Оглавление

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 с 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:

$$S(t) = S_0 * \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right) \quad (1)$$

где:

$$r_k = \frac{S_k - S_{k-1}}{S_{k-1}} \quad (2)$$

$$\mu = \frac{1}{|k|} * \sum r_k \quad (3)$$

$$\sigma = \sqrt{\frac{1}{|k|} * \sum (r_k - \mu)^2} \quad (4)$$

В ходе работы были получены и проанализированы следующие статистические характеристики:

1. Среднее значение $$\mu = \frac{1}{N}\sum_{k=0}^{N-1} y(t)_k \quad (5)$$
2. Дисперсия $$\sigma^2 = \frac{1}{N}\sum_{k=0}^{N-1} (y(t)_k - \mu)^2 \quad (6)$$
3. Стандартное отклонение $$\sigma = \sqrt{\frac{1}{N}\sum_{k=0}^{N-1} (y(t)_k - \mu)^2} \quad (7)$$
4. Коэффициент асимметрии $$\gamma_1 = \frac{\mu_3}{\sigma^3} \quad (8)$$
5. Асимметрия $$\mu_3 = \frac{1}{N}\sum_{k=0}^{N-1} (y(t)_k - \mu)^3 \quad (9)$$
6. Центральный момент четвертого порядка $$\mu_4 = \frac{1}{N}\sum_{k=0}^{N-1} (y(t)_k - \mu)^4 \quad (10)$$
7. Куртозис $$\gamma_2 = \frac{\mu_4}{\sigma^4} - 3 \quad (11)$$
8. Минимальное значение $$\min = \min\left(y(t)_0, \dots, y(t)_{N-1}\right) \quad (12)$$

9. Максимальное значение $\max = \max(y(t)_0, \dots, y(t)_{N-1})$ (13)

А также рассчитана взаимная корреляция:

$$C_{yg}(\tau) = \frac{1}{N}\sum_{k=0}^{N-1}(y(t)_k - \mu_y)(g(t)_k - \mu_g) \quad (14)$$

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.

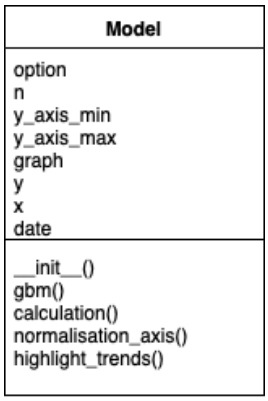

Рисунок 1 – Схема класса Model

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

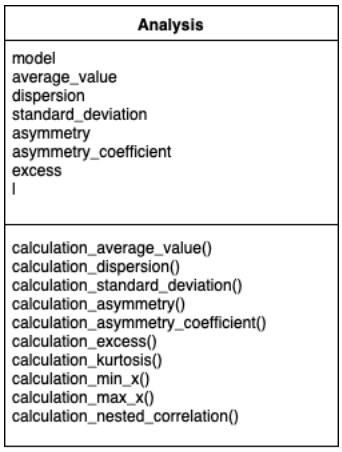

Рисунок 2 – Схема класса Analysis

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

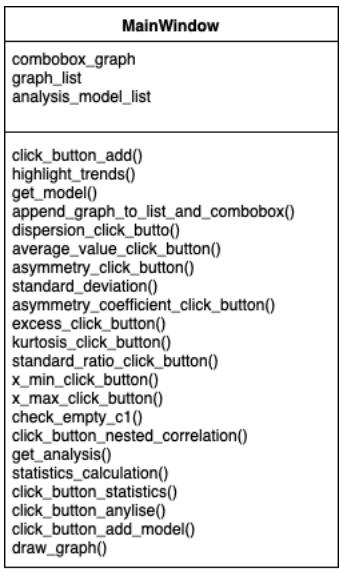

Рисунок 3 – Схема класса MainWindow

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.

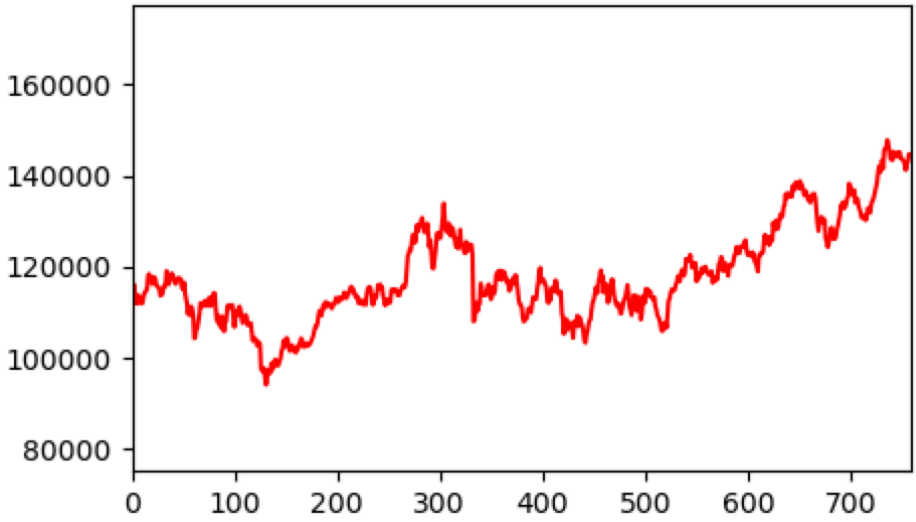

Рисунок 4 – Индекс РТС

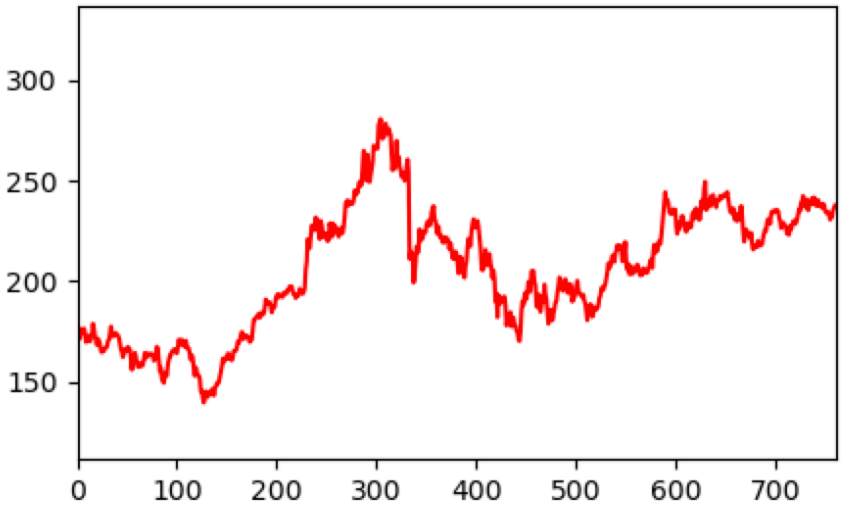

Рисунок 5 – Курс акций Сбербанка

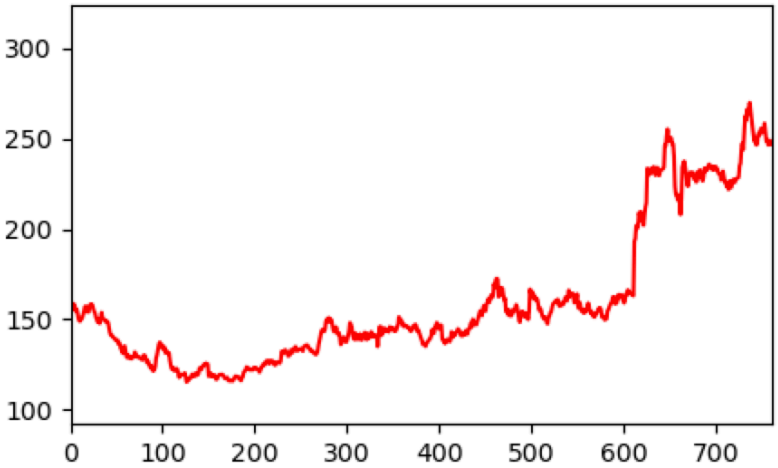

Рисунок 6 – Курс акций Газпрома

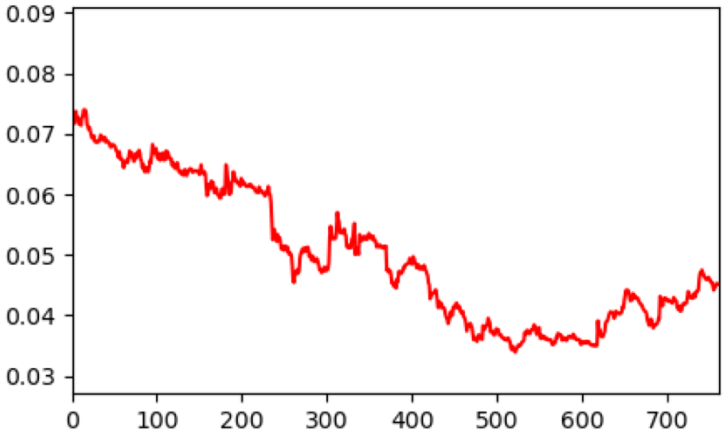

Рисунок 7 – Курс акций ВТБ

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.

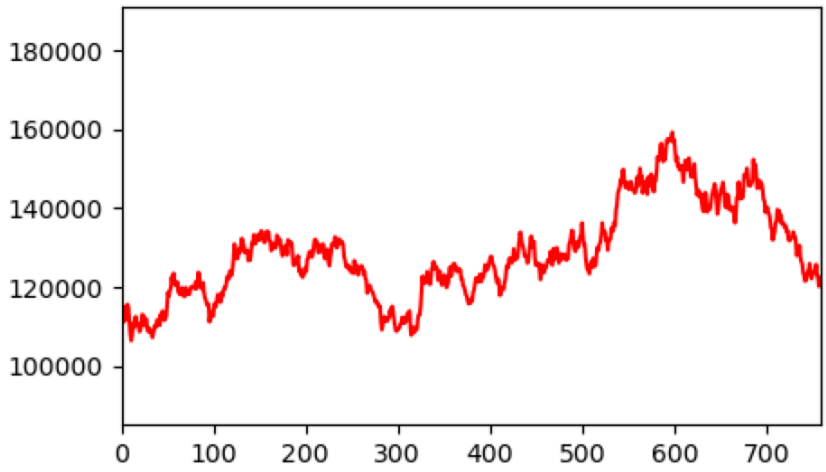

Рисунок 8 – Индекс РТС, смоделированный с помощью GBM

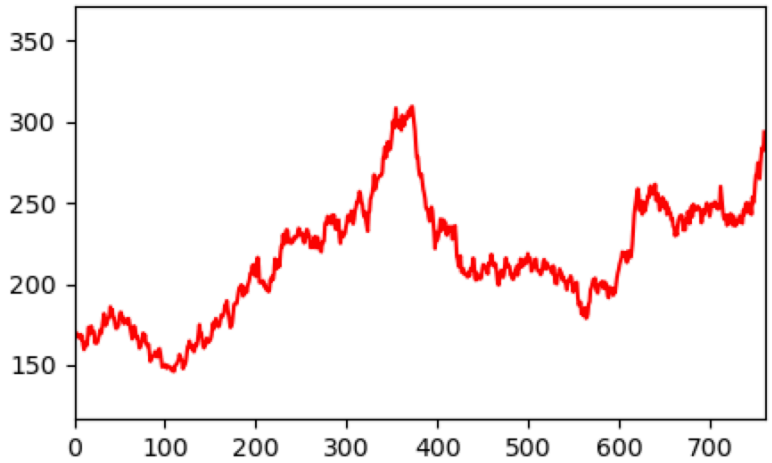

Рисунок 9 – Курс акций Сбербанка, смоделированный с помощью GBM

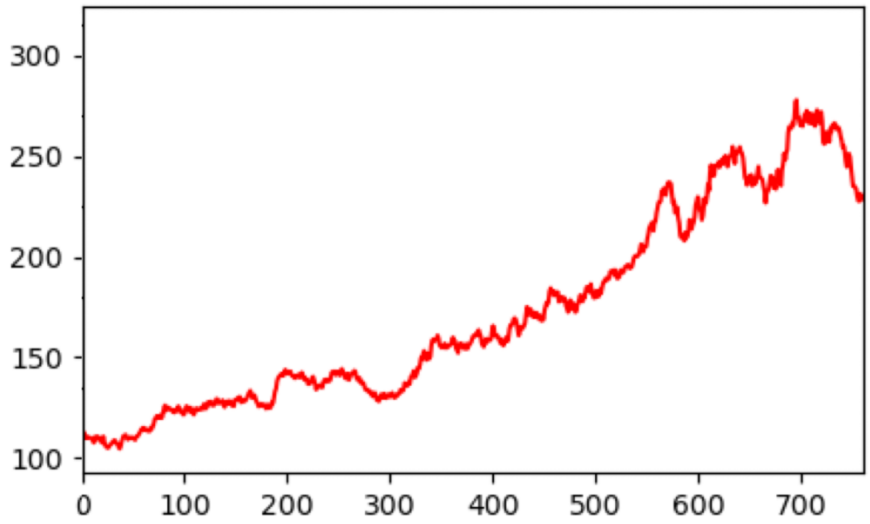

Рисунок 10 – Курс акций Газпрома, смоделированный с помощью GBM

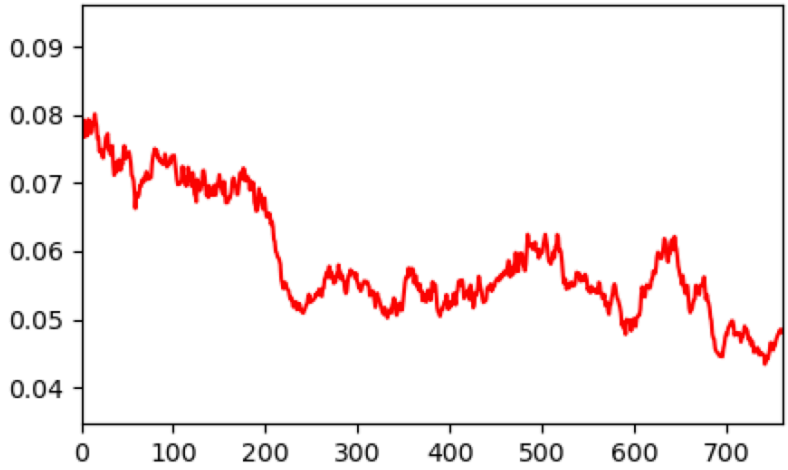

Рисунок 11 – Курс акций ВТБ, смоделированный с помощью GBM

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.

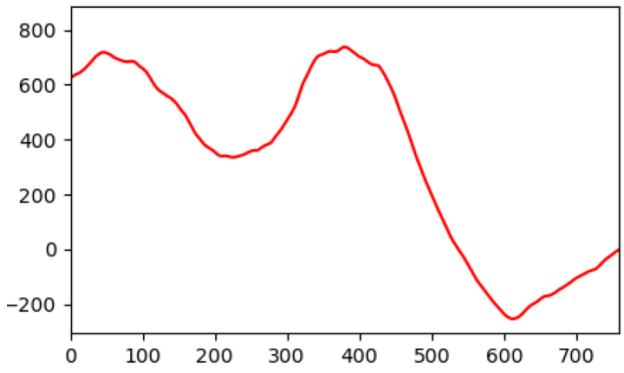

Рисунок 12 – Взаимная корреляция курса акций Сбербанка и курса акций Газпрома

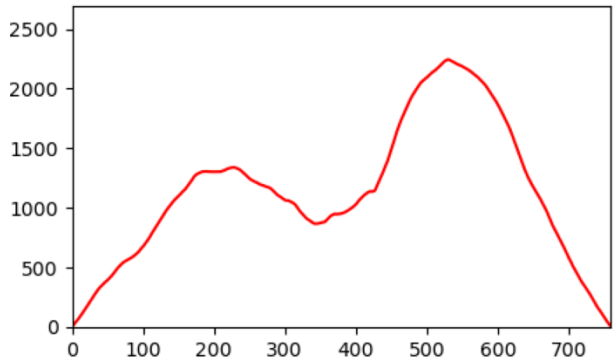

Рисунок 13 – Взаимная корреляция курса акций Сбербанка и курса акций ВТБ

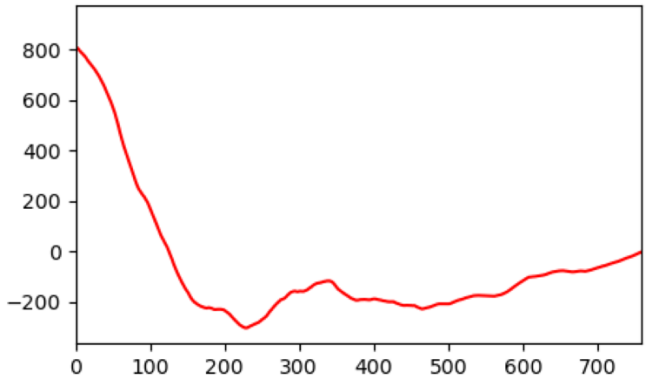

Рисунок 14 – Взаимная корреляция курса акций Сбербанка и смоделированного курса акций Сбербанка

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 X
    def calculation_min_x(self):
        x = np.amin(self.model.y)
        return x

    # Поиск максимального X
    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
```