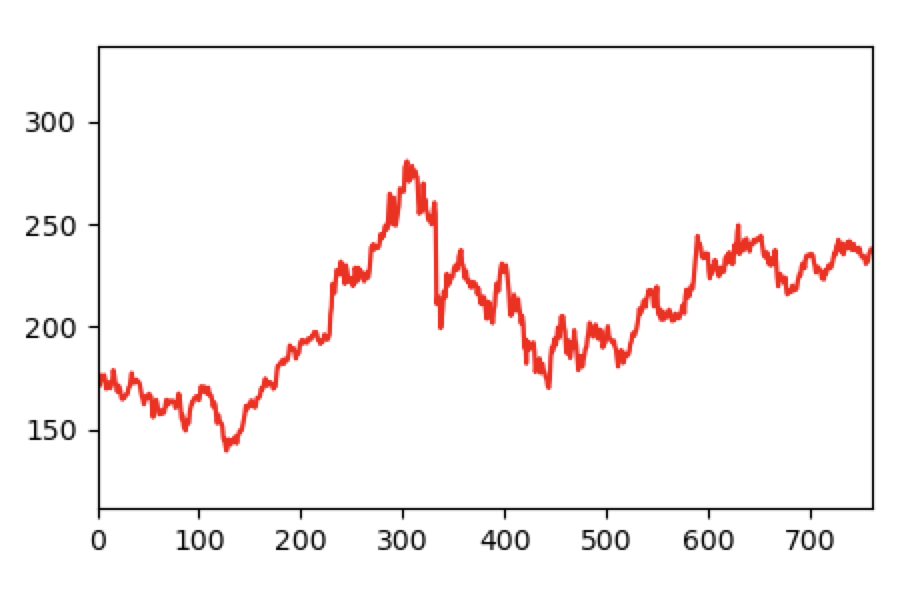


Рисунок 5 – Курс акций Сбербанка

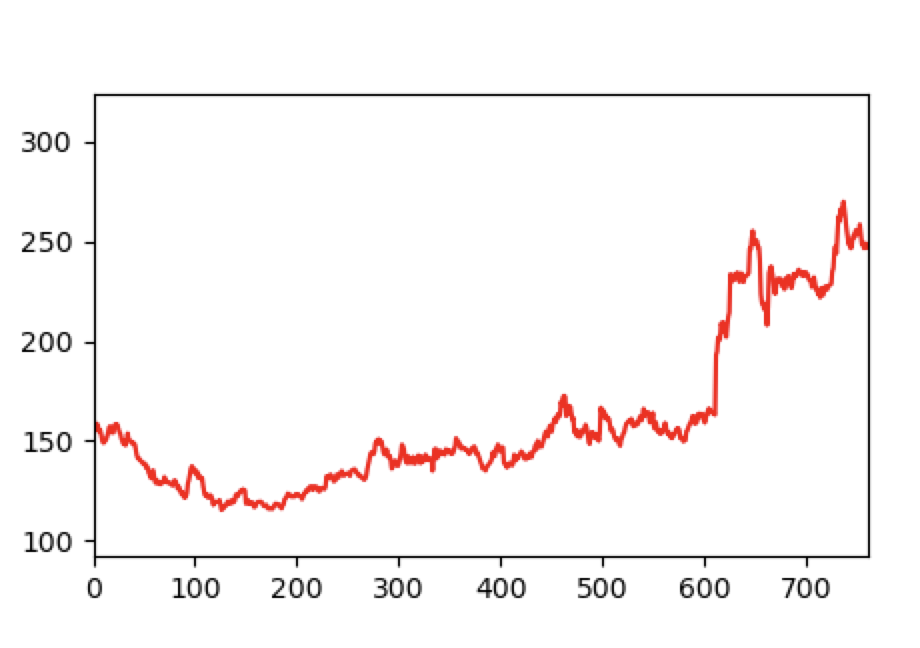


Рисунок 6 – Курс акций Газпрома

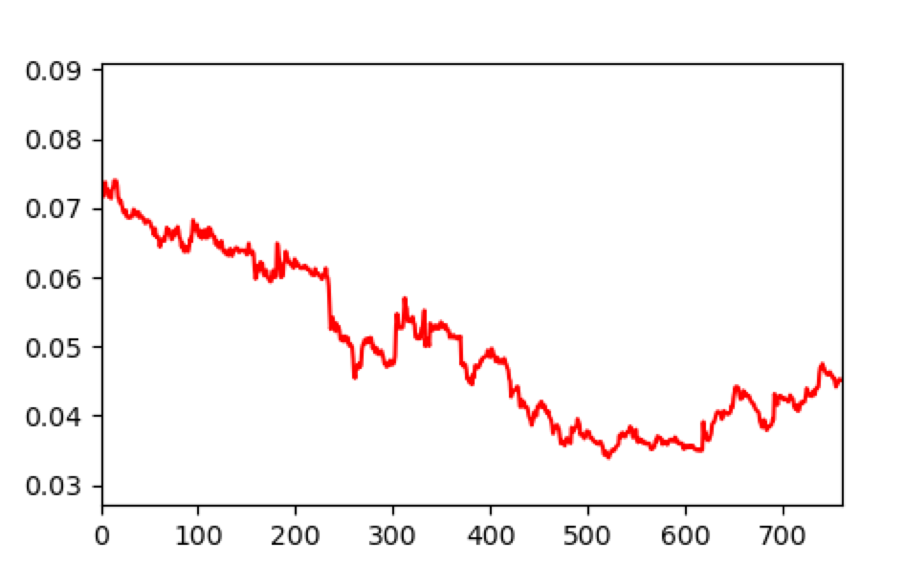


Рисунок 7 – Курс акций ВТБ

С помощью геометрического броуновского движения были рассчитаны коэффициенты и смоделированы графики согласно формулам 1, 2, 3, 4. Результаты отображены на рисунках 8-11, код функции размещен в приложении.

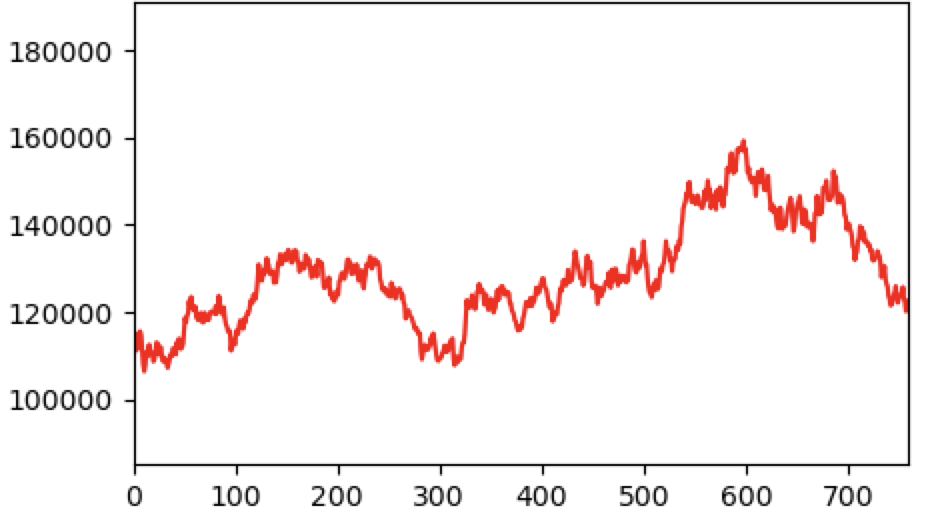


Рисунок 8 – Индекс РТС, смоделированный с помощью GBM

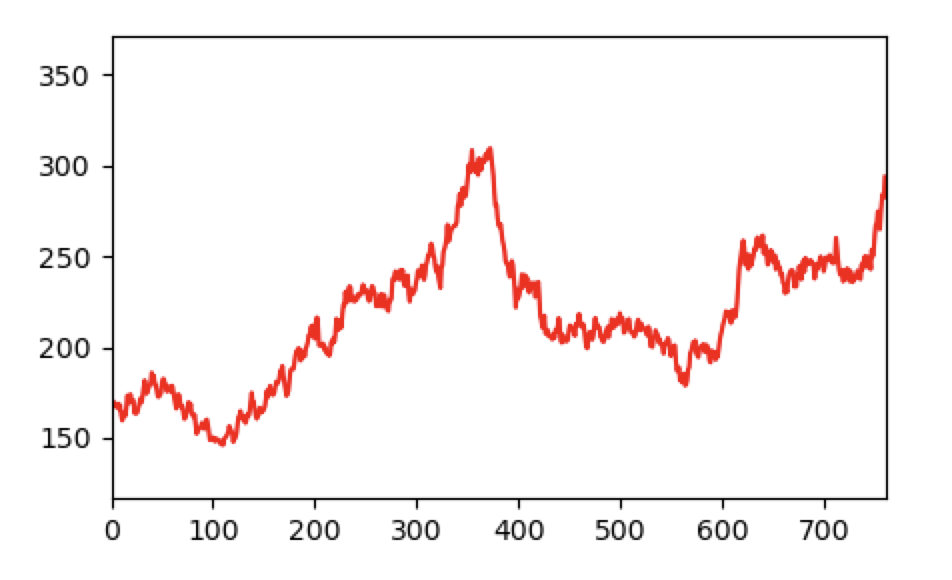


Рисунок 9 – Курс акций Сбербанка, смоделированный с помощью GBM

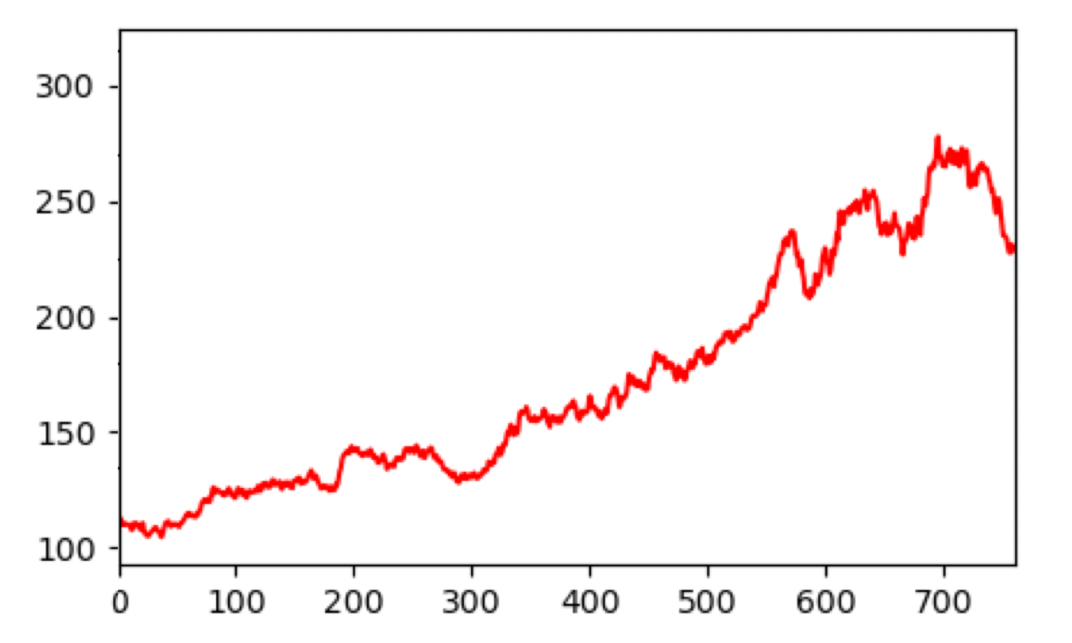


Рисунок 10 – Курс акций Газпрома, смоделированный с помощью GBM

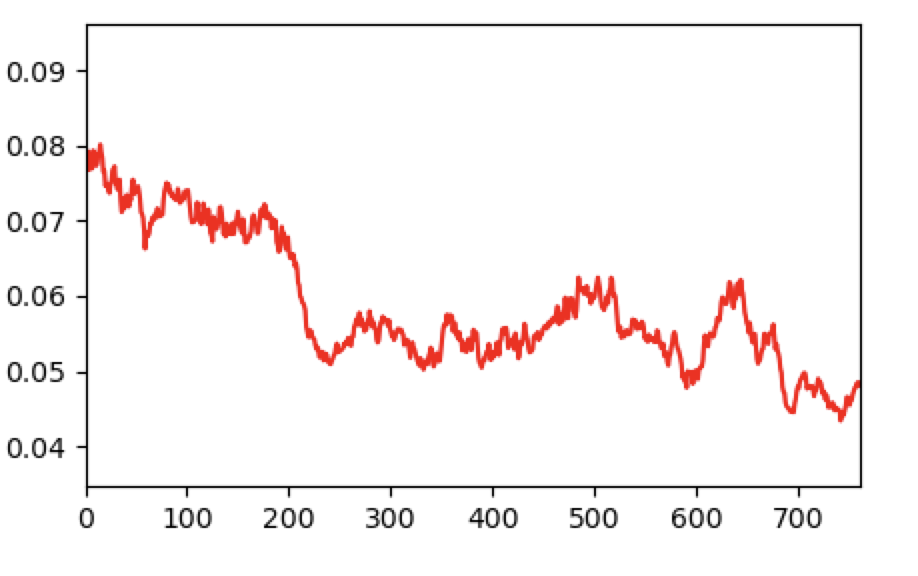


Рисунок 11 – Курс акций ВТБ, смоделированный с помощью GBM

Для расчета статистик используется класс Analysis и формулы с 5-13. Результаты представлены в таблице 1 и 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Индекс РТС | Сбербанк | РТС - GBM | Сбербанк - GBM |
| Среднее значение | 117953.35 | 205.8 | 112852.6 | 197.36 |
| Дисперсия | 112828404.12 | 961.8 | 151634414.8 | 906.97 |
| Стандартное отклонение | 10622.07 | 31.01 | 12313.9 | 30.1161 |
| Коэффициент асимметрии | 0.629202 | -0.0423 | 0.78123 | -0.0325 |
| Асимметрия | 754081415339.4 | -1264.5 | 754336872764.6 | -1354.5 |
| Центральный момент четвертого порядка | 3.8221e^16 | 2005425.36 | 3.8342e^16 | 2004322.41 |
| Куртозис | 0.0024423 | -0.832 | 0.0045663 | -0.911 |
| Минимальное значение | 94090.0 | 139.61 | 86560.19 | 118.671 |
| Максимальное значение | 147790.0 | 280.82 | 152633.3 | 284.447 |

Таблица 1 – Результаты расчетов статистик данных исходных данных и смоделированных данных



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Индекс РТС | Сбербанк | РТС - GBM | Сбербанк - GBM |
| СЗ промежуток 1 | 113954.0 | 167.8990 | 113974.1 | 171.45320 |
| СЗ промежуток 2 | 105368.8 | 157.28855 | 105370.2 | 169.34526 |
| СЗ промежуток 3 | 108618.9 | 183.31368 | 108623.8 | 179.43248 |
| СЗ промежуток 4 | 119055.8 | 236.94671 | 119024.6 | 220.85782 |
| СЗ промежуток 5 | 120183.3 | 237.24881 | 120180.1 | 242.23486 |
| СЗ промежуток 6 | 110949.0 | 200.52157 | 110943.2 | 221.54363 |
| СЗ промежуток 7 | 112548.9 | 192.33815 | 112580.4 | 199.65435 |
| СЗ промежуток 8 | 119721.4 | 216.98184 | 119780.4 | 234.54323 |
| СЗ промежуток 9 | 130399.0 | 232.69657 | 130340.0 | 236.65423 |
| СЗ промежуток 10 | 135704.0 | 232.825394 | 135732.0 | 235.82432 |
| Д промежуток 1 | 11060608.0 | 31.81099 | 11060625.0 | 41.814352 |
| Д промежуток 2 | 29479045.2 | 76.554278 | 29479234.1 | 100.53443 |
| Д промежуток 3 | 22957572.1 | 122.7568 | 22957967.1 | 143.45843 |
| Д промежуток 4 | 39435208.2 | 296.486090 | 39435672.5 | 275.53454 |
| Д промежуток 5 | 37360598.2 | 511.79289 | 37360123.7 | 490.53445 |
| Д промежуток 6 | 15978779.1 | 257.909515 | 15978432.2 | 200.66543 |
| Д промежуток 7 | 8672526.8 | 38.660146 | 8672347.2 | 46.435435 |
| Д промежуток 8 | 7056148.5 | 137.480362 | 7056190.2 | 100.34243 |
| Д промежуток 9 | 32566896.4 | 67.8709804 | 32566345.9 | 45.534344 |
| Д промежуток 10 | 41552389.3 | 34.391677 | 41552871.5 | 38.435435 |

Таблица 2 – Результаты расчетов среднего значения и дисперсии исходных смоделированных данных по промежуткам

Также была рассчитана взаимная корреляция между исходными данными курса акций Сбербанка и Газпрома, исходными данными курса акций Сбербанка и ВТБ и исходными данными курса акций Сбербанка и смоделированными данными курса акций Сбербанка. Результаты представлены на рисунках 12 – 14.

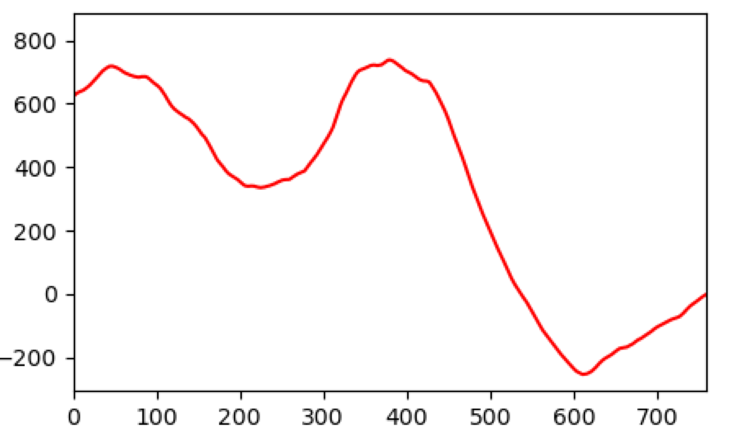


Рисунок 12 – Взаимная корреляция курса акций Сбербанка и курса акций Газпрома

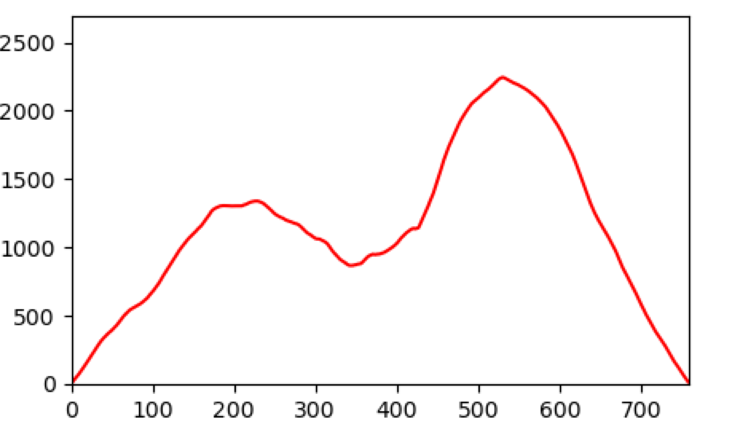


Рисунок 13 – Взаимная корреляция курса акций Сбербанка и курса акций ВТБ

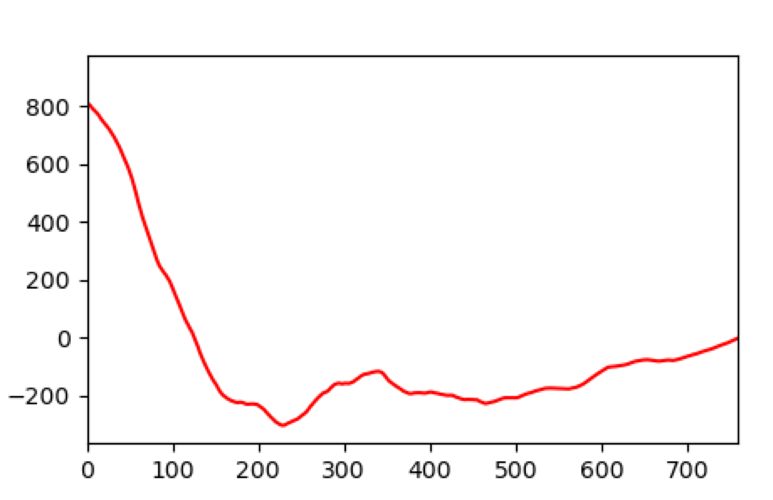


Рисунок 14 – Взаимная корреляция курса акций Сбербанка и смоделированного курса акций Сбербанка

Также стоит отметить, что процессы исходных данных курса акций Сбербанка и смоделированных данных акций Сбербанка связаны согласно построению функций взаимной корреляций, а исходные данные курса акций Сбербанка и Газпрома и исходные данные курса акций Сбербанка и ВТБ связаны на некоторых промежутках, а на некоторых не связаны.

# Заключение

В данной работе был проведен анализ экономических временных рядов фондового рынка РФ. С помощью геометрического броуновского движения удалось добиться похожих моделей внешне.

Также входе работы были выделены тренды для исходных данных и смоделированных. По результатам этой работы можно отметить совпадение направления тренда и скорость изменений смоделированных данных по сравнению с исходными данными. Также рассчитанные статистики показывают незначительные изменения. Но все равно исходные данные являются недетерминированными и соответственно есть различия между исходными данными, где изменения показателей происходит под влиянием процессов в экономики и математически смоделированными данными с помощью геометрического броуновского движения.

Таким образом, задачу моделирования случайных данных можно считать успешно достигнутой. Но для того, чтобы предсказывать с высокой точностью курс акций необходимо анализировать экономические показатели, и одного математического моделирования недостаточно.

# Список использованной литературы

1. Лекции Белых И.Н. по «Методам обработки экспериментальных данных».

2. Дж. Бендат, А. Пирсол."Прикладной анализ случайных данных". Мир, 1989, 540 с.

# Приложение 1

model. Py

**import** numpy **as** np  
**import** csv  
**import** copy  
  
  
*# Функция импорта котировок***def** import\_value(filename):  
 cost\_rts = []  
 date\_rts = []  
  
 csv.register\_dialect(**'pipes'**, delimiter=**';'**)  
 **with** open(filename, **'r'**, newline=**''**) **as** csv\_file:  
 reader = csv.reader(csv\_file, dialect=**'pipes'**)  
  
 **for** row **in** reader:  
 **try**:  
 r = float(row[2])  
 cost\_rts.append(r)  
 date\_rts.append(row[0])  
 **except**:  
 **pass** new\_data\_rts = []  
 **for** i **in** date\_rts:  
 temp = i[:4] + **"-"** + i[4:6] + **"-"** + i[6:8]  
 new\_data\_rts.append(temp)  
  
 **return** new\_data\_rts, cost\_rts  
  
  
**class** Model:  
 **def** \_\_init\_\_(self, option):  
 self.option = option  
 self.n = 0  
  
 self.y\_axis\_min = 0  
 self.y\_axis\_max = 0  
  
 self.y\_gaps\_10 = [] *# Промежутки исходных данных* self.graph = 0  
  
 *# Расчет геометрического Броуновского движения* **def** gbm(self):  
 all\_y = []  
 **for** i **in** range(len(self.y\_gaps\_10)):  
 y = copy.deepcopy(self.y\_gaps\_10[i])  
 n = len(y)  
  
 returns = []  
 **for** i **in** range(1, n):  
 value = (y[i] - y[i - 1]) / y[i - 1]  
 returns.append(value)  
 returns = np.array(returns)  
 average\_value = np.mean(returns)  
 standard\_deviation = np.std(returns)  
  
 T = n  
 mu = average\_value  
 sigma = standard\_deviation  
 S0 = y[0]  
 dt = 1  
 N = round(T / dt)  
 t = np.linspace(0, T, N)  
 W = np.random.standard\_normal(size=N)  
 W = np.cumsum(W) \* np.sqrt(dt) *### standard brownian motion ###* X = (mu - 0.5 \* sigma \*\* 2) \* t + sigma \* W  
 S = S0 \* np.exp(X) *### geometric brownian motion ###* y = S.tolist()  
 all\_y += y  
  
 n = len(all\_y)  
 self.x = np.arange(n)  
 self.y = np.array(all\_y)  
  
 *# Расчет промежутков* **def** highlight\_gaps(self):  
 interval = int(self.n / 10)  
 **for** i **in** range(10):  
 a = i \* interval  
 b = (i + 1) \* interval  
 y = self.y[a:b]  
 self.y\_gaps\_10.append(y)  
  
 **def** calculation(self):  
  
 *# РТС* **if** self.option == 1:  
 filename = **"input\_files/SPFB.RTS\_161210\_191210 (1).csv"** self.date, self.y = import\_value(filename)  
 self.n = len(self.y)  
 self.x = np.arange(self.n)  
 self.highlight\_gaps() *# Разбиваем данные на 10 равных промежутков  
  
 # Сбербанк* **if** self.option == 3:  
 filename = **"input\_files/SBER\_161212\_191210.csv"** self.date, self.y = import\_value(filename)  
 self.n = len(self.y)  
 self.x = np.arange(self.n)  
  
 *# Газпром* **if** self.option == 4:  
 filename = **"input\_files/GAZP\_161212\_191210 (1).csv"** self.date, self.y = import\_value(filename)  
 self.n = len(self.y)  
 self.x = np.arange(self.n)  
  
 *# ВТБ* **if** self.option == 5:  
 filename = **"input\_files/VTBR\_161212\_191210 (1).csv"** self.date, self.y = import\_value(filename)  
 self.n = len(self.y)  
 self.x = np.arange(self.n)  
  
 *# GBM РТС* **if** self.option == 2:  
 filename = **"input\_files/SPFB.RTS\_161210\_191210 (1).csv"** self.date, self.y = import\_value(filename)  
 self.n = len(self.y)  
 self.highlight\_gaps() *# Разбиваем данные на 10 равных промежутков* self.gbm()  
  
 *# GBM Сбербанк* **if** self.option == 6:  
 filename = **"input\_files/SBER\_161212\_191210.csv"** self.date, self.y = import\_value(filename)  
 self.n = len(self.y)  
 self.highlight\_gaps() *# Разбиваем данные на 10 равных промежутков* self.gbm()  
  
 *# GBM Газпром* **if** self.option == 7:  
 filename = **"input\_files/GAZP\_161212\_191210 (1).csv"** self.date, self.y = import\_value(filename)  
 self.n = len(self.y)  
 self.highlight\_gaps() *# Разбиваем данные на 10 равных промежутков* self.gbm()  
  
 *# GBM ВТБ* **if** self.option == 8:  
 filename = **"input\_files/VTBR\_161212\_191210 (1).csv"** self.date, self.y = import\_value(filename)  
 self.n = len(self.y)  
 self.highlight\_gaps() *# Разбиваем данные на 10 равных промежутков* self.gbm()  
  
 *# Нормализация осей* **def** normalisation\_axis(self):  
 self.y\_axis\_max = np.amax(self.y) \* 1.2  
 min = np.amin(self.y)  
 **if** min > 0:  
 self.y\_axis\_min = np.amin(self.y) \* 0.8  
 **else**:  
 self.y\_axis\_min = np.amin(self.y) \* 1.2  
  
 *# Выделяем в ручную тренд методом скользящего окна* **def** highlight\_trends(self, analyzed\_model):  
  
 size\_of\_window = 50  
 analysis\_model\_n = len(analyzed\_model.y)  
 sum\_value\_of\_window = 0  
 y = np.copy(analyzed\_model.y)  
  
 **for** i **in** range(analysis\_model\_n - size\_of\_window):  
 **for** j **in** range(size\_of\_window):  
 sum\_value\_of\_window += y[i + j]  
  
 average = sum\_value\_of\_window / size\_of\_window  
 y[i] = average  
 sum\_value\_of\_window = 0  
  
 **for** i **in** range(analysis\_model\_n - size\_of\_window, analysis\_model\_n):  
 **for** j **in** range(size\_of\_window):  
 sum\_value\_of\_window += y[i - j]  
  
 average = sum\_value\_of\_window / size\_of\_window  
 y[i] = average  
 sum\_value\_of\_window = 0  
  
 self.y = np.copy(y)  
 self.x = np.arange(len(self.y))

Analysis.py

**import** math  
**import** copy  
  
**import** numpy **as** np  
**from** model **import** Model  
  
  
**class** Analysis:  
 **def** \_\_init\_\_(self, model):  
  
 self.model = model *# Модель, которую анализиурем* self.all\_average\_value = [] *# Все средние значения* self.average\_value = 0 *# Среднее значения тренда* self.dispersion = 0 *# Дисперсия* self.standard\_deviation = 0 *# Стандартное отклонение* self.asymmetry = 0 *# Асимметрия* self.asymmetry\_coefficient = 0 *# Коэффициент асимметрии* self.standard\_ratio = 0 *# Стандартный коэффициент* self.excess = 0 *# Эксцесс* self.l = model.n - 1 *# Сдвиг  
  
 # Рассчет среднего значения* **def** calculation\_average\_value(self):  
  
 self.average\_value = np.mean(self.model.y)  
  
 print(**"Расчет среднего на 10 интервалах"**)  
  
 **for** i **in** range(len(self.model.y\_gaps\_10)):  
 average\_value = np.mean(self.model.y\_gaps\_10[i])  
 self.all\_average\_value.append(average\_value)  
 print(**"Среднее значение промежутка № "** + str(i + 1) + **" = "** + str(average\_value))  
  
 **return** self.average\_value  
  
 *# Рассчет дисперсии* **def** calculation\_dispersion(self):  
  
 **if** self.average\_value ==0:  
 self.calculation\_average\_value()  
  
  **dispersion = 0  
 n = self.model.n - 2  
 for i in range(n):  
 dispersion += (self.model.y[i] - self.average\_value) \* (self.model.y[i] - self.average\_value)  
  
 self.dispersion = dispersion / self.model.n**  
  
 **for** i **in** range(len(self.model.y\_gaps\_10)):  
 y = copy.deepcopy(self.model.y\_gaps\_10[i])  
 dispersion = 0  
 **for** j **in** range(len(y)):  
 dispersion += (y[j] - self.all\_average\_value[i]) \* (y[j] - self.all\_average\_value[i])  
  
 dispersion = dispersion / len(y)  
 print(**"Дисперсия промежутка № "** + str(i + 1) + **" = "** + str(dispersion))  
  
 **return** self.dispersion  
  
 *# Рассчет стандартного отклонения* **def** calculation\_standard\_deviation(self):  
  
 **if** self.dispersion == 0: *# Если не была расчитана диспресия* self.calculation\_dispersion(1)  
  
 self.standard\_deviation = math.sqrt(self.dispersion)  
  
 **return** self.standard\_deviation  
  
 *# Рассчет асимметрии* **def** calculation\_asymmetry(self):  
  
 **if** self.average\_value == 0:  
 self.calculation\_average\_value()  
  
 sum\_of\_values = 0  
  
 **for** i **in** range(self.model.n):  
 temp\_value = (self.model.y[i] - self.average\_value)  
 temp\_value = temp\_value \* temp\_value \* temp\_value  
 sum\_of\_values = sum\_of\_values + temp\_value  
  
 self.asymmetry = sum\_of\_values / self.model.n  
  
 **return** self.asymmetry  
  
 *# Рассчет коэффициента асимметрии* **def** calculation\_asymmetry\_coefficient(self):  
  
 **if** self.standard\_deviation == 0:  
 self.calculation\_standard\_deviation()  
  
 **if** self.asymmetry == 0:  
 self.calculation\_asymmetry()  
  
 sigma3 = self.standard\_deviation \* self.standard\_deviation \* self.standard\_deviation  
 self.asymmetry\_coefficient = self.asymmetry / sigma3  
  
 **return** self.asymmetry\_coefficient  
  
 *# Рассчет эксцесса* **def** calculation\_excess(self):  
  
 **if** self.average\_value == 0:  
 self.calculation\_average\_value()  
  
 sum\_of\_values = 0  
  
 **for** i **in** range(self.model.n):  
 temp\_value = (self.model.y[i] - self.average\_value)  
 temp\_value = temp\_value \*\* 4 *# Возведение в степень 4* sum\_of\_values = sum\_of\_values + temp\_value  
  
 self.excess = sum\_of\_values / self.model.n  
  
 **return** self.excess  
  
 *# Рассчет куртозис* **def** calculation\_kurtosis(self):  
  
 **if** self.standard\_deviation == 0:  
 self.calculation\_standard\_deviation()  
  
 **if** self.excess == 0:  
 self.calculation\_excess()  
  
 kurtosis = self.excess / self.standard\_deviation \*\* 4  
 kurtosis = kurtosis - 3  
  
 **return** kurtosis  
  
 *# Рассчет стандартного коэфициента* **def** calculation\_standard\_ratio(self):  
  
 sum\_of\_values = 0  
  
 **for** i **in** range(self.model.n):  
 temp\_value = self.model.y[i] \*\* 2  
 sum\_of\_values = sum\_of\_values + temp\_value  
  
 self.standard\_ratio = sum\_of\_values / self.model.n  
  
 **return** self.standard\_ratio  
  
 *# Рассчет среднеквадратичной ошибки* **def** calculation\_standard\_error(self):  
 **if** self.standard\_ratio == 0:  
 self.calculation\_standard\_ratio()  
  
 standard\_error = math.sqrt(self.standard\_ratio)  
  
 **return** standard\_error  
  
 *# Рассчет среднего абсолютного отклонения* **def** calculation\_mean\_absolute\_deviation(self):  
  
 **if** self.average\_value == 0:  
 self.calculation\_average\_value()  
  
 sum\_of\_values = 0  
  
 **for** i **in** range(self.model.n):  
 sum\_of\_values = sum\_of\_values + math.fabs(self.model.y[i] - self.average\_value)  
  
 mean\_absolute\_deviation = sum\_of\_values / self.model.n  
  
 **return** mean\_absolute\_deviation  
  
 *# Поиск минимального Х* **def** calculation\_min\_x(self):  
 x = np.amin(self.model.y)  
 **return** x  
  
 *# Поиск максимального Х* **def** calculation\_max\_x(self):  
 x = np.amax(self.model.y)  
 **return** x  
  
 *# Взаимной корреляция* **def** calculation\_nested\_correlation(self, model\_1, model\_2):  
  
 model = Model(9) *# Модель графика взаимной корреляция* y\_list\_1 = copy.deepcopy(model\_1.y)  
 self.calculation\_average\_value()  
 average\_value1 = self.average\_value  
  
 y\_list\_2 = copy.deepcopy(model\_2.y)  
 self.calculation\_average\_value()  
 average\_value2 = self.average\_value  
  
 y = []  
 n = model\_1.n  
 **for** i **in** range(self.l):  
 new\_value = 0  
  
 **for** j **in** range(n-i):  
 new\_value += (y\_list\_1[j] - average\_value1) \* (y\_list\_2[j+ i] - average\_value2)  
 new\_value = new\_value / n  
 y.append(new\_value)  
  
 model.y = np.array(y)  
 model.n = len(model.y)  
 model.x = np.arange(model.n)  
  
 **return** model