

Реализация генератора псевдослучайных чисел на основе системы Лоренца с использованием Python

Болтовский Гавриил Александрович

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

В рамках данной работы для разработки генератора псевдослучайных чисел был выбран генератор на основе системы Лоренца – это устройство, которое генерирует хаотические сигналы на основе дифференциальных уравнений динамической системы Лоренца. Система Лоренца, изначально разработанная для моделирования конвекции в жидкостях, демонстрирует чувствительность к начальным условиям – небольшие изменения параметров приводят к радикально разным траекториям. Этот хаотический характер используется для создания последовательности чисел, имитирующих случайность. В результате продемонстрирована программа, визуализирующая аттрактор Лоренца и распределение сгенерированных чисел. Для реализации поставленной задачи был выбран язык программирования Python.

Ключевые слова: Python, псевдослучайный генератор чисел, система Лоренца, PyGame

Implementation of a pseudorandom number generator based on the Lorenz system using Python

Boltovsky Gavriil Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

As part of this work, a generator based on the Lorenz system was chosen to develop a pseudorandom number generator. This device generates chaotic signals based on the differential equations of the Lorenz dynamic system. The Lorenz system, originally developed to simulate convection in fluids, demonstrates sensitivity to initial conditions – small changes in parameters lead to radically different trajectories. This chaotic nature is used to create a sequence of numbers that mimic randomness. As a result, a program was demonstrated that visualizes the Lorenz attractor and the distribution of generated numbers. The Python programming language was chosen to implement the task.

Keywords: Python, pseudo-random number generator, Lorenz system, PyGame

1. Введение

1.1 Актуальность

Генераторы псевдослучайных чисел (ГПСЧ) являются в настоящее время актуальными и их актуальность обусловлена уникальным сочетанием математической необходимости и практической полезности.

Правильно написанный ГСПЧ может быть использован во многих областях человеческой деятельности: криптографии, научных исследованиях, цифровой культуре и даже сфере развлечений. Но реализация такого инструмента зачастую представляет собой достаточно сложную задачу, в первую очередь с математической точки зрения.

Таким образом, разработанный ГСПЧ на корректной математической модели может гарантировать корректность работы на любых входных данных и его использование будет актуальным во многих информационных системах, применяющих алгоритмы генерации псевдослучайных чисел.

1.2 Обзор исследований

Лобов В.А. в своей работе провёл исследование цифровой системы связи с дифференциальным переключением хаотических режимов системы Лоренца [1]. Для исследования был использован Matlab, а в итоге разработана модель цифровой системы связи с дифференциальным переключением хаотических режимов на основе дискретно-нелинейной системы Лоренца.

О.Н. Хатунцева в своей работе провела исследование на тему стохастических свойств динамического хаоса [2]. Исследование показало, что детерминированный хаос, возникающий в САДУ типа Лоренца, при задании любого конечного фиксированного шага по времени, вполне может быть ассоциирован со стохастическим процессом, а не являться, по сути, детерминированным хаосом.

М.А. Русанов и И.Д. Котилевец рассмотрели проблема использования хаотических систем в качестве основы при проектировании новых криптосистем [3]. Также в статье проводится сравнение систем классической криптографии и физических хаотических систем, а также на основе системы дифференциальных уравнений Лоренца представлен новый алгоритм шифрования, написанный на языке программирования `njs`.

В.А. Лобов описал моделирование блоков системы передачи информации на основе метода переключения хаотических режимов дискретно-нелинейной системы Лоренца [4]. В базовом пакете программ Matlab была разработана система передачи информации на основе метода переключения хаотических режимов системы Лоренца.

1.3 Цель исследования

Цель – разработать генератор псевдослучайных чисел на основе системы Лоренца и визуально продемонстрировать его работоспособность с помощью аттрактора Лоренца и гистограммы распределения сгенерированных чисел.

2 Материалы и методы

Для решения поставленной цели выбран язык программирования Python, а также дополнительные модули PyGame и NumPy.

3 Результаты и обсуждения

В первую очередь необходимо понять принцип работы системы Лоренца, для того чтобы можно было реализовать генератор псевдослучайных чисел на её основе.

Система Лоренца – это система из трёх нелинейных дифференциальных уравнений, впервые предложенная Эдвардом Лоренцем в 1963 году для упрощённого математического описания атмосферной конвекции. Эта система стала одной из наиболее известных в теории хаоса и демонстрирует сложное, хаотическое поведение при определённых значениях параметров.

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases}$$

Рисунок 1. Математическое описание системы Лоренца

В уравнениях системы Лоренца задействованы следующие переменные:

1. x, y, z – переменные, зависящие от времени t .
2. σ – число Прандтля, характеризующее соотношение вязкости и теплопроводности.
3. ρ – число Рэлея, описывающее меру разности температур между верхним и нижним слоями жидкости или газа.
4. β – геометрический параметр, связанный с размерами области.

Траектории системы стремятся к аттрактору Лоренца – фрактальному множеству в фазовом пространстве, на котором движение никогда не повторяется, но остаётся ограниченным.

В теории динамических систем аттрактор – это множество состояний, к которому стремится система со временем, независимо от начальных условий (в пределах некоторой области притяжения). Можно выделить 2 вида аттракторов, в зависимости от формы и регулярности: регулярные (например, маятник, останавливающийся в нижней точке) и странные (они же хаотические). Аттрактор Лоренца относится к хаотическим, потому что незначительное изменение изначальных параметров приводит к уникальному поведению системы. При этом все траектории на аттракторе Лоренца остаются в конечной области пространства и не уходят на бесконечность.

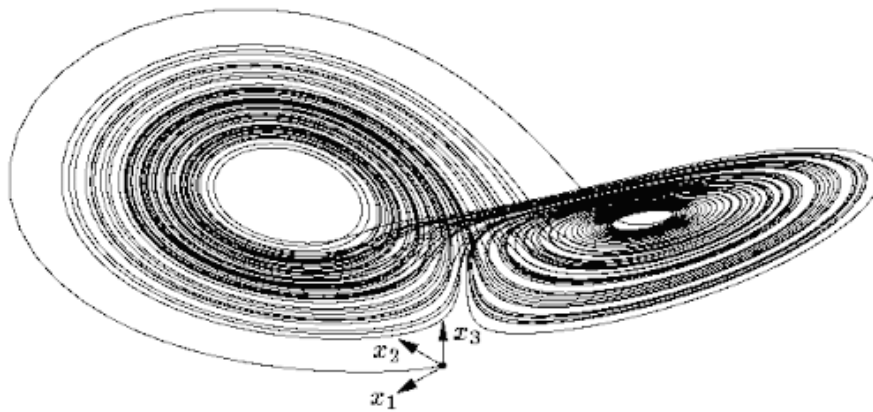


Рисунок 2. Пример аттрактора Лоренца

Система Лоренца возникла в физике и метеорологии, но также имеет значение и для программирования – как с практической, так и с концептуальной точки зрения.

Реализовать аттрактор Лоренца возможно на любом современном языке программирования, например на Python. Для визуализации аттрактора можно задействовать сторонние модули, такие как Scipy и Matplotlib. Была написана программа на Python, в которой реализована функция вычисления системы Лоренца:

```
def lorenz(t, state, sigma=10, rho=28, beta=8/3):  
    x, y, z = state  
    dx = sigma * (y - x)  
    dy = x * (rho - z) - y  
    dz = x * y - beta * z  
    return [dx, dy, dz]
```

Полученные значения затем можно использовать для построения соответствующего аттрактора (см. рис. 3). Для реализации были также заданы стартовые значения, характеризующие начальные условия. Все они сохранены в переменные `initial`, `t_span` и `t_eval`:

```
initial = [1.0, 1.0, 1.0]  
t_span = (0, 50)  
t_eval = np.linspace(0, 50, 10000)  
  
sol = solve_ivp(lorenz, t_span, initial, t_eval=t_eval)  
  
fig = plt.figure(figsize=(10, 7))  
ax = fig.add_subplot(111, projection='3d')  
ax.plot(sol.y[0], sol.y[1], sol.y[2], lw=0.5)  
ax.set_title("Аттрактор Лоренца")  
plt.show()
```

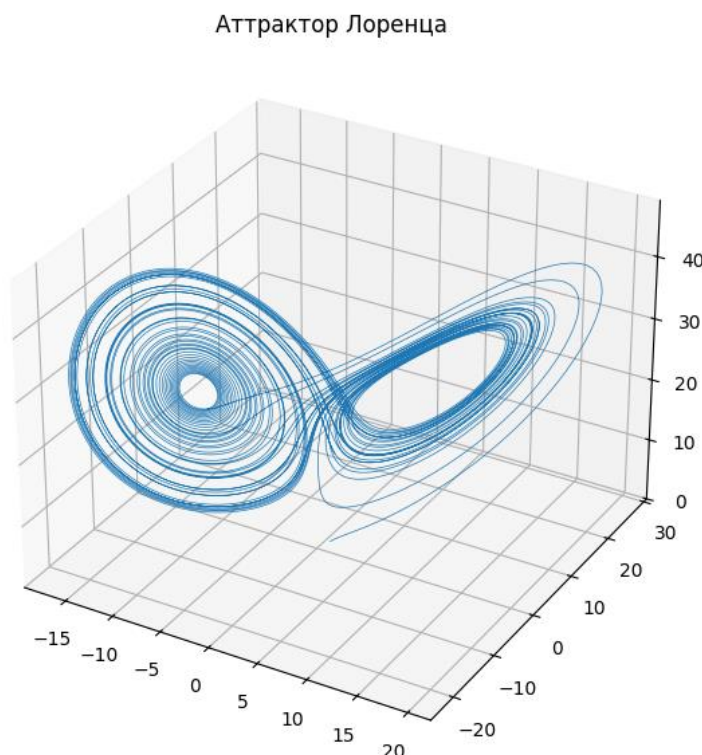


Рисунок 3. Пример построения аттрактора Лоренца с помощью Python

Незначительное изменение начальных условий приведёт к совершенно другому результату формирования аттрактора, что в свою очередь можно использовать для генерации псевдослучайных чисел.

Для проверки данной гипотезы была разработана программа на Python с использованием модуля PyGame (для реализации возможности взаимодействия с программой с помощью клавиатуры), в которой представлена возможность изменения переменных системы Лоренца с дальнейшей демонстрацией последовательности чисел, сформированных алгоритмом, а также построением аттрактора (см. рис. 4).

В программе реализованы три основных класса: LorenzRNG, App и Stats.

Основу генератора составляет класс LorenzRNG, который численно решает систему дифференциальных уравнений Лоренца. На каждом шаге обновляются три переменные (x , y , z), описывающие состояние системы. Для получения псевдослучайного значения используется дробная часть одной из координат, циклически выбираемой из x , y , z . Для улучшения равномерности распределения к значению добавляется небольшая поправка, зависящая от номера шага. При инициализации класса задаются ключевые параметры системы: начальное значение $seed$, определяющее отправную точку траектории в фазовом пространстве, а также классические параметры аттрактора – σ (число Прандтля), ρ (число Рэлея) и β (геометрический коэффициент). Эти параметры напрямую контролируют режим работы системы: при стандартных значениях ($\sigma = 10$, $\rho = 28$, $\beta \approx 8/3$) система демонстрирует хаотическое поведение, тогда как их изменение может

привести к переходу в устойчивые стационарные состояния или предельные циклы, что критически влияет на качество генерируемых чисел.

После инициализации начальных координат (x, y, z), слегка смещённых относительно seed для избежания симметрии, выполняется 1000 итераций метода `_step()`. Этот этап необходим, так как траектория системы Лоренца сходится к странному аттрактору не мгновенно: начальные точки могут лежать вне области хаотической динамики, что исказило бы статистические свойства последовательности. Такой подход позволяет обеспечить стабильность распределения точек на аттракторе.

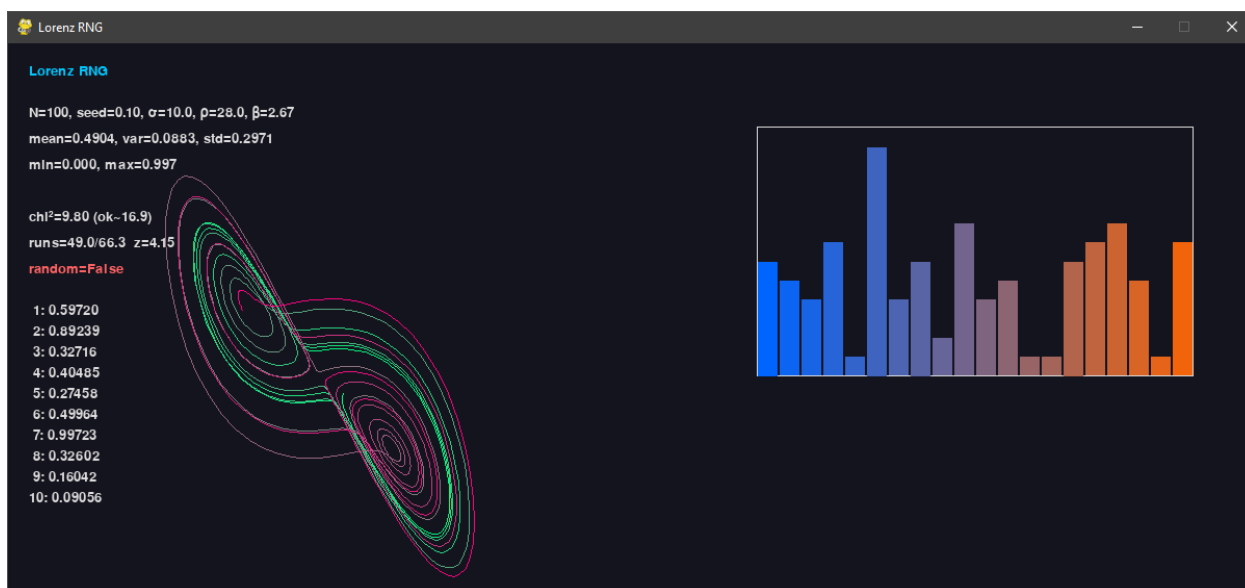


Рисунок 4. Интерфейс генератора псевдослучайных чисел

Для верификации гипотезы о равномерности распределения реализован метод `chi2()`, выполняющий критерий согласия Пирсона (хи-квадрат). Алгоритм разбивает интервал $[0, 1)$ на заданное количество значений (по умолчанию 10), подсчитывает частоту попаданий значений в каждый интервал и сравнивает наблюдаемые частоты с теоретическими ожиданиями для равномерного распределения (общее количество элементов, делённое на количество значений):

```
def chi2(nums, bins=10):
    if len(nums) < bins * 5: return 0
    hist, _ = np.histogram(nums, bins=bins, range=(0, 1))
    exp = len(nums) / bins
    return np.sum((hist - exp) ** 2 / exp)
```

Ключевой метод для оценки независимости элементов последовательности – `runs()`, реализующий тест серий (runs test). Этот тест выявляет скрытые корреляции между последовательными значениями, что особенно важно для хаотических генераторов, где детерминированная природа системы может создавать ложные закономерности. Алгоритм работает следующим образом:

```
def runs(nums):
    if len(nums) < 20: return {'z': 0, 'ok': False}
    m, runs, last = np.mean(nums), 1, nums[0] > np.mean(nums)
    for v in nums[1:]:
        cur = v > m
        if cur != last: runs += 1
        last = cur
    n = len(nums)
    exp = (2 * n - 1) / 3
    var = (16 * n - 29) / 90
    z = abs(runs - exp) / math.sqrt(var)
    return {'runs': runs, 'exp': exp, 'z': z, 'ok': z < 1.96}
```

Управление осуществляется через клавиатуру: пробел перезапускает генератор со случайным начальным значением, цифровые клавиши изменяют объём выборки, а буквы (s, d, z, x и др.) корректируют параметры системы: клавиши s и d меняют значение seed (номер генерации) на 0.1 (увеличивают и уменьшают), клавиши z и x меняют коэффициент σ на 0.5, клавиши с и v меняют коэффициент ρ на 1, клавиши b и n меняют коэффициент β на 0.1.

В верхней части окна отображаются текущие значения параметров, ниже – сводная статистика, а в самом низу выводятся первые десять сгенерированных чисел с фиксированной точностью, что позволяет проводить ручную проверку результатов.

Для проверки работоспособности программы было создано 5 наборов псевдослучайных чисел. Стандартные значения sigma, rho и beta оставались неизменными, но в переменную seed сохранялось случайное число от 1 до 10 (с плавающей точкой). Результаты продемонстрировали разные значения при незначительных изменениях переменной seed (см. рис. 5). Но также нужно отметить и воспроизводимость – при полностью одинаковых значениях переменных программа создаёт совпадающие наборы, что тоже характеризует систему Лоренца.

1: 0.4570784657006144	1: 0.7334486499871169	1: 0.6547490462508032	1: 0.004439156020496824	1: 0.626888040976646
2: 0.743764462926899	2: 0.13763344214268336	2: 0.6261352467303575	2: 0.21347357076649576	2: 0.42126004345667617
3: 0.890873167199127	3: 0.5435773367156984	3: 0.9062446135798816	3: 0.9612675504273818	3: 0.04954659406517914
4: 0.9975930778912875	4: 0.39316731828895535	4: 0.6446025741542015	4: 0.7937491183990942	4: 0.5749733932157841
5: 0.7580752489404572	5: 0.7674429937495366	5: 0.9804813905714427	5: 0.22491184161299294	5: 0.6013006668271036
6: 0.09208780912472248	6: 0.9774830606633514	6: 0.03310755827897648	6: 0.49659938699252704	6: 0.732294181023731
7: 0.6691284641493249	7: 0.04576434839831477	7: 0.4613607862814364	7: 0.6442518181866967	7: 0.5861210597090631
8: 0.12198905204207265	8: 0.5810579010078826	8: 0.1921312675976271	8: 0.3773758088454141	8: 0.7949644355885123
9: 0.2339696297181979	9: 0.7324686199754622	9: 0.3811177870218556	9: 0.005484534477309788	9: 0.5212764345878773
10: 0.5304712783493891	10: 0.742653092914442	10: 0.11324396829522586	10: 0.5774173785141564	10: 0.6470571132403473

Рисунок 5. Результаты генерации пяти наборов псевдослучайных чисел

Также существуют математические способы верификации работы написанного генератора псевдослучайных чисел. Один из способов верификации предполагает сравнение траектории, построенной программой, с результатом решения системы методом Рунге-Кутты 4-го порядка. Для решения системы предложенным методом был написан отдельный алгоритм:

```

def rk4_step(state, dt, sigma=SIGMA, rho=RHO, beta=BETA):
    def derivatives(s):
        x, y, z = s
        return np.array([
            sigma * (y - x),
            x * (rho - z) - y,
            x * y - beta * z
        ])

    k1 = derivatives(state)
    k2 = derivatives(state + 0.5 * dt * k1)
    k3 = derivatives(state + 0.5 * dt * k2)
    k4 = derivatives(state + dt * k3)

    return state + (dt / 6) * (k1 + 2*k2 + 2*k3 + k4)

```

Также была написана отдельная программа, сравнивающая результаты генерации псевдослучайных чисел с результатами решения системы методом Рунге-Кутты 4-го порядка.

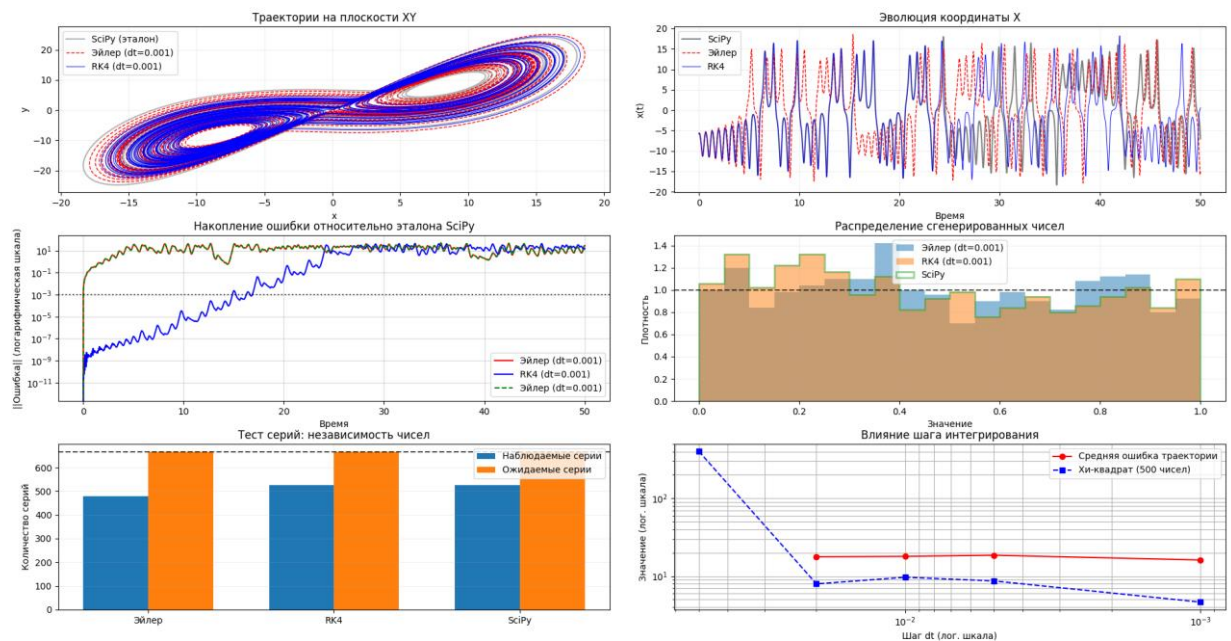


Рисунок 6. Сравнение результатов генерации псевдослучайных чисел методом Эйлера с методом Рунге-Кутты

Можно отметить, что метод Эйлера, лежащий в основе разработанной программы генератора псевдослучайных чисел, способен обеспечить высококачественные результаты при условии использования достаточно малого шага интегрирования, сохранённый в переменной $dt = 0.001$. При таком шаге траектория системы Лоренца воспроизводит аттрактор с высокой детализацией, сохраняя его характерную фрактальную структуру, а сгенерированные числа демонстрируют статистические свойства, близкие к эталонным: равномерное распределение подтверждается критерием хи-

квадрат (значение примерно равное 5 при критическом 16.92), а тест серий подтверждает статистическую независимость последовательных значений.

Однако вычислительная цена такого качества существенна: уменьшение шага в 10 раз увеличивает время расчёта также в 10 раз. В этом контексте метод Рунге-Кутты 4-го порядка (RK4) с шагом $dt=0.01$ представляет собой оптимальный компромисс, обеспечивая точность, сопоставимую с методом Эйлера при $dt=0.001$, но при значительно меньших вычислительных затратах.

С практической точки зрения, система Лоренца с правильно настроенными параметрами может служить надёжным источником псевдослучайных чисел для некриптографических задач: моделирования методом Монте-Карло, процедурной генерации контента в играх, визуализации сложных систем. Однако для криптографических применений требуется дополнительная постобработка последовательностей или переход к более сложным хаотическим системам с гарантированными свойствами непредсказуемости. Особенно важно отметить, что качество генератора напрямую зависит от баланса между точностью численного метода и вычислительной эффективностью: при $dt=0.001$ метод Эйлера обеспечивает приемлемое качество.

Библиографический список

1. Лобов В.А. Математическое моделирование цифровой системы связи с дифференциальным переключением хаотических режимов системы Лоренца // Прикладная математика и информатика: современные исследования в области естественных и технических наук: Материалы V Международной научно-практической конференции (школы-семинара) молодых ученых, Тольятти, 22–24 апреля 2019 года. – Тольятти: Издатель Качалин Александр Васильевич. - 2019. - С. 249-254.
2. Хатунцева О.Н. О стохастических свойствах динамического хаоса в системах автономных дифференциальных уравнений, типа системы Лоренца // Труды МАИ. 2020. № 112. С. 1.
3. Русанов М.А., Котилевец И.Д. Криптографическая система на основе хаотической системы Лоренца // Хроники цифровых трансформаций: Материалы межкафедрального круглого стола, Москва, 19–26 мая 2022 года. Том Выпуск 4. Волгоград: ИП ЧЕРНЯЕВА ЮЛИЯ ИГОРЕВНА (Издательский дом "Сириус"). 2022. С. 128-133.
4. Лобов В.А. Моделирование блоков системы передачи информации на основе метода переключения хаотических режимов дискретно-нелинейной системы Лоренца // XXIV Туполевские чтения (школа молодых ученых) : Материалы Международной молодёжной научной конференции. В 6-ти томах, Казань, 07–08 ноября 2019 года. Том V. Казань: ИП Сагиева А.Р.. 2019. С. 379-383.