Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Санкт-Петербургский государственный университет»
(СПбГУ)

УДК 004.896
Рег. № НИОКТР 121041500280–4

УТВЕРЖДАЮ
Проректор по научной работе
_____ С. В. Микушев
«19» декабря 2022 г.

ОТЧЁТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

МОДЕЛИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА С МАЛЫМ КОЛИЧЕСТВОМ ОБУЧАЮЩИХ ДАННЫХ В ЗАДАЧАХ ПРОМЫШЛЕННОГО ИНТЕРНЕТА ВЕЩЕЙ (IIOT)

(промежуточный, этап 2)

Руководитель НИР,
доцент, к.ф.-м.н. _____ А. Г. Головкина
24.11.2022

Санкт-Петербург 2022

СПИСОК ИСПОЛНИТЕЛЕЙ

Руководитель НИР: Доцент, к.ф.-м.н.	24.11.2022 (подпись, дата)	А. Г. Головкина (введение, заключение, раздел 3, раздел 4)
Отв. исполнитель: Доцент, к.ф.-м.н.	24.11.2022 (подпись, дата)	В. А. Козынченко (раздел 2, раздел 3)
Исполнители: Инженер-исследователь	24 ноября 2022 г. (подпись, дата)	В. О. Ружников (раздел 3)
Инженер-исследователь	24.11.2022 (подпись, дата)	И. С. Клименко (раздел 3)
Инженер-исследователь	24.11.2022 (подпись, дата)	Д. Д. Ганаева (раздел 1, раздел 4)
Инженер-исследователь	24.11.2022 (подпись, дата)	Н. А. Князев (раздел 2)
Нормоконтроль		**Н.В. Квадрицус**

30.11.2022

РЕФЕРАТ

Отчет на 74 стр., 25 рис., 47 источников, 4 таблиц

ПРЕОБРАЗОВАНИЕ ЛИ, ИДЕНТИФИКАЦИЯ ДИНАМИЧЕСКИХ СИСТЕМ, ПОЛИНОМИАЛЬНЫЕ НЕЙРОННЫЕ СЕТИ, ПРЕДСКАЗАНИЕ ВРЕМЕННЫХ РЯДОВ, МОДЕЛИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА, ОБУЧЕНИЕ НА МАЛЫХ ДАННЫХ, ПРОМЫШЛЕННЫЙ ИНТЕРНЕТ ВЕЩЕЙ

Объектом исследования являются искусственные нейронные сети для моделирования динамических систем, которые могут быть описаны системой нелинейных дифференциальных уравнений.

Целью второго этапа научно-исследовательской работы является улучшение архитектуры и усовершенствование алгоритмов обучения нейронной сети на основе нелинейного матричного преобразования оператора Ли для случая автономных и неавтономных нелинейных систем с полиномиальными нелинейностями и демонстрация результатов работы на практических примерах.

В результате исследования была разработана архитектура и алгоритм обучения нейронной сети на основе нелинейного матричного преобразования оператора Ли для случая неавтономных нелинейных систем дифференциальных уравнений с полиномиальными нелинейностями. Разработан алгоритм реконструкции неавтономного дифференциального уравнения по данным измерений в случае, если вид уравнения заранее неизвестен, работающий в том числе для случая нерегулярного распределения данных. Результаты работы проиллюстрированы на модельных задачах из теории нелинейных систем: нелинейный дефлектор, система Лотки-Вольтерры, а также на реальном наборе данных измерений трафика в телекоммуникационной сети. Получены аналитические оценки точности приближенного решения. Также оценены необходимый для достижения заданной точности порядок полиномиального разложения и шаг дискретизации данных. Описанные алгоритмы были реализованы в виде библиотеки с открытым исходным кодом tmflow.

Рекомендации по внедрению или итоги внедрения результатов НИР: На основе полученных результатов можно сделать положительный вывод о потенциале рассматриваемого метода и возможности его внедрения в матобеспечение контроллеров с прогнозирующими моделями. Данный вывод также подтверждается успешно завершившимся первым этапом прикладного проекта с индустриальным заказчиком, заключавшимся в построении прогнозной модели на основе преобразования Ли для определения содержания NOx соединений в выхлопных газах дизельного двигателя после селективного каталитического восстановления.

Прогнозные предположения о развитии объекта исследования: Дальнейшим направлением развития данных исследований станет обобщение построенных алгоритмов на случай систем дифференциальных уравнений с параметрами, а также с запаздывающим аргументом. Таким образом, с помощью методов построения отображений связь между нейронными сетями и динамическими системами в задачах анализа и управления сложными объектами будет исследована комплексно.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Промышленный интернет вещей (IIoT) это многоуровневая система, включающая в себя датчики и контроллеры, установленные на узлах и агрегатах промышленного объекта, средства передачи, хранения и визуализации данных [1]. Однако истинная ценность IIoT заключается в данных, анализ которых может помочь улучшить ключевые параметры бизнеса, такие как работоспособность оборудования, сохранение экономических ресурсов и эффективность процессов. Тем не менее, более 99% этих данных либо не используются, отправляясь в облако для пакетной обработки и генерирования отчетов, либо анализируются базовыми статистическими методами [2,3].

В свою очередь извлечение аналитической информации из данных IIoT в реальном времени требует разработки эффективного инструмента построения предиктивных моделей рассматриваемых процессов, к которым при этом предъявляется ряд требований к производительности, качеству (физичности) и обобщающей способности [4]. В основном предиктивные модели на основе данных измерений строятся, опираясь либо на теорию идентификации математических моделей, либо на современные методы машинного обучения.

Идентификация линейных моделей является теоретически хорошо проработанной и изученной областью. В этом случае чаще всего осуществляется переход из временной области в частотную, что позволяет свести дифференциальную зависимость к алгебраической и использовать для описания линейной системы передаточную функцию с известным порядком нулей и полюсов. Процесс идентификации при этом сводится к нахождению коэффициентов характеристических полиномов числителя и знаменателя. Этим объясняется тот факт, что широкое распространение в промышленности получили именно линейные модели, а решение задачи регулирования нелинейными явлениями возлагается на промышленные контроллеры.

Однако в силу сложности конструкций промышленных устройств и протекающими нелинейными процессами такими как, процессы теплообмена, течения вязких жидкостей, волновые явления, химические реакции и др. регулирование реально функционирующих системы на основе линейной модели производится не только не оптимально, но даже в режиме ручной подрегулировки настроек управляющих элементов.

В этом случае могут применяться нелинейные модели, для построения которых используется один из трех способов: использование стохастического моделирования, вывод эволюционных уравнений, исходя из физических свойств всех протекающих в технических системах явлений или же получение нелинейной модели с сосредоточенными параметрами, с последующей идентификацией ее параметров. В силу сложности

моделирования реальных функционирующих систем третий способ получил наибольшее распространение.

Стоит отметить, что качество идентификации модели зависит не только от определения неизвестных параметров, но и от выбора структуры. Для нахождения параметров разработаны различные математические процедуры, а оценке структуры уделяется меньше внимания. Одним из распространенных способов учета нелинейных эффектов являются блочно-ориентированные модели, например модель Гаммерштейна, (представляющая собой последовательное соединение статического нелинейного блока и линейного динамического блока) или модель Винера (представляющая собой последовательное соединение линейного динамического блока и статического нелинейного блока). Однако свойства нелинейности и динамичности объектов управления в ряде случаев невозможно четко разделить. При таких условиях нелинейный динамический объект представляют в виде некоторой комбинации линейных динамических и статических нелинейных блоков, но структурная идентификация такой системы является затруднительной.

Современной альтернативой классическим методам математического моделирования является машинное обучение. Методы машинного обучения для прогнозного моделирования также можно разделить на линейные (например, линейная регрессия, метод опорных векторов, логистическая регрессия) и нелинейные (например, решающие деревья, нелинейная регрессия, бустинг, нейронные сети). Линейные модели дают качественное понимание взаимосвязи между входными и выходными значениями, и могут эффективно использовать в случае линейной корреляции между ними. В противном случае исследователи обращаются к нелинейным алгоритмам, которые в первую очередь стремятся к минимизации отклонения между реальным и прогнозным значением, зачастую жертвуя объяснимостью модели. В связи с этим, их часто называют моделями, работающими по принципу черного ящика, т.к. при обучении используется большое количество данных и не гарантируется математическая строгость результата. Разные коэффициенты и параметры, задающие предсказательную модель, не имеют физической интерпретации. Это представляет собой серьезное препятствие для внедрения машинного обучения в отрасли автоматических систем управления.

В настоящее время появился ряд исследований, направленных на создание методов для объяснения или интерпретации вывода моделей черного ящика, связывая с ними надежды на повышение доверия к искусственному интеллекту и его внедрению в промышленные системы управления (advanced process control). Тем не менее, по мнению многих авторов [5], наиболее перспективный путь развития все же заключается в

разработке моделей искусственного интеллекта, которые являются интерпретируемыми по своей природе.

Одним из рассмотренных в литературе подходов, устанавливающих связь между теорией динамических систем и искусственными нейронными сетями, являются так называемые «нейронные обыкновенные дифференциальные уравнения (Neural ODE)», где обучение рекуррентной нейронной сети предлагается проводить с помощью численного решения соответствующего обыкновенного дифференциального уравнения. После выхода статьи [6], появились работы развивающие полученные авторами результаты для решения задач моделирования и управления гамильтоновыми и лагранжевыми системами [7–12].

Другие подходы учета физических свойств при проектировании архитектуры и обучении нейронной сети связаны с использованием численных решателей дифференциальных уравнений (в обыкновенных и частных производных) [13–15] или же регуляризационных членов, включающих в себя физические симметрии и инварианты объекта моделирования или соотношения между дифференциальными операторами, применяемые к функции потерь [16–20].

В выполняемой научно-исследовательской работе предлагается решение, которое заключается в объединении математических методов дифференциальной алгебры и нейросетевых моделей. А именно, рассматривается метод построения нелинейного матричного преобразования Ли, основанный на эволюции отображения для решения нелинейных систем обыкновенных дифференциальных уравнений. Вычисление оператора Ли в матричной форме на основе степенных рядов позволяет строить отображение, описывающее динамику системы за конечный интервал времени. В результате чего решение дифференциального уравнения описывается отображением в полиномиальной форме по кронекеровским степеням начального состояния фазовых переменных, а коэффициенты инкапсулируют информацию о динамических характеристиках системы. Точность аппроксимации контролируется порядком нелинейности отображения и шагом дискретизации.

Для построения интеллектуальных систем управления на основе измеренных данных важным оказывается тот факт, что отображение Ли может использоваться не только как эффективный численный метод исследования динамических систем, но и как нейросетевое представление динамической системы. Этот метод позволяет объединить присущие искусственным нейронным сетям параллельные вычислительные архитектуру со строгой теорией динамических систем и теорией дифференциальных уравнений.

Целью второго этапа научно-исследовательской работы является улучшение архитектуры и усовершенствование алгоритмов обучения нейронной сети на основе

нелинейного матричного преобразования оператора Ли для случая автономных и неавтономных нелинейных систем с полиномиальными нелинейностями и демонстрация результатов работы на практических примерах. Для достижения поставленной цели необходимо решить ряд ***задач***:

1. Усовершенствовать реализацию разработанных численных алгоритмов в рамках первого этапа НИР для случая произвольной размерности вектора фазовых переменных.
2. Разработать архитектуру полиномиальной нейронной сети на основе нелинейного матричного преобразования оператора Ли для случая произвольных нелинейных неавтономных систем с полиномиальными нелинейностями.
3. Разработать алгоритм инициализации весовых коэффициентов нейронной сети (п. 2) с использованием результатов, полученных в рамках первого этапа НИР
4. Определить требования к количеству, дискретности данных и порядку нелинейности для обеспечения требуемой точности прогнозирования с использованием полиномиальной нейронной сети.
5. Разработать алгоритм реконструкции дифференциального уравнения по данным временных рядов (в случае, если вид уравнения заранее неизвестен, а в данных присутствует шум и различные статистические возмущения).
6. Построить оценку точности нейросетевой аппроксимации
7. Разработать алгоритм обучения построенной нейронной сети на малых данных
8. Продемонстрировать разработанные алгоритмы на классических модельных задачах из теории нелинейных систем: нелинейный дефлектор, система Лотки-Вольтерры, а также на реальных данных.
9. Реализовать описанные алгоритмы в виде библиотеки с открытым исходным кодом с использованием набора библиотек TensorFlow.

Второй этап научно-исследовательской работы продолжает и углубляет результаты, полученные на предыдущем этапе, расширяя возможности применения методики на случай произвольных нестационарных систем. Таким образом, с помощью методов построения отображений связь между нейронными сетями и динамическими системами в задачах анализа и управления сложными объектами будет исследована комплексно.

Настоящий отчет имеет следующую структуру: во *введении* сформулированы цель и задачи второго этапа исследования, кратко описаны используемые подходы, выполнена оценка современного состояния решаемой научно-технической проблемы и дано обоснование актуальности и необходимости проведения НИР. *Раздел 1* содержит описание

метода реконструкции дифференциальных уравнений по временным рядам [21,22]. *Раздел 2* посвящен теоретическому обоснованию и описанию алгоритмов построения предложенной нейронной сети на основе нелинейного преобразования Ли, а устанавливает связь между полиномиальной архитектурой и теорией динамических систем, а также содержит аналитическую оценку точности приближенного решения [23,24]. В *разделе 3* описаны особенности реализации разработанной архитектуры рекуррентной полиномиальной нейронной сети, а также представлен алгоритм инициализации весовых коэффициентов и обучения [25]. В *разделе 4* содержится иллюстрация разработанных алгоритмов на различных примерах. В частности, рассмотрены модельные задачи, такие как: нелинейный дефлектор [23], уравнения Лотки-Вольтерры [24], для которых продемонстрированы преимущества предлагаемого метода перед классическими нейросетевыми архитектурами в задаче идентификации модели по некоторому частному решению, выступающему в данном случае в качестве измерений. Кроме того, показаны результаты применения предложенной методики на наборе реальных данных, соответствующих временным рядам трафика в сети [26]. В *заключении* кратко формулируются основные результаты работы и даны рекомендации по их внедрению.

1 РЕКОНСТРУКЦИЯ ДИНАМИЧЕСКОЙ СИСТЕМЫ ПО ВРЕМЕННЫМ РЯДАМ

Одним из способов структурно-параметрической идентификации нелинейных систем является приближенное представление неизвестной правой части системы дифференциальных уравнений в виде разложения по базисным функциям. В этом случае первым шагом является определение подходящей системы базисных функций, а вторым – определение неизвестных коэффициентов при них [27].

Многие существующие методы успешно справляются с реконструкцией динамики системы в идеальных условиях, то есть в условиях, когда данные во временном ряду расположены часто, равномерно и не имеют шумов [28–32]. Однако они часто не справляются с задачей идентификации динамической системы в том случае, когда временной ряд измерений является неравномерным (или нерегулярным, то есть измерения расположены через неравные промежутки времени) или редким. Это происходит, потому что нахождение коэффициентов на шаге параметрической идентификации базируется на аппроксимирующей функции, а распространенные методы аппроксимации хорошо работают с данными, расположенными часто и равномерно. К тому же, часто в методах требуется численно аппроксимировать значения производной, стоящей в левой части системы, и для этого применяется метод конечных разностей, погрешность которого также увеличивается при увеличении расстояния между имеющимися измерениями.

Так как известно, что любую аналитическую функцию с заданной степенью точности можно представить в виде полиномиального разложения [33], то было решено искать неизвестную правую часть системы дифференциальных уравнений в виде комбинации полиномиальных базисных функций.

Стоит отметить, что часто реальные наборы данных помимо динамической составляющей содержат шум и статистические возмущения, поэтому для идентификации динамики такой системы необходимо отделить основной сигнал, включающий в себя тренд и периодичность, от шумов и прочих возмущений. В качестве инструмента идентификации структуры и разделения временного ряда на компоненты в данной работе используется метод на основе анализа сингулярного спектра (Singular Spectrum Analysis, SSA) [34].

Таким образом, реконструкция динамики выполняется по выделенному основному сигналу, а задача прогнозирования разделяется на прогнозирование динамической и статистической составляющей с последующем суммированием результатов.

Прогнозирования динамической составляющей временного ряда с использованием реконструированной динамической системы опирается на подходы, изложенные в Главах 2 и 3 настоящего отчета. Для прогнозирования остаточной компоненты временного ряда

могут быть использованы различные статистические модели, например, авторегрессионные модели, экспоненциальное сглаживание и др. [35,36].

1.1 Обработка данных временного ряда

1.1.1. Декомпозиция данных

Как было отмечено выше, в качестве метода декомпозиции временного ряда был использован метод сингулярного спектрального анализа SSA, являющимся обобщением метода главных компонент (Principal Component Analysis, PCA) для выделения линейного подпространства. Данный метод основан на преобразовании исходного одномерного временного ряда в последовательность многомерных векторов (по сути, представляющих собой матрицу, содержащую фрагменты временного ряда, полученные с некоторым сдвигом) с последующим сингулярным разложением полученной матрицы. Таким образом, базовый алгоритм состоит из двух дополняющих друг друга этапов, разложения и восстановления [37].

Этап 1: разложение

Шаг 1. Вложение

Пусть L – некоторое целое число (длина окна), $1 < L < M$. Процедура вложения образует $K = M - L + 1$ векторов вложения

$$X_i = \left(x(t_{i-1}), \dots, x(t_{i+L-2})\right)^T, 1 \le i \le K$$

имеющих размерность L. Здесь $TS = (x_0, \dots, x_{M-1}) = (x(t_0), \dots, x(t_{M-1}))$ – временной ряд длины M.

Траекторная матрица ряда TS состоит из векторов вложения в качестве столбцов.

Шаг 2. Сингулярное разложение

В результате этого шага получается сингулярное разложение траекторной матрицы X исходного временного ряда. Оно может быть записано в следующем виде:

$$X = X_1 + \dots + X_d$$

Здесь $d = \max\{i: \lambda_i > 0\}$, где $\lambda_1 \dots \lambda_L$ – собственные числа матрицы $S = XX^T$, взятые в неубывающем порядке; $X_i = \sqrt{\lambda_i} U_i V_i^T$, где $U_1, \dots U_L$ – ортонормированная система собственных векторов матрицы S, а $V_i = X^T U_i / \sqrt{\lambda_i}$.

Этап 2: восстановление

Шаг 3. Группировка

Следующим шагом является группировка членов сингулярного разложения, соответствующая разделимым динамической (включает в себя тренд и периодичность) и остаточной составляющим ряда.

В результате группировки индексы $\{1, \dots, d\}$ разделяются на непересекающиеся подмножества $\{I_1, \dots, I_m\}$. Разложение, полученное на предыдущем шаге, может быть записано в сгруппированном виде

$$X = X_{I_1} + \cdots + X_{I_m}.$$

Здесь $X_{I_k} = X_{i_1} + \cdots + X_{i_{p_k}},\ k = \overline{1,m},\ I_k = \{i_1, \dots i_{p_k}\}$.

Шаг 4. Диагональное усреднение

На этом шаге каждая матрица сгруппированного разложения переводится в новый ряд длины M. Диагональное усреднение переводит матрицу Y размерности $L \times K$ с элементами y_{ij} в ряд $g_0 \dots g_{N-1}$ по формуле

$$g_k = \begin{cases} \dfrac{1}{k+1} \displaystyle\sum_{m=1}^{k+1} y^*_{m,k-m+2}\ , 0 \le k \le L^* - 1 \\ \dfrac{1}{L^*} \displaystyle\sum_{m=1}^{L^*} y^*_{m,k-m+2}\ , L^* - 1 \le k \le K^* \\ \dfrac{1}{M-k} \displaystyle\sum_{m=k-K^*+2}^{M-K^*+1} y^*_{m,k-m+2}\ , K^* \le k \le M \end{cases}$$

Здесь $L^* = \min(L, K)$, $K^* = \max(L, K)$, $M = L + K - 1$; $y^*_{ij} = y_{ij}$, если $L < K$, и $y^*_{ij} = y_{ji}$, если $L \ge K$.

Применяя диагональное разложение к матрицам X_{I_k}, получаем ряды $\widetilde{TS}^{(k)} = \left(\tilde{x}_0^{(k)}, \dots, \tilde{x}_{M-1}^{(k)}\right)$. Исходный ряд $(x_0, \dots, x_{M-1})$ раскладывается в сумму m рядов

$$x_n = \sum_{k=1}^{m} \tilde{x}_n^{(k)}$$

1.1.2. Регуляризация данных с помощью сплайнов

Может оказаться, что данные во временных рядах расположены редко или нерегулярно. Это может повлечь за собой большую погрешность в вычислении производных, а также несовместность линейной системы (см. пункт 1.3). В этом случае предлагается выполнять регуляризацию данных, состоящую из двух последовательных шагов: (а) аппроксимация данных, (б) генерация дополнительных данных с помощью аппроксимирующей функции.

Основным кандидатом на роль аппроксимирующей функции стали интерполяционные сплайны, потому что, во-первых, правую часть СДУ требуется восстановить в полиномиальной форме, поэтому использование полиномиальных функций даст лучшие результаты, чем тригонометрические, и тем более, если нет оснований предполагать, что процесс, описываемый временным рядом, является периодическим.

Во-вторых, если мы говорим именно о полиномиальных функциях, то если интервал времени, на котором описывается процесс, велик, и нет оснований считать описывающую процесс функцию достаточно гладкой, то нет смысла использовать единый полином для аппроксимации этой функции, и нет смысла повышать качество аппроксимации за счет использования полиномов высоких степеней – в этом случае лучше использовать кусочно-заданную функцию, составленную из отдельных полиномов небольших степеней, определенных на своей части глобального временного интервала.

В-третьих, так как нелинейные непрерывные функции в окрестности заданной точки можно представить в виде степенного ряда с требуемой точностью, а частичными суммами этого ряда являются именно полиномы, то и было решено обратиться к аппроксимации полиномиальными функциями, только модифицированными – кусочно-заданными.

Постановка задачи аппроксимации временного ряда с помощью сплайнов выглядит следующим образом [38] (в одномерном случае): на $[a, b]$ – интервале времени, в течение которого проводились измерения, – задана сетка $\delta_M = \{t_0, t_1 \dots t_M\}$, узлами которой являются моменты времени, в которые делались измерения, и в этих узлах известны значения – сами измерения $x_i = x(t_i) \in \mathbb{R}^n, i \in \overline{0, M}$. На этом же интервале задана другая сетка $\Delta_N = \{z_0, z_1 \dots z_N\}$, $N \leq M$. Необходимо аппроксимировать имеющийся временной ряд на заданном отрезке $[a, b]$ с использованием кусочно-заданной функции $g(x) \in C[a, b]$, являющейся полиномами на заданных частичных отрезках интервала $[a, b]$ – сетке Δ_N.

- В случае кусочно-линейной аппроксимации интерполяционный полином на каждом из N частичных интервалов $[z_i, z_{i+1}], i = 1,2 \dots N$, выглядит следующим образом:

$$h_i = z_{i+1} - z_i$$

$$g_1(t) = \frac{z_{i+1} - t}{h_i} x_i + \frac{t - z_i}{h_i} x_{i+1}$$

- В случае кусочно-квадратичной аппроксимации количество узлов $z_0, z_1 \dots z_n$, в которых известны значения функции, предполагается четным: $n = 2m$. На m частичных интервалах $I_i = [z_{2i-2}, z_{2i+2}]$, $i = 1,2 \dots m$, определены квадратичные полиномы:

$$g_2(t) = \begin{cases} p_{2,1}(t) = a_1 t^2 + b_1 t + c_1, & \text{при } t \in [z_0, z_2] \\ p_{2,2}(t) = a_2 t^2 + b_2 t + c_2, & \text{при } t \in [z_2, z_4] \\ \dots & \\ p_{2,m}(t) = a_m t^2 + b_m t + c_m, & \text{при } t \in [z_{2m-2}, z_{2m}] \end{cases}$$

Неизвестные параметры квадратичного полинома a_i, b_i, c_i находятся из последовательного решения (при $i = 1,2 \dots m$) линейных систем:

$$\begin{cases} a_i t_{2i-2}^2 + b_i t_{2i-2} + c_i = x_{2i-2} \\ a_i t_{2i-1}^2 + b_i t_{2i-1} + c_i = x_{2i-1} \\ a_i t_{2i}^2 + b_i t_{2i} + c_i = x_{2i} \end{cases}$$

Для соблюдения условий непрерывности на границах отрезков необходимо добавить ограничения на непрерывность и дифференцируемость [28].

- В случае кусочно-кубической аппроксимации кусочно-кубический полином $g_3(t)$ является дважды непрерывно дифференцируемой функцией, и на каждом отрезке $I_i = [z_i, z_{i+1}], i = 0,1 \dots N-1$, является кубическим полиномом $g_{3,i}(t) = a_i t^3 + b_i t^2 + c_i t + e_i$, $t \in I_i$.

Для построения функции $g_3(t)$ надо определить $4N$ коэффициентов a_i, b_i, c_i, e_i, а значит, нужно составить $4N$ соотношений для однозначного определения функции.

В случае, когда $N = M$, то есть, когда в качестве узлов сетки Δ_N взяты элементы вектора $x(t_i) = \{x_i\}_{i=0}^{M}$, эти соотношения строятся следующим образом:

- $3N-3$ соотношений составляются из требования непрерывности на отрезке $[a, b]$ самой функции $g_3(t)$ и ее производных $g_3'(t), g_3''(t)$:

$$g_{3,i}^{(k)}(z_{i+1}) = g_{3,i+1}^{(k)}(z_{i+1}),\ k = 0,1,2,\ i = 0,1 \dots N-2.$$

1) $N+1$ равенств получаются из того факта, что в узлах z_i известно значение полинома $g_{3,i}$:

$$g_{3,i}(z_i) = y_i, \qquad i = 0,1 \dots N$$

2) Необходимые оставшиеся два условия могут назначаться из самых разных соображений, например:

$$g_3''(z_0) = g_3''(z_N) = 0$$

В случае же, когда $N < M$, то есть, когда в качестве узлов аппроксимации выбраны не все элементы $\{x_i\}_{i=0}^{M}$, также составляется $3N-3$ соотношений, следующих из непрерывности, а также имеется $M+1$ значений полинома в промежуточных между узлами точках, и этих соотношений в случае, когда M намного больше N, уже достаточно для однозначного определения коэффициентов. Если же оказывается, что их недостаточно, можно построить соотношения, аналогичные описанным в пункте 3.

1.2 Поиск зависимостей в данных

Очень часто в реальных задачах данные временных рядов могут иметь очень высокую размерность, что приводит к невозможности установить зависимости между ними. Таким образом, восстановление динамической многомерной системы является нереализуемым на практике в силу эффекта «проклятия размерности». Чтобы преодолеть эту сложность, предлагается схема реконструкции системы с предварительной группировкой фазовых переменных.

Коэффициент корреляции Пирсона – наиболее часто используемая мера корреляции между переменными. Для пары X и Y коэффициент корреляции Пирсона определяется как

$$r_{XY} = \frac{cov_{XY}}{\sigma_X \sigma_Y},$$

где cov_{XY} – ковариация между X и Y, σ_X и σ_Y – стандартные отклонения X и Y. Коэффициент корреляции Пирсона чувствителен к выбросам и является мерой линейной корреляции между рядами. Коэффициенты же ранговой корреляции Спирмена и Кендалла являются устойчивыми к выбросам и лучше подходят для данных, распределенных не по нормальному закону. Коэффициент корреляции Спирмена вычисляется по формуле

$$\rho_{XY} = \frac{\sum_{i=1}^{n}(R(x_i) - \overline{R(x)})(R(y_i) - \overline{R(y)})}{\sqrt{\left(\sum_{i=1}^{n}\left(R(x_i) - \overline{R(x)}\right)^2\right)\left(\sum_{i=1}^{n}\left(R(y_i) - \overline{R(y)}\right)^2\right)}},$$

где $R(x_i), R(y_i)$ – ранги i-х значений рядов X, Y; $\overline{R(x)} = \overline{R(y)} = \frac{n+1}{2}$, n – размер выборки. Коэффициент Кедалла вычисляется по формуле

$$\tau_{XY} = \frac{n_c - n_d}{\frac{1}{2}n(n-1)},$$

где n_c – число согласующихся пар, n_d – число не согласующихся пар (пары (x_i, y_i) и (x_j, y_j) называются согласующимися, если $x_i > x_j, y_i > y_j$ или $x_i < x_j, y_i < y_j$, иначе они называются не согласующимися).

Взаимная информация – это мера обмена информации между двумя случайными величинами, и она содержит информацию обо всех зависимостях, то есть как о линейных, так и о нелинейных корреляциях. Взаимная информация X и Y вычисляется по формуле

$$I(X,Y) = \sum_{X,Y} p(X,Y) \log_2 \frac{p(X,Y)}{p(X)p(Y)},$$

где $p(X,Y)$ – совместное распределение X и Y; $p(X)$, $p(Y)$ – маргинальные распределения X и Y соответственно.

Временные ряды группируются на основе их корреляций. Для хорошей группировки ряды в одной группе должны быть как можно сильнее коррелированы, тогда как корреляции между группами должны быть как можно меньше. Оптимальная группировка рядов получается путем максимизации оценки

$$G_{total} = G_{intrag} + G_{interg} = \sum_{k=1}^{K} \sum_{i,j \in g_k, i \neq j} \frac{|\rho_{ij}|}{m_k - 1} + \sum_{k=1}^{K} \sum_{i \notin g_k, j \in g_k} \frac{1 - |\rho_{ij}|}{M - m_k},$$

где i, j обозначают i-й и j-й ряды, ρ_{ij} – мера корреляции между рядами i, j; g_k обозначает k-ю группу, K – количество групп, m_k – количество рядов в k-й группе, M – общее

количество рядов, G_{intrag} – сумма попарных корреляций между рядами в одной группе, G_{interg} – сумма попарных независимостей между переменными в разных группах.

В процессе группировки сначала определяется начальное распределение рядов по группам, а затем группы переменных итеративно обновляются путем перевода рядов из одной группы в другую с целью максимизации оценки G_total.

1.3 Реконструкция обыкновенных дифференциальных уравнений в полиномиальной форме

В качестве основы для предлагаемого алгоритма был использован алгоритм реконструкции, описанный в [22].

Пусть вектор $\boldsymbol{X}$ – это набор параметров, описывающих процесс, с зависящими от времени компонентами $X_j(t), j = \overline{1,n}$. Пусть известны значения векторной функции $\boldsymbol{X}(t)$, измеренные в моменты времени $t_0, \dots t_{M+1}$: $\boldsymbol{X}(t_0), \dots \boldsymbol{X}(t_{M+1})$.

Предполагается, что временной ряд данных, описывающих динамический процесс, может быть описан моделью, представляющей собой систему автономных дифференциальных уравнений:

$$\frac{d\boldsymbol{X}}{dt} = \sum_{k=0}^{N} P^k \boldsymbol{X}^{[k]} \tag{1.1}$$

где $\boldsymbol{X} \in \mathbb{R}^n$ – вектор состояний системы, и $\boldsymbol{X}^{[k]}$ – k-я степень Кронекера вектора $\boldsymbol{X}$ (например, для $\boldsymbol{X} = (x_1, x_2)$ вторая степень равна $\boldsymbol{X}^{[2]} = (x_1^2, x_1 x_2, x_2^2)$).

Матрицы P^k неизвестны и должны быть найдены с использованием имеющихся измерений $\boldsymbol{X}(t_0), \dots \boldsymbol{X}(t_{M+1})$. Если во временном ряде эти измерения расположены часто, то эти матрицы можно легко вычислить, решая систему линейных уравнений, которая получается после замены производных $\frac{d\boldsymbol{X}}{dt}$ в левой части системы конечными разностями:

$$\frac{\boldsymbol{X}(t_{i+1}) - \boldsymbol{X}(t_{i-1})}{t_{i+1} - t_{i-1}} = \sum_{k=0}^{N} P^k \boldsymbol{X}^{[k]}(t_i), \qquad i = \overline{1,M}$$

Эту систему можно переписать в матричной форме:

$$AP = B \tag{1.2}$$

еде:

$$A = \begin{bmatrix} \left(\boldsymbol{X}(t_1)\right)^T & \left(\boldsymbol{X}^{[2]}(t_1)\right)^T & \dots & \left(\boldsymbol{X}^{[N]}(t_1)\right)^T \\ \left(\boldsymbol{X}(t_2)\right)^T & \left(\boldsymbol{X}^{[2]}(t_2)\right)^T & \dots & \left(\boldsymbol{X}^{[N]}(t_2)\right)^T \\ \vdots & \vdots & \ddots & \vdots \\ \left(\boldsymbol{X}(t_M)\right)^T & \left(\boldsymbol{X}^{[2]}(t_M)\right)^T & \dots & \left(\boldsymbol{X}^{[N]}(t_M)\right)^T \end{bmatrix}$$

$$P = (P^1, \dots P^N)^T$$

$$B = \begin{bmatrix} \dfrac{\boldsymbol{X}(t_2) - \boldsymbol{X}(t_0)}{t_2 - t_0} \\ \dots \\ \dfrac{\boldsymbol{X}(t_{M+1}) - \boldsymbol{X}(t_{M-1})}{t_{M+1} - t_{M-1}} \end{bmatrix}$$

В системе (1.2) $n \cdot (n + n_2 + \cdots + n_N)$ неизвестных и $n \cdot M$ уравнений. Здесь n – размерность вектора $\boldsymbol{X}$ и $n_i,\ i \geq 2$ – размерность его i-й степени Кронекера. Следовательно, чтобы система (1.2) была однозначно разрешима (а для этого нужно, чтобы количество неизвестных было равно количеству уравнений), нужно, чтобы выполнялось следующее условие:

$$M = n + n_2 + \cdots + n_N. \qquad (1.3)$$

Однако, если имеющиеся измерения расположены во времени редко, описанный подход нельзя применить, так как система (1.2) может оказаться неопределенной (нарушится условие (1.3)), а также конечные разности производных будут вычислены с большой погрешностью.

Чтобы обойти это препятствие, предлагается сначала аппроксимировать компоненты вектора измерений с помощью сплайнов и затем сгенерировать с помощью аппроксимирующих функций дополнительные точки.

Однако, чтобы это сделать, нужно определить параметры сплайна, которые влияют на итоговый результат аппроксимации, а следовательно, на точность аппроксимации сплайном истинной функции, стоящей за временным рядом. Исходя из определения сплайна, у него имеются три таких параметра: степень n, гладкость k и сетка Δ_N (эта сетка в общем случае для каждой компоненты своя).

Если изначально неизвестно, полиномы какой степени присутствуют в исходной модели, или насколько гладкой она является, нет никаких оснований склоняться к выбору определенной степени n или гладкости k.

Выбор оптимального расположения узлов сетки Δ_{Ni}, которая в общем случае является разной для каждой компоненты, путем перебора разных комбинаций исходных точек, которых может быть огромное количество, будет занимать очень много времени и вычислительной мощности, поэтому нужно сузить круг вариантов этих возможных расположений, например, следующим образом: в качестве узлов сетки Δ_{Ni} для каждой компоненты $X_i,\ i = \overline{1, N}$ берутся каждые l-е точки $X_i(t_0), X_i(t_1) \dots X_i(t_M)$. Величина l подлежит определению. И, таким образом, чтобы задать сетку Δ_N в предположении, что для каждой компоненты значения l различны, нужно задать вектор $L = [l_1, l_2 \dots l_n]$, который описывает, с каким шагом берутся точки во временном ряде каждой компоненты вектора

X. Для упрощения же вычислений, уменьшения их количества и уменьшения временных затрат можно предположить, что l для всех компонент одинаково.

В сумме с параметром s_d – шагом генерации точек для решения линейной системы, у алгоритма, использующего аппроксимацию сплайнами, есть четыре параметра, влияющие на итоговое решение. Jуи должны быть найдены как решение оптимизационной задачи

$$n, k, l, s_d = \arg\min \|\boldsymbol{X} - \boldsymbol{Y}\|.$$

Итак, с учетом описания применения аппроксимации сплайнами, можно составить аналогичный алгоритму [22] алгоритм восстановления правой части системы дифференциальных уравнений:

1. Построение аппроксимирующей функции для каждой из компонент вектора $X(t)$. Определение коэффициентов n, k, l, s_d
2. Генерация новых точек, используя аппроксимирующую функцию
3. Решение системы линейных уравнений и нахождение матриц коэффициентов для представления правой части СДУ в полиномиальном виде

2 ПОСТРОЕНИЕ МАТРИЧНОГО ПРЕОБРАЗОВАНИЯ ЛИ ДЛЯ НЕСТАЦИОНАРНОЙ ДИНАМИЧЕСКОЙ СИСТЕМЫ

В данной главе рассматривается задача инициализации матриц весов полиномиальной нейронной сети, моделирующей нестационарную динамическую систему. При решении задачи инициализации предполагается, что матрицы полиномиальной правой части системы дифференциальных уравнений, описывающей моделируемый процесс, уже определены на этапе решения задачи реконструкции системы. Для нахождения весовых матриц полиномиальной нейронной сети рассматривается подход, заключающийся в переходе от реконструированной системы дифференциальных уравнений к системе дифференциальных уравнений повышенной размерности. Приведены выводы рекуррентных аналитических представлений для матриц указанной системы. Приведены формулы решения линеаризованной системы дифференциальных уравнений повышенной размерности, а также разработан алгоритм приближенного решения исходной реконструированной системы, содержащий формулы для расчёта весовых матриц. Выполнено сравнение двух приближённых решений реконструированной системы дифференциальных уравнений, приведена аналитическая оценка точности приближенного решения. Также оценены необходимый для достижения заданной точности порядок полиномиального разложения и временной шаг.

2.1 Постановка задачи

Пусть динамический процесс моделируется нелинейной системой обыкновенных дифференциальных уравнений с полиномиальной правой частью:

$$\frac{d\boldsymbol{X}}{dt} = \sum_{k=1}^{N} \boldsymbol{P^{1k}}(\boldsymbol{t})\boldsymbol{X}^{[k]} \qquad (2.1)$$

Здесь $\boldsymbol{P^{1k}}(\boldsymbol{t})$ – двумерная матрица, $\boldsymbol{X}^{[k]} = \underbrace{\boldsymbol{X} * \ldots * \boldsymbol{X}}_{k \text{ раз}}$– степень Кронекера k-го порядка для фазового вектора $\boldsymbol{X}$, которую будем называть вектором фазовых моментов k-го порядка. Умножение $\boldsymbol{X}*\boldsymbol{Y}$ является кронекеровым произведением вектора $\boldsymbol{X}$ на вектор $\boldsymbol{Y}$ [14].

Предположим, что решение задачи Коши

$$\boldsymbol{X}(\boldsymbol{t_0}) = \boldsymbol{X}_0 \qquad (2.2)$$

для системы (2.1) можно найти в виде ряда по кронекеровским степеням начального вектора $\boldsymbol{X}_0$:

$$\boldsymbol{X}(t,t_0) = \sum_{k=1}^{\infty} R^{1k}(\boldsymbol{t})\boldsymbol{X}_0^{[k]} \tag{2.3}$$

В этом случае приближённое решение задачи Коши (2.2) для системы (2.1) можно найти в виде полинома по кронекеровским степеням начального вектора $\boldsymbol{X}_0$:

$$\boldsymbol{X}_{\boldsymbol{K}}(t,t_0) = \sum_{k=1}^{K} R^{1k}(\boldsymbol{t})\boldsymbol{X}_0^{[k]} \tag{2.4}$$

2.2 Повышение размерности системы дифференциальных уравнений

Для построения приближённого решения системы (2.1) вида (2.4) увеличим размерность системы (2.1). Для этого введём вектор фазовых моментов до порядка $\boldsymbol{K}$: $\boldsymbol{X}^{\boldsymbol{K}} = \left[\boldsymbol{X}^{[1]}, \boldsymbol{X}^{[2]}, \dots, \boldsymbol{X}^{[K]}\right]$ и вычислим производные для всех векторов $\boldsymbol{X}^{[i]}, \boldsymbol{i} = \overline{\boldsymbol{2,K}}$

Вычислим первую производную по t для каждого вектора фазовых моментов i-го порядка $\boldsymbol{X}^{[i]}, \boldsymbol{i} = \overline{\boldsymbol{1,K}}$. Производная вектора фазовых моментов первого порядка представлена системой (2.1). Вычислим производную вектора фазовых моментов второго порядка:

$$\frac{d\boldsymbol{X}^{[2]}}{dt} = \frac{d(\boldsymbol{X}*\boldsymbol{X})}{dt} = \left\{\frac{d\boldsymbol{X}}{dt}*\boldsymbol{X} + \boldsymbol{X}*\frac{d\boldsymbol{X}}{dt}\right\} = \left\{\left(\sum_{k=1}^{N}\boldsymbol{P}^{1k}(\boldsymbol{t})X^{[k]}\right)*\boldsymbol{X} + \boldsymbol{X}*\left(\sum_{k=1}^{N}\boldsymbol{P}^{1k}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N}\left(\boldsymbol{P}^{1k}(\boldsymbol{t})X^{[k]}\right)*\boldsymbol{X} + \sum_{k=1}^{N}\boldsymbol{X}*\left(\boldsymbol{P}^{1k}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N}\left(\boldsymbol{P}^{1k}(\boldsymbol{t})X^{[k]}\right)*(E\boldsymbol{X}) + \sum_{k=1}^{N}(E\boldsymbol{X})*\left(\boldsymbol{P}^{1k}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \sum_{k=1}^{N}\left\{\left(\boldsymbol{P}^{1k}(\boldsymbol{t})X^{[k]}\right)*(E\boldsymbol{X}) + (E\boldsymbol{X})*\left(\boldsymbol{P}^{1k}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \sum_{k=1}^{N}\left\{\left(\boldsymbol{P}^{1k}(\boldsymbol{t})*E\right)\left(X^{[k]}*\boldsymbol{X}\right) + \left(E*\boldsymbol{P}^{1k}(\boldsymbol{t})\right)\left(\boldsymbol{X}*X^{[k]}\right)\right\}$$

$$= \sum_{k=1}^{N}\left\{\left(\boldsymbol{P}^{1k}(\boldsymbol{t})*E\right) + \left(E*\boldsymbol{P}^{1k}(\boldsymbol{t})\right)\right\}X^{[k+1]} = \sum_{k=2}^{N+1}\boldsymbol{P}^{2k}(\boldsymbol{t})X^{[k]}$$

где

$$\boldsymbol{P}^{2k}(\boldsymbol{t}) = \left\{\left(\boldsymbol{P}^{1k-1}(\boldsymbol{t})*E\right) + \left(E*\boldsymbol{P}^{1k-1}(\boldsymbol{t})\right)\right\}$$

Здесь и далее фигурные скобки означают необходимость сложения столбцов, соответствующих одинаковым мономам в векторе $\boldsymbol{X}^{[i]}$ и удаление равных строк матрицы, полученной в результате вычисления выражения внутри фигурных скобок.

Матрица E – единичная матрица размерности вектора $\boldsymbol{X}$.

Вычислим производную вектора фазовых моментов третьего порядка:

$$\frac{d\boldsymbol{X}^{[3]}}{dt} = \frac{d\left(\boldsymbol{X} * \boldsymbol{X}^{[2]}\right)}{dt} = \left\{\frac{d\boldsymbol{X}}{dt} * \boldsymbol{X}^{[2]} + \boldsymbol{X} * \frac{d\boldsymbol{X}^{[2]}}{dt}\right\}$$

$$= \left\{\left(\sum_{k=1}^{N} \boldsymbol{P^{1k}}(\boldsymbol{t})X^{[k]}\right) * \boldsymbol{X}^{[2]} + \boldsymbol{X} * \left(\sum_{k=2}^{N+1} \boldsymbol{P^{2k}}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N} \left(\boldsymbol{P^{1k}}(\boldsymbol{t})X^{[k]}\right) * \boldsymbol{X}^{[2]} + \sum_{k=2}^{N+1} \boldsymbol{X} * \left(\boldsymbol{P^{2k}}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N} \left(\boldsymbol{P^{1k}}(\boldsymbol{t})X^{[k]}\right) * \left(E^{[2]}\boldsymbol{X}^{[2]}\right) + \sum_{k=2}^{N+1} (E\boldsymbol{X}) * \left(\boldsymbol{P^{2k}}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N} \left(\boldsymbol{P^{1k}}(\boldsymbol{t}) * E^{[2]}\right)\left(X^{[k]} * \boldsymbol{X}^{[2]}\right) + \sum_{k=2}^{N+1} \left(E * \boldsymbol{P^{2k}}(\boldsymbol{t})\right)\left(\boldsymbol{X} * X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N} \left(\boldsymbol{P^{1k}}(\boldsymbol{t}) * E^{[2]}\right)X^{[k+2]} + \sum_{k=2}^{N+1} \left(E * \boldsymbol{P^{2k}}(\boldsymbol{t})\right)X^{[k+1]}\right\}$$

$$= \sum_{k=3}^{N+2} \left\{\left(\boldsymbol{P^{1k-2}}(\boldsymbol{t}) * E^{[2]}\right) + \left(E * \boldsymbol{P^{2k-1}}(\boldsymbol{t})\right)\right\}X^{[k]} = \sum_{k=3}^{N+2} \boldsymbol{P^{3k}}(\boldsymbol{t})X^{[k]}$$

где

$$\boldsymbol{P^{3k}}(\boldsymbol{t}) = \left\{\left(\boldsymbol{P^{1k-2}}(\boldsymbol{t}) * E^{[2]}\right) + \left(E * \boldsymbol{P^{2k-1}}(\boldsymbol{t})\right)\right\}$$

Матрица $\boldsymbol{E}^{[2]} = \boldsymbol{E} * \boldsymbol{E}$ –вторая кронекеровская степень единичной матрицы E

Покажем по индукции, что

$$\frac{d\boldsymbol{X}^{[i]}}{dt} = \sum_{k=i}^{N+i-1} \boldsymbol{P^{ik}}(\boldsymbol{t})X^{[k]} \tag{2.5}$$

где

$$\boldsymbol{P^{ik}}(\boldsymbol{t}) = \left\{\left(\boldsymbol{P^{1k-i+1}}(\boldsymbol{t}) * E^{[i-1]}\right) + \left(E * \boldsymbol{P^{i-1k-1}}(\boldsymbol{t})\right)\right\} \tag{2.6}$$

Предположив верность индукционного предположения для некоторого *i*, покажем его верность для *i*+*1*:

$$\frac{d\boldsymbol{X}^{[i+1]}}{dt} = \frac{d\left(\boldsymbol{X} * \boldsymbol{X}^{[i]}\right)}{dt} = \left\{\frac{d\boldsymbol{X}}{dt} * \boldsymbol{X}^{[i]} + \boldsymbol{X} * \frac{d\boldsymbol{X}^{[i]}}{dt}\right\}$$

$$= \left\{\left(\sum_{k=1}^{N} \boldsymbol{P^{1k}}(\boldsymbol{t})X^{[k]}\right) * \boldsymbol{X}^{[i]} + \boldsymbol{X} * \left(\sum_{k=i}^{N+i-1} \boldsymbol{P^{ik}}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N} \left(\boldsymbol{P^{1k}}(\boldsymbol{t})X^{[k]}\right) * \boldsymbol{X}^{[i]} + \sum_{k=i}^{N+i-1} \boldsymbol{X} * \left(\boldsymbol{P^{ik}}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N} \left(\boldsymbol{P^{1k}}(\boldsymbol{t})X^{[k]}\right) * \left(E^{[i]}\boldsymbol{X}^{[i]}\right) + \sum_{k=i}^{N+i-1} (E\boldsymbol{X}) * \left(\boldsymbol{P^{ik}}(\boldsymbol{t})X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N} \left(\boldsymbol{P^{1k}}(\boldsymbol{t}) * E^{[i]}\right)\left(X^{[k]} * \boldsymbol{X}^{[i]}\right) + \sum_{k=i}^{N+i-1} \left(E * \boldsymbol{P^{ik}}(\boldsymbol{t})\right)\left(\boldsymbol{X} * X^{[k]}\right)\right\}$$

$$= \left\{\sum_{k=1}^{N} \left(\boldsymbol{P^{1k}}(\boldsymbol{t}) * E^{[i]}\right)X^{[k+i]} + \sum_{k=i}^{N+i-1} \left(E * \boldsymbol{P^{ik}}(\boldsymbol{t})\right)X^{[k+1]}\right\}$$

$$= \sum_{k=i+1}^{N+i} \left\{\left(\boldsymbol{P^{1k-i}}(\boldsymbol{t}) * E^{[i]}\right) + \left(E * \boldsymbol{P^{ik-1}}(\boldsymbol{t})\right)\right\}X^{[k]} = \sum_{k=i+1}^{N+i} \boldsymbol{P^{i+1k}}(\boldsymbol{t})X^{[k]}$$

где

$$\boldsymbol{P^{i+1k}}(\boldsymbol{t}) = \left\{\left(\boldsymbol{P^{1k-i}}(\boldsymbol{t}) * E^{[i]}\right) + \left(E * \boldsymbol{P^{ik-1}}(\boldsymbol{t})\right)\right\}$$

Таким образом, верность индукционного предположения (2.5) – (2.6) доказана.

Из доказанного представления **(2.5)** следует, что уравнению $(2.\mathbf{1})$ можно сопоставить уравнение в пространстве фазовых моментов до порядка K:

$$\frac{\boldsymbol{dX^K}}{\boldsymbol{dt}} = \boldsymbol{H^K X^{K+N-1}} \qquad (2.\mathbf{7})$$

где,

$$P^K = \begin{pmatrix} P^{11} & P^{12} & \dots & P^{1K} \\ \mathbb{0} & P^{22} & \dots & P^{2K} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbb{0} & \mathbb{0} & \dots & P^{KK} \end{pmatrix}$$

$$G^K = \begin{pmatrix} P^{1K+1} & P^{1K+2} & \dots & P^{1K+N-1} \\ P^{2K+1} & P^{2K+2} & \dots & P^{2K+N-1} \\ \vdots & \vdots & \vdots & \vdots \\ P^{KK+1} & P^{KK+2} & \dots & P^{KK+N-1} \end{pmatrix}$$

$$\boldsymbol{H^K} = (P^K, G^K)$$

Здесь матрицы $\boldsymbol{P^{ik}}(\boldsymbol{t})$ вычисляются по рекуррентным формулам (2.6), а матрицы $\boldsymbol{P^{1k}}(\boldsymbol{t})$ известны из правой части уравнения (2.1), причём $\boldsymbol{P^{1k}}(\boldsymbol{t}) = \boldsymbol{O}$ при $k>N$.

В частности, уравнению (2.1) можно сопоставить уравнение в пространстве фазовых моментов до порядка *2N-1*:

$$\frac{d\boldsymbol{X}^{\boldsymbol{N}}}{\boldsymbol{dt}} = \boldsymbol{H}^{\boldsymbol{N}}\boldsymbol{X}^{\boldsymbol{2N-1}} \tag{2.8}$$

где,

$$P^N = \begin{pmatrix} P^{11} & P^{12} & \dots & P^{1N} \\ \mathbb{0} & P^{22} & \dots & P^{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbb{0} & \mathbb{0} & \dots & P^{NN} \end{pmatrix}$$

$$G^N = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ P^{2N+1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ P^{NN+1} & P^{NN+2} & \dots & 0 \end{pmatrix}$$

$$\boldsymbol{H}^{\boldsymbol{N}} = (P^N, G^N)$$

2.3 Линеаризованная система повышенной размерности и её решение

Рассмотрим линейную часть системы (2.7)

$$\frac{d\boldsymbol{X}^{\boldsymbol{K}}}{\boldsymbol{dt}} = \boldsymbol{P}^{\boldsymbol{K}}\boldsymbol{X}^{\boldsymbol{K}} \tag{2.9}$$

Система дифференциальных уравнений **Ошибка! Источник ссылки не найден.** по форме является линейной, поэтому решение задачи Коши для данной системы

$$\boldsymbol{X}^K(t_0) = (\boldsymbol{X}_0)^{[K]} \equiv \boldsymbol{X}_0^K \tag{2.10}$$

соответствующее задаче Коши (2.2) для системы (2.1) может быть записано с помощью нелинейного матрицанта

$$\boldsymbol{X}^{\boldsymbol{K}} = \boldsymbol{R}^{\boldsymbol{K}}(\boldsymbol{t}, \boldsymbol{t_0})\boldsymbol{X}_{\boldsymbol{0}}^{\boldsymbol{K}} \tag{2.11}$$

где $R^K(t_0, \boldsymbol{t_0}) = E^K$.

Матрицант R^K имеет блочную структуру:

$$R^K = \begin{pmatrix} R^{11} & R^{12} & \dots & R^{1K} \\ \mathbb{0} & R^{22} & \dots & R^{2K} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbb{0} & \mathbb{0} & \dots & R^{NK} \end{pmatrix}$$

для элементов которого необходимо определить явные выражения.

Для нахождения блочных элементов матрицанта R^K представим линейную систему (2.9) как систему, состоящую из K подсистем линейных дифференциальных уравнений:

$$\begin{cases} \dfrac{dX}{dt} = \boldsymbol{P^{11}(t)X} + \boldsymbol{P^{12}(t)X^{[2]}} + \cdots + \boldsymbol{P^{1K}(t)X^{[K]}} \\ \dfrac{dX^{[2]}}{dt} = \boldsymbol{P^{22}(t)X^{[2]}} + \cdots + \boldsymbol{P^{2K}(t)X^{[K]}} \\ \cdots \\ \dfrac{dX^{[K-1]}}{dt} = \boldsymbol{P^{K-1K-1}(t)X^{[K-1]}} + \boldsymbol{P^{K-1K}(t)X^{[K]}} \\ \dfrac{dX^{[K]}}{dt} = \boldsymbol{P^{KK}(t)X^{[K]}} \end{cases} \quad (2.12)$$

Будем решать данную систему начиная с последнего уравнения:

$$\frac{dX^{[K]}}{dt} = \boldsymbol{P^{KK}(t)X^{[K]}}$$

Решение этого линейного уравнения можно представить в виде

$$\boldsymbol{X^{[K]}} = R^{KK}(\boldsymbol{t, t_0})\boldsymbol{X_0^{[K]}}$$

Для нахождения матрицанта R^{KK}, а также всех диагональных матрицантов R^{ii} рассмотрим линеаризованное первое уравнение системы (2.8)

$$\frac{dX}{dt} = \boldsymbol{P^{11}(t)X} \quad (2.13)$$

Решение задачи Коши $\boldsymbol{X}(t_0) = \boldsymbol{X_0}$ для этого уравнения будет иметь вид

$$\boldsymbol{X(t, t_0) = R^{11}(t, t_0)X_0} \quad (2.14)$$

Матрица $R^{11}(t, t_0)$ для случая постоянной матрицы $\boldsymbol{P^{11}}$ может быть вычислена как матричная экспонента

$$\boldsymbol{R^{11}(t, t_0) = e^{P^{11}(t - t_0)}} \quad (2.15)$$

Возведя обе части линеаризованного уравнения (2.13) в кронекеровскую степень можно построить уравнение для кронекеровской степени k:

$$\frac{dX^{[k]}}{dt} = \boldsymbol{P^{kk}X^{[k]}}$$

Возводя последовательно выражение (2.14) в кронекеровские степени, получим

$$X^{[k]}(\boldsymbol{t, t_0}) = \left(\boldsymbol{R^{11}(t, t_0)}\right)^{[k]} \boldsymbol{X_0^{[k]}}$$

Из данного представления следует, что

$$\boldsymbol{R^{kk}(t, t_0)} = \left(\boldsymbol{R^{11}(t, t_0)}\right)^{[k]}, \quad (2.16)$$

в частности, $\boldsymbol{R^{NN}(t, t_0)} = \left(\boldsymbol{R^{11}(t, t_0)}\right)^{[N]}$.

Далее будем решать предпоследнюю подсистему системы (2.12)

$$\frac{d\boldsymbol{X}^{[K-1]}}{dt} = \boldsymbol{P}^{K-1K-1}(t)\boldsymbol{X}^{[K-1]} + \boldsymbol{P}^{K-1K}(t)\boldsymbol{X}^{[K]}$$

Подставляя в неё представление для $\boldsymbol{X}^{[K]}$ получим линейную неоднородную систему относительно $\boldsymbol{X}^{[K-1]}$:

$$\frac{d\boldsymbol{X}^{[K-1]}}{dt} = \boldsymbol{P}^{K-1K-1}(t)\boldsymbol{X}^{[K-1]} + \boldsymbol{P}^{K-1K}(t)R^{KK}(t,t_0)\boldsymbol{X}_0^{[K]}$$

Решение данной системы можно записать с использованием формулы Коши в виде:

$$\boldsymbol{X}^{[K-1]}(t,t_0) = R^{K-1K-1}(t,t_0)\left(\boldsymbol{X}_0^{[K-1]} + \int_{t_0}^{t}\left(R^{K-1K-1}(\tau,t_0)\right)^{-1}\boldsymbol{P}^{K-1K}(\tau)R^{KK}(\tau,t_0)\boldsymbol{X}_0^{[K]}d\tau\right)$$

$$= R^{K-1K-1}(t,t_0)\boldsymbol{X}_0^{[K-1]} + R^{K-1K}(t,t_0)\boldsymbol{X}_0^{[K]}, \qquad \#\ (2.17)$$

где

$$\boldsymbol{R}^{K-1K}(t,t_0) = \int_{t_0}^{t}\boldsymbol{R}^{K-1K-1}(t,\tau)\boldsymbol{P}^{K-1K}(\tau)\boldsymbol{R}^{KK}(\tau,t_0)d\tau\,. \qquad (2.18)$$

Продолжим последовательно решать подсистемы системы (2.12).

Предположим, что подсистема

$$\frac{d\boldsymbol{X}^{[i]}}{dt} = \boldsymbol{P}^{ii}(t)\boldsymbol{X}^{[i]} + \cdots + \boldsymbol{P}^{iN}(t)\boldsymbol{X}^{[N]} \qquad (2.19)$$

имеет решение

$$\boldsymbol{X}^{[i]}(t,t_0) = \sum_{k=i}^{K} R^{ik}(t,t_0)\boldsymbol{X}_0^{[k]}, \qquad (2.20)$$

где

$$R^{ij}(t,t_0) = \sum_{l=i+1}^{j}\int_{t_0}^{t} R^{ii}(t,\tau)P^{il}(\tau)R^{lj}(\tau,t_0)d\tau \qquad (2.21)$$

Покажем по индукции справедливость указанных формул для произвольного i.

Рассмотрим подсистему

$$\frac{d\boldsymbol{X}^{[i-1]}}{dt} = \boldsymbol{P}^{i-1i-1}(t)\boldsymbol{X}^{[i-1]} + \cdots + \boldsymbol{P}^{i-1K}(t)\boldsymbol{X}^{[K]} \qquad (2.22)$$

Подставим в неё представления для $\boldsymbol{X}^{[i]},\ldots,\ \boldsymbol{X}^{[K]}$ из индукционного предположения (2.20)-(2.21).

Получим линейную неоднородную систему относительно $\boldsymbol{X}^{[i-1]}$:

$$\frac{d\boldsymbol{X}^{[i-1]}}{dt} = \boldsymbol{P^{i-1i-1}(t)X^{[i-1]}} + \boldsymbol{P^{i-1i}(t)}\left(R^{ii}(t,t_0)\boldsymbol{X}_0^{[i]} + R^{ii+1}(t,t_0)\boldsymbol{X}_0^{[i+1]} + \cdots + R^{iK}(t,t_0)\boldsymbol{X}_0^{[K]}\right) +$$
$$+\boldsymbol{P^{i-1i+1}(t)}\left(R^{i+1i+1}(t,t_0)\boldsymbol{X}_0^{[i+1]} + \cdots + R^{i+1K}(t,t_0)\boldsymbol{X}_0^{[K]}\right) +$$
$$\cdots$$
$$+\boldsymbol{P^{i-1K-1}(t)}\left(R^{K-1K-1}(t,t_0)\boldsymbol{X}_0^{[K-1]} + R^{K-1K}(t,t_0)\boldsymbol{X}_0^{[K]}\right) +$$
$$+\boldsymbol{P^{i-1K}(t)}\left(R^{KK}(t,t_0)\boldsymbol{X}_0^{[K]}\right) =$$
$$= \boldsymbol{P^{i-1i-1}(t)X^{[i-1]}} +$$
$$+\boldsymbol{P^{i-1i}(t)}R^{ii}(t,t_0)\boldsymbol{X}_0^{[i]} +$$
$$+\left(\boldsymbol{P^{i-1i}(t)}R^{ii+1}(t,t_0) + \boldsymbol{P^{i-1i+1}(t)}R^{i+1i+1}(t,t_0)\right)\boldsymbol{X}_0^{[i+1]} +$$
$$\cdots$$
$$+\left(\boldsymbol{P^{i-1i}(t)}R^{iK-1}(t,t_0) + \boldsymbol{P^{i-1i+1}(t)}R^{i+1K-1}(t,t_0) + \cdots + \boldsymbol{P^{i-1K-1}(t)}R^{K-1K-1}(t,t_0)\right)\boldsymbol{X}_0^{[K-1]}$$
$$+\left(\boldsymbol{P^{i-1i}(t)}R^{iK}(t,t_0) + \boldsymbol{P^{i-1i+1}(t)}R^{i+1K}(t,t_0) + \cdots + \boldsymbol{P^{i-1K}(t)}R^{KK}(t,t_0)\right)\boldsymbol{X}_0^{[K]}$$

Введя обозначения

$$V_{i-1j}(t,t_0) = \sum_{\boldsymbol{l=i}}^{\boldsymbol{j}} \boldsymbol{P^{i-1l}(t)}R^{lj}(t,t_0) \qquad \textbf{(2.23)}$$

получим линейную неоднородную систему относительно $\boldsymbol{X^{[i-1]}}$:

$$\frac{d\boldsymbol{X}^{[i-1]}}{dt} = \boldsymbol{P^{i-1i-1}(t)X^{[i-1]}} + \sum_{\boldsymbol{j=i}}^{\boldsymbol{K}} V_{i-1j}(t,t_0)\boldsymbol{X}_0^{[j]}$$

Эта система имеет решение

$$\boldsymbol{X}^{[i-1]}(t,t_0) = R^{i-1i-1}(t,t_0)\left(\boldsymbol{X}_0^{[i-1]} + \int_{t_0}^{t}\left(R^{i-1i-1}(\tau,t_0)\right)^{-1}\sum_{\boldsymbol{j=i}}^{\boldsymbol{K}} V_{i-1j}(\tau,t_0)\boldsymbol{X}_0^{[j]}\,d\tau\right)$$
$$= R^{i-1i-1}(t,t_0)\boldsymbol{X}_0^{[i-1]} + \sum_{\boldsymbol{j=i}}^{\boldsymbol{K}}\int_{t_0}^{t} R^{i-1i-1}(t,\tau)V_{i-1j}(\tau,t_0)\boldsymbol{X}_0^{[j]}d\tau$$
$$= R^{i-1i-1}(t,t_0)\boldsymbol{X}_0^{[i-1]} + \sum_{\boldsymbol{j=i}}^{\boldsymbol{K}}\left(\int_{t_0}^{t} R^{i-1i-1}(t,\tau)V_{i-1j}(\tau,t_0)d\tau\right)\boldsymbol{X}_0^{[j]}$$

Обозначив

$$R^{i-1j}(\tau,t_0) = \int_{t_0}^{t} R^{i-1i-1}(t,\tau)V_{i-1j}(\tau,t_0)d\tau, j \geq i$$

и воспользовавшись представлением для $V_{i-1j}(t,t_0)$ (2.23), получим при $j \geq i$:

$$R^{i-1j}(\tau,t_0)=\int_{t_0}^{t}R^{i-1i-1}(t,\tau)\sum_{l=i}^{j}\boldsymbol{P^{i-1l}}(\tau)R^{lj}(\tau,t_0)\,d\tau$$

$$=\sum_{l=(i-1)+1}^{j}\int_{t_0}^{t}R^{i-1i-1}(t,\tau)\boldsymbol{P^{i-1l}}(\tau)R^{lj}(\tau,t_0)d\tau$$

При этом

$$\boldsymbol{X}^{[i-1]}(t,t_0)=\sum_{k=i-1}^{K}R^{ik}(t,t_0)\boldsymbol{X}_0^{[k]}$$

является решением подсистемы (2.22), ч т.д.

Таким образом, представления для матрицантов $R^{ij}(t,t_0)$ (2.21) доказаны.

Таким образом, приближённое решение системы (2.1) может быть представлено в виде (2.4), где матрицы $R^{1k}(t,t_0)$ могут быть вычислены с использованием рекуррентных соотношений

$$R^{1j}(t,t_0)=\sum_{l=2}^{j}\int_{t_0}^{t}R^{11}(t,\tau)P^{1l}(\tau)R^{lj}(\tau,t_0)d\tau\,, \tag{2.24}$$

$$R^{ij}(t,t_0)=\sum_{l=i+1}^{j}\int_{t_0}^{t}R^{ii}(t,\tau)P^{il}(\tau)R^{lj}(\tau,t_0)d\tau, 2\le i\le K, \tag{2.25}$$

$$R^{ii}(t,t_0)=\left(R^{11}(t,t_0)\right)^{[i]} \tag{2.26}$$

2.4 Оценка нормы разности двух приближённых решений

Рассмотрим первую подсистему системы вида (2.12)

$$\frac{\boldsymbol{dX}^{N_1}}{\boldsymbol{dt}}=\boldsymbol{P}^{N_1}\boldsymbol{X}^{N_1} \tag{2.27}$$

и аналогичную ей первую подсистему той же системы

$$\frac{\boldsymbol{dX}^{N_2}}{\boldsymbol{dt}}=\boldsymbol{P}^{N_2}\boldsymbol{X}^{N_2} \tag{2.28}$$

при $N_2>N_1$.

Рассмотрим первые n компонент решения системы (2.27):

$$\boldsymbol{Y}_{N_1}(t,t_0)=\sum_{\boldsymbol{k=1}}^{N_1}R^{1k}(\boldsymbol{t})\boldsymbol{X}_0^{[k]} \tag{2.29}$$

и первые n компонент решения системы (2.27):

$$\boldsymbol{Y}_{N_2}(t,t_0)=\sum_{\boldsymbol{k=1}}^{N_2}R^{1k}(\boldsymbol{t})\boldsymbol{X}_0^{[k]} \tag{2.30}$$

Построим оценку разности данных решений

$$\delta(\boldsymbol{N_1},\boldsymbol{N_2},t,t_0)=\left\|\boldsymbol{Y}_{N_2}(t,t_0)-\boldsymbol{Y}_{N_1}(t,t_0)\right\|$$

Для этого введём в пространстве прямоугольных вещественных матриц $\left\{A=\left\{a_{ij}\right\}_{i=1,j=1}^{p_1p_2},a_{ij}\in\mathbf{R}\right\}$ норму $\|A\|_\infty=\max\limits_{i=\overline{1,p_1}}\sum_{j=1}^{p_2}|a_{ij}|$. Как известно (см. например []), указанная норма является подчинённой по отношению к векторной норме $\|x\|_\infty=\max\limits_{i=\overline{1,p_1}}|x_i|$, а всякая подчинённая норма является согласованной и мультипликативной, т. е. для неё выполняются условия $\|Ax\|\le\|A\|\|x\|$ и $\|AB\|\le\|A\|\|B\|$.

Покажем, что введённая норма является мультипликативной относительно тензорного и кронекеровского произведения матриц, т. е. $\|A*B\|\le\|A\|\|B\|$.

$$\|A*B\|=\left\|\begin{pmatrix}a_{11}\begin{pmatrix}b_{11}&\cdots&b_{1q}\\\cdots&\cdots&\cdots\\b_{p1}&\cdots&b_{pq}\end{pmatrix}&\cdots&a_{1k}\begin{pmatrix}b_{11}&\cdots&b_{1q}\\\cdots&\cdots&\cdots\\b_{p1}&\cdots&b_{pq}\end{pmatrix}\\\cdots&\cdots&\cdots\\a_{n1}\begin{pmatrix}b_{11}&\cdots&b_{1q}\\\cdots&\cdots&\cdots\\b_{p1}&\cdots&b_{pq}\end{pmatrix}&\cdots&a_{nk}\begin{pmatrix}b_{11}&\cdots&b_{1q}\\\cdots&\cdots&\cdots\\b_{p1}&\cdots&b_{pq}\end{pmatrix}\end{pmatrix}\right\|$$

$$=\left\|\begin{pmatrix}a_{11}b_{11}&\cdots&a_{11}b_{1q}&\cdots&a_{1k}b_{11}&\cdots&a_{1k}b_{1q}\\\cdots&\cdots&\cdots&&\cdots&\cdots&\cdots\\a_{11}b_{p1}&\cdots&a_{11}b_{pq}&\cdots&a_{1k}b_{p1}&\cdots&a_{1k}b_{pq}\\\cdots&&\cdots&&\cdots&&\\a_{n1}b_{11}&\cdots&a_{n1}b_{1q}&\cdots&a_{nk}b_{11}&\cdots&a_{nk}b_{1q}\\\cdots&\cdots&\cdots&&\cdots&\cdots&\cdots\\a_{n1}b_{p1}&\cdots&a_{n1}b_{pq}&\cdots&a_{nk}b_{p1}&\cdots&a_{nk}b_{pq}\end{pmatrix}\right\|$$

$$=\max\begin{pmatrix}|a_{11}b_{11}|+\cdots+|a_{1k}b_{1q}|\\\cdots\\|a_{n1}b_{p1}|+\cdots+|a_{nk}b_{pq}|\end{pmatrix}$$

$$=\max\begin{pmatrix}|a_{11}|(|b_{11}|+\cdots+|b_{1q}|)+\cdots+|a_{1k}|(|b_{11}|+\cdots+|b_{1q}|)\\\cdots\\|a_{n1}|(|b_{p1}|+\cdots+|b_{pq}|)+\cdots+|a_{nk}|(|b_{p1}|+\cdots+|b_{pq}|)\end{pmatrix}$$

$$=\max\begin{pmatrix}(|b_{11}|+\cdots+|b_{1q}|)(|a_{11}|+\cdots+|a_{1k}|)\\\cdots\\(|b_{p1}|+\cdots+|b_{pq}|)(|a_{n1}|+\cdots+|a_{nk}|)\end{pmatrix}$$

$$=\max\begin{pmatrix}(|a_{11}|+\cdots+|a_{1k}|)\\\cdots\\(|a_{n1}|+\cdots+|a_{nk}|)\end{pmatrix}\max\begin{pmatrix}(|b_{11}|+\cdots+|b_{1q}|)\\\cdots\\(|b_{p1}|+\cdots+|b_{pq}|)\end{pmatrix}=\|A\|\|B\|$$

Очевидно, что свойство мультипликативности нормы относительно тензорного произведения сохраняет силу и для кронекеровского произведения, т.к. сложение столбцов в матрице

$$\begin{pmatrix} a_{11}b_{11} & \cdots & a_{11}b_{1q} & \cdots & a_{1k}b_{11} & \cdots & a_{1k}b_{1q} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{11}b_{p1} & \cdots & a_{11}b_{pq} & \cdots & a_{1k}b_{p1} & \cdots & a_{1k}b_{pq} \\ \cdots & & \cdots & & \cdots & & \\ a_{n1}b_{11} & \cdots & a_{n1}b_{1q} & \cdots & a_{nk}b_{11} & \cdots & a_{nk}b_{1q} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1}b_{p1} & \cdots & a_{n1}b_{pq} & \cdots & a_{nk}b_{p1} & \cdots & a_{nk}b_{pq} \end{pmatrix}$$

приведёт лишь к замене знака равенства на знак $\leq$ в равенстве

$$\left\| \begin{pmatrix} a_{11}b_{11} & \cdots & a_{11}b_{1q} & \cdots & a_{1k}b_{11} & \cdots & a_{1k}b_{1q} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{11}b_{p1} & \cdots & a_{11}b_{pq} & \cdots & a_{1k}b_{p1} & \cdots & a_{1k}b_{pq} \\ \cdots & & \cdots & & \cdots & & \\ a_{n1}b_{11} & \cdots & a_{n1}b_{1q} & \cdots & a_{nk}b_{11} & \cdots & a_{nk}b_{1q} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1}b_{p1} & \cdots & a_{n1}b_{pq} & \cdots & a_{nk}b_{p1} & \cdots & a_{nk}b_{pq} \end{pmatrix} \right\| = \max \begin{pmatrix} |a_{11}b_{11}| + \cdots + |a_{1k}b_{1q}| \\ \cdots \\ |a_{n1}b_{p1}| + \cdots + |a_{nk}b_{pq}| \end{pmatrix},$$

а удаление одинаковых строк в данной матрице не изменяет данного равенства.

Введём обозначения:

$$L = \max_{t,\tau \in J} \max_{1 \leq i \leq n} \sum_{j=1}^{k} \left| R_{ij}^{11}(t,\tau) \right| < \infty, \tag{2.31}$$

$$\left\| P^{ij}(t) \right\| \leq \frac{\phi(t)}{j!}, t \in J. \tag{2.32}$$

Здесь $\phi(t)$ – некоторая неотрицательная функция, заданная на промежутке изменения переменной t.

Нетрудно заметить, что $L = \max\limits_{t,\tau \in J} \|R^{11}(t,\tau)\|$.

Введём обозначение

$$M = \int_J \phi(t)\, dt \tag{2.33}$$

Для случая автономной системы ($P^{ij}(t) = P^{ij}$) в качестве функции $\phi(t)$ может быть использована константа

$$\phi(t) = P_{max} \tag{2.34}$$

Алгоритм нахождения этой константы P_{max} приведён ниже.

Если при этом полагать, что $J = [0, T]$, то

$$M = P_{max} T \tag{2.35}$$

Для оценки по норме разности $\left\| \boldsymbol{Y}_{N_2}(t, t_0) - \boldsymbol{Y}_{N_1}(t, t_0) \right\|$ нам необходимо оценить по норме $R^{1k}(t,\tau)$. Для оценки $\|R^{1k}(t,\tau)\|$ оценим по норме $R^{ik}(t,\tau), i,k = \overline{1, N_2}$.

Из свойства мультипликативности нормы относительно тензорного произведения получим

$$\left\|R^{ii}(t,\tau)\right\| \le L^i \tag{2.36}$$

Будем оценивать $\left\|R^{ik}(t,\tau)\right\|$ для произвольного фиксированного k по i от k-1 до 1. Воспользуемся методом математической индукции.

1) i=k

$$\|R^{kk}(t,\tau)\| \le L^k$$

2) i=k-1

$$\|R^{k-1\,k}(t,\tau)\| = \left\|\int_{t_0}^{t} R^{k-1\,k-1}(t,\tau)P^{k-1\,k}(\tau)R^{kk}(\tau,t_0)d\tau\right\| \le L^{k-1}L^k\left\|\int_{t_0}^{t} P^{k-1\,k}(\tau)d\tau\right\| \le \frac{L^{2k-1}M}{k!}$$

3) i=k-2

$$\|R^{k-2\,k}(t,\tau)\| = \left\|\int_{t_0}^{t} R^{k-2\,k-2}(t,\tau)P^{k-2\,k-1}(\tau)R^{k-1\,k}(\tau,t_0)d\tau + \int_{t_0}^{t} R^{k-2\,k-2}(t,\tau)P^{k-2\,k}(\tau)R^{k\,k}(\tau,t_0)d\tau\right\| \le \frac{L^{k-2}M}{k!}\left(\frac{L^{2k-1}M}{(k-1)!}+L^k\right) = \frac{L^{2k-2}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)$$

4) i=k-3

$$\|R^{k-3\,k}(t,\tau)\| = \left\|\int_{t_0}^{t} R^{k-3\,k-3}(t,\tau)P^{k-3\,k-2}(\tau)R^{k-2\,k}(\tau,t_0)d\tau + \int_{t_0}^{t} R^{k-3\,k-3}(t,\tau)P^{k-3\,k-1}(\tau)R^{k-1\,k}(\tau,t_0)d\tau + \int_{t_0}^{t} R^{k-3\,k-3}(t,\tau)P^{k-3\,k}(\tau)R^{k\,k}(\tau,t_0)d\tau\right\| \le L^{k-3}\left(\frac{M}{(k-2)!}\frac{L^{2k-2}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)+\frac{M}{(k-1)!}\frac{L^{2k-1}M}{k!}+\frac{M}{k!}L^k\right) = \frac{L^{2k-3}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\left(\frac{L^{k-2}M}{(k-2)!}+1\right)$$

5) Индукционное предположение

$$\left\|R^{k-i\,k}(t,\tau)\right\| \le \frac{L^{2k-i}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\left(\frac{L^{k-2}M}{(k-2)!}+1\right)\cdots\left(\frac{L^{k-i+1}M}{(k-i+1)!}+1\right) \qquad (2.37)$$

6) Покажем, что для *i-1* справедлива оценка

$$\left\|R^{k-i-1\,k}(t,\tau)\right\| \le \frac{L^{2k-i-1}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\left(\frac{L^{k-2}M}{(k-2)!}+1\right)\cdots\left(\frac{L^{k-i+1}M}{(k-i+1)!}+1\right)\left(\frac{L^{k-i}M}{(k-i)!}+1\right)$$

$$\begin{aligned}\left\|R^{k-i-1\,k}(t,\tau)\right\| &= \left\|\int_{t_0}^{t} R^{k-i-1\,k-i-1}(t,\tau)P^{k-i-1\,k-i}(\tau)R^{k-i\,k}(\tau,t_0)d\tau\right. \\ &+ \int_{t_0}^{t} R^{k-i-1\,k-i-1}(t,\tau)P^{k-i-1\,k-i+1}(\tau)R^{k-i+1\,k}(\tau,t_0)d\tau \\ &+ \int_{t_0}^{t} R^{k-i-1\,k-i-1}(t,\tau)P^{k-i-1\,k-i+2}(\tau)R^{k-i+2\,k}(\tau,t_0)d\tau + \cdots \\ &+ \int_{t_0}^{t} R^{k-i-1\,k-i-1}(t,\tau)P^{k-i-1\,k-1}(\tau)R^{k-1\,k}(\tau,t_0)d\tau \\ &\left.+ \int_{t_0}^{t} R^{k-i-1\,k-i-1}(t,\tau)P^{k-i-1\,k}(\tau)R^{k\,k}(\tau,t_0)d\tau\right\| \le \end{aligned}$$

$$\le L^{k-i-1}\left(\frac{M}{(k-i)!}\frac{L^{2k-i}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\left(\frac{L^{k-2}M}{(k-2)!}+1\right)\cdots\left(\frac{L^{k-i+1}M}{(k-i+1)!}+1\right)+\right.$$

$$\frac{M}{(k-i+1)!}\frac{L^{2k-i+1}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\left(\frac{L^{k-2}M}{(k-2)!}+1\right)\cdots\left(\frac{L^{k-i+2}M}{(k-i+2)!}+1\right)+\cdots+$$

$$\frac{M}{(k-3)!}\frac{L^{2k-3}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\left(\frac{L^{k-2}M}{(k-2)!}+1\right)$$

$$\frac{M}{(k-2)!}\frac{L^{2k-2}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)+$$

$$\left.\frac{M}{(k-1)!}\frac{L^{2k-1}M}{k!}+\frac{M}{k!}L^{k}\right)=$$

$$= \frac{L^{k-i-1}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\left(\frac{ML^{2k-i}}{(k-i)!}\left(\frac{L^{k-2}M}{(k-2)!}+1\right)\cdots\left(\frac{L^{k-i+1}M}{(k-i+1)!}+1\right)+\right.$$
$$\frac{ML^{2k-i+1}}{(k-i+1)!}\left(\frac{L^{k-2}M}{(k-2)!}+1\right)\cdots\left(\frac{L^{k-i+2}M}{(k-i+2)!}+1\right)+\cdots+$$
$$\frac{ML^{2k-3}}{(k-3)!}\left(\frac{L^{k-2}M}{(k-2)!}+1\right)+$$
$$\left.\frac{ML^{2k-2}}{(k-2)!}+L^{k}\right)=$$

$$= \frac{L^{k-i-1}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\left(\frac{L^{k-2}M}{(k-2)!}+1\right)\left(\frac{ML^{2k-i}}{(k-i)!}\left(\frac{L^{k-3}M}{(k-3)!}+1\right)\cdots\left(\frac{L^{k-i+1}M}{(k-i+1)!}+1\right)+\right.$$
$$\frac{ML^{2k-i+1}}{(k-i+1)!}\left(\frac{L^{k-3}M}{(k-3)!}+1\right)\cdots\left(\frac{L^{k-i+2}M}{(k-i+2)!}+1\right)+\cdots+$$
$$\frac{ML^{2k-3}}{(k-3)!}+$$
$$\left.+L^{k}\right)=\cdots$$

$= \frac{L^{2k-i-1}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\left(\frac{L^{k-2}M}{(k-2)!}+1\right)\cdots\left(\frac{L^{k-i+1}M}{(k-i+1)!}+1\right)\left(\frac{L^{k-i}M}{(k-i)!}+1\right)$, ч. т. д.

Таким образом, справедливость оценки (2.37) доказана.

В частности, для $i=k-1$, получим:

$$\|R^{1k}(t,\tau)\| \le \frac{L^{k+1}M}{k!}\left(\frac{L^{k-1}M}{(k-1)!}+1\right)\cdots\left(\frac{L^{2}M}{2!}+1\right) \tag{2.38}$$

Введём обозначения:

$$Q(L,p,q) = \left(\frac{L^{p}M}{p!}+1\right)\cdots\left(\frac{L^{q}M}{q!}+1\right), p > q;\ p,q \in \mathbf{N} \tag{2.39}$$

$$Q(L) = \prod_{i=1}^{\infty}\left(\frac{L^{i}M}{i!}+1\right) \tag{2.40}$$

Отметим, что бесконечное произведение (2.40) имеет смысл (сходится), т.к. $\frac{L^{i}M}{i!} \to 0$, что является достаточным условием сходимости бесконечного произведения вида $\prod_{i=1}^{\infty}(x_i+1)$.

С учётом обозначения (2.39) оценка (2.38) может быть записана в виде

$$\|R^{1k}(t,\tau)\| \le \frac{L^{k+1}M}{k!}Q(L,k-1,2) \le \frac{L^{k+1}M}{k!}Q(L) \tag{2.41}$$

Теперь оценим $\left\|\boldsymbol{Y}_{N_2}(t,t_0)-\boldsymbol{Y}_{N_1}(t,t_0)\right\|$ для $\|x_0\| \le r$:

$$\delta(N_1,N_2,t,t_0,r)=\left\|\boldsymbol{Y}_{N_2}(t,t_0)-\boldsymbol{Y}_{N_1}(t,t_0)\right\|=\left\|\sum_{j=N_1+1}^{N_2}R^{1j}(t,t_0)(x_0)^{[j]}\right\|$$

$$\leq\sum_{j=N_1+1}^{N_2}\left\|R^{1j}(t,t_0)\right\|\left\|(x_0)^{[j]}\right\|\leq\sum_{j=N_1+1}^{N_2}\left\|R^{1j}(t,t_0)\right\|r^j\leq$$

$$M\left(\frac{L^{N_1+2}r^{N_1+1}}{(N_1+1)!}\left(\frac{L^{N_1}M}{N_1!}+1\right)\left(\frac{L^{N_1-1}M}{(N_1-1)!}+1\right)\cdots\left(\frac{L^2M}{2!}+1\right)+\right.$$

$$\frac{L^{N_1+3}r^{N_1+2}}{(N_1+2)!}\left(\frac{L^{N_1+1}M}{(N_1+1)!}+1\right)\left(\frac{L^{N_1}M}{N_1!}+1\right)\left(\frac{L^{N_1-1}M}{(N_1-1)!}+1\right)\cdots\left(\frac{L^2M}{2!}+1\right)+\cdots+$$

$$\left.\frac{L^{N_2+1}r^{N_2}}{N_2!}\left(\frac{L^{N_2-1}M}{(N_2-1)!}+1\right)\cdots\left(\frac{L^2M}{2!}+1\right)\right)=$$

$$=M\frac{L^{N_1+2}r^{N_1+1}}{(N_1+1)!}\left(\frac{L^{N_1}M}{N_1!}+1\right)\left(\frac{L^{N_1-1}M}{(N_1-1)!}+1\right)\cdots\left(\frac{L^2M}{2!}+1\right)\left(1+\frac{Lr}{N_1+2}\left(\frac{L^{N_1+1}M}{(N_1+1)!}+1\right)\right.$$

$$+\frac{(Lr)^2}{(N_1+2)(N_1+3)}\left(\frac{L^{N_1+2}M}{(N_1+2)!}+1\right)\left(\frac{L^{N_1+1}M}{(N_1+1)!}+1\right)+\cdots$$

$$\left.+\frac{(Lr)^{N_2-N_1-1}}{(N_1+2)\cdots N_2}\left(\frac{L^{N_2-1}M}{(N_2-1)!}+1\right)\cdots\left(\frac{L^{N_1+1}M}{(N_1+1)!}+1\right)\right)$$

Таким образом, норма разности двух решений $\delta(N_1,N_2,t,t_0)$ с начальными значениями $\|x_0\|\leq r$ может быть оценена следующим образом

$$\delta(N_1,N_2,t,t_0,r)$$

$$\leq M\frac{L^{N_1+2}r^{N_1+1}}{(N_1+1)!}Q(L,N_1,2)\left(1+\frac{Lr}{N_1+2}Q(L,N_1+1,N_1+1)\right.$$

$$+\frac{(Lr)^2}{(N_1+2)(N_1+3)}Q(L,N_1+2,N_1+1)$$

$$\left.+\cdots\frac{(Lr)^{N_2-N_1-1}}{(N_1+2)\cdots N_2}Q(L,N_2-1,N_1+1)\right)\quad(2.42)$$

Для оценивания нормы разности приближённого и точного решений системы ОДУ ослабим оценку для $\delta(N_1,N_2,t,t_0)$:

$$\delta(N_1, N_2, t, t_0, r)$$

$$\leq M\frac{L^{N_1+2}r^{N_1+1}}{(N_1+1)!}Q(L, N_1, 2)\left(Q(L) + \frac{Lr}{1}Q(L) + \frac{(Lr)^2}{2}Q(L) + \frac{(Lr)^3}{2\cdot 3}Q(L)\right.$$

$$\left. + \cdots\frac{(Lr)^{N_2-N_1-1}}{2\cdot 3\cdots(N_2-N_1-1)}Q(L)\right)$$

$$\leq M\frac{(Lr)^{N_1+2}}{(N_1+1)!\,r}Q(L)^2\left(1 + \frac{Lr}{1} + \frac{(Lr)^2}{2!} + \frac{(Lr)^3}{3!} + \cdots\frac{(Lr)^{N_2-N_1-1}}{(N_2-N_1-1)!}\right)$$

$$\leq M\frac{(Lr)^{N_1+2}}{(N_1+1)!\,r}Q(L)^2\sum_{k=0}^{\infty}\frac{(Lr)^k}{k!} = M\frac{(Lr)^{N_1+2}}{(N_1+1)!\,r}Q(L)^2e^{Lr}$$

Введём обозначение

$$C_1(T) = \frac{MLQ(L)^2e^{Lr}}{r} \tag{2.43}$$

Используя это обозначение, для произвольного $\widetilde{N} > 0$ и для любых $N_1, N_2 > \widetilde{N}$ получим

$$\delta(N_1, N_2, t, t_0, r) \leq C_1(T)\frac{(Lr)^{\widetilde{N}+1}}{(\widetilde{N}+1)!} \tag{2.44}$$

Оценим правую часть неравенства (2.44)

$$C_1(T)\frac{(Lr)^{\widetilde{N}+1}}{(\widetilde{N}+1)!\,r} = C_1(T)\left(\frac{Lr}{1}\frac{Lr}{2}\cdots\frac{Lr}{[Lr]}\right)\left(\frac{Lr}{[Lr]+1}\cdots\frac{Lr}{\widetilde{N}+1}\right)$$

$$= C_1(T)\frac{(Lr)^{[Lr]}}{[Lr]!}\left(\frac{Lr}{[Lr]+1}\right)^{\widetilde{N}+1-[Lr]}$$

Таким образом, для произвольного сколь угодно малого положительного $\varepsilon > 0$

$$\delta(N_1, N_2, t, t_0) < \varepsilon$$

для любых $N_1, N_2 > N(\varepsilon)$

где

$$N(\varepsilon) = log_{\frac{Lr}{[Lr]+1}}\left(\frac{[Lr]!\,\varepsilon r}{(Lr)^{[Lr]}MLQ(L)^2e^{Lr}}\right) + [Lr] - 1 \tag{2.45}$$

Из данной оценки следует, что последовательность решений $\boldsymbol{Y}_k(t, t_0)$ является фундаментальной, а следовательно, сходится в пространстве непрерывных функций.

Из оценки для номера (2.45) можно получить представление для шага интегрирования T для фиксированного значения точности ε и номера шага $\widetilde{N} \in \mathbf{N}$

$$C_1(T)\frac{(Lr)^{[Lr]}}{[Lr]!}\left(\frac{Lr}{[Lr]+1}\right)^{\widetilde{N}+1-[Lr]} = \frac{P_{max}TLQ(L)^2e^{Lr}}{r}\frac{(Lr)^{[Lr]}}{[Lr]!}\left(\frac{Lr}{[Lr]+1}\right)^{\widetilde{N}+1-[Lr]} < \varepsilon$$

Отсюда следует

$$T < \frac{[Lr]!\,\varepsilon r}{P_{max} L Q(L)^2 e^{Lr} (Lr)^{[Lr]}} \left(\frac{[Lr]+1}{Lr} \right)^{\tilde{N}+1-[Lr]} \tag{2.46}$$

Построим алгоритм нахождения константы P_{max} из формулы (2.34) для случая автономных систем ($P^{1k}(t) = P^{1k}, k = \overline{1,N}$). Рассмотрим два случая.

В первом случае известны матрицы $P^{1k}, k = \overline{1,N}$, а также вычислены все матрицы $P^{ik}, i = \overline{2,K}, k = \overline{1,K+N}$.

Вычислим следующие константы

$$D_k = k! \max_{i=\overline{1,K}} \left\| P^{ik} \right\| \tag{2.47}$$

В этом случае, полагая

$$P_{max} = \max_{k=\overline{1,K}} D_k \tag{2.48}$$

нетрудно убедиться в выполнении неравенства (2.32).

Во втором случае известны матрицы $P^{1k}, k = \overline{1,N}$, а матрицы $P^{ik}, i = \overline{2,K}, k = \overline{1,K+N}$ не вычислены (речь идёт о использовании оценок точности для приближённых решений, рассчитанных методами в которых не используются матрицы P^{ik}). В том случае также можно применять формулы (2.47) – (2.48), однако вместо норм $\left\| P^{ik} \right\|, k \neq 1$ следует использовать оценки сверху для этих норм. С использованием математической индукции, формулы (2.6) и доказанных выше свойств нормы кронекеровского произведения матриц, нетрудно показать, что справедлива оценка

$$\left\| P^{ik} \right\| \leq i \left\| P^{1\,k-i+1} \right\|, k > 1 \tag{2.49}$$

2.5 Алгоритм оценивания минимально необходимого порядка решения для заданной точности, промежутка интегрирования для заданной точности и порядка решений, а также оценивания разности двух приближенных решений или разности точного и приближённого решений

1. Задано значение радиуса для начальных значений $\|x_0\| \leq r$
2. Задано значение точности (отличие по норме от точного решения) $\varepsilon > 0$
3. Задано значение длины промежутка T.
4. Расчёт параметра P_{max} по формулам (2.47) – (2.48). В случае, если матрицы $\left\| P^{ik} \right\|, k \neq 1$ не вычислены, вместо этих норм в формуле (2.47) используются оценки для них (2.49)
5. Расчёт параметра M по формуле (2.35)
6. Расчёт матрицы $R^{k11}(\tau, t_0)$ как матричной экспоненты по формуле (2.15)
7. Расчёт константы L по формуле (2.31)
8. Приближённое вычисление значения $Q(L)$ по формуле (2.40), в которой бесконечность следует заменить на некоторое конечное число (достаточно большое)

9 Расчёт минимально необходимого порядка решения по формуле (2.45)

10. При необходимости оценки разности точного решения и приближённого решения и приближённого решения порядка $\widetilde{N}$ (если значение точности $\varepsilon > 0$ не задано) используется формула (2.44)

11. При необходимости оценки разности двух решений порядков N_1, N_2 по формуле (2.42), в которой значения $Q(L, p, q)$ вычисляются по формуле (2.39).

12. По формуле (2.46) можно найти значение промежутка интегрирования при заданной точности и заданном порядке.

3 ПОСТРОЕНИЕ ПОЛИНОМИАЛЬНОЙ НЕЙРОННОЙ СЕТИ И ОБУЧЕНИЕ НА ДАННЫХ

В данной главе рассматривается построение архитектуры и схемы обучения рекуррентной полиномиальной нейронной сети, моделирующей динамические процессы, описываемые автономными и неавтономными системами обыкновенных дифференциальных уравнений. При обучении сети инициализация матриц весовых коэффициентов может быть произвольной (чаще всего матрицы нелинейных порядков инициализируются нулями, а матрица линейной части разложения – единичной матрицей) или с использованием матриц, найденных из нестационарной динамической системы в соответствии с алгоритмом, изложенным в Главе 2. В процессе обучения могут использоваться различные способы регуляризации, представленные ниже.

Необходимость дополнительного обучения построенных в Главе 2 матриц R^j обусловлена следующими факторами:

- ОДУ может быть восстановлена по данным не точно из-за особенностей методов восстановления;
- ОДУ может приближенно описывать моделируемый процесс;
- В реальных условиях на процесс могут влиять различные внешние силы, не учтенные в математической модели, по тем или иным причинам.

3.1 Архитектура полиномиального нейронного слоя

Архитектура полиномиального слоя определяется тремя параметрами:

- Размерность входного вектора (input_dim)
- Размерность выходного вектора (output_dim)
- Порядок нелинейности (order)

Размерность входного вектора обычно соответствует размерности фазового вектора нестационарной системы обыкновенных дифференциальных уравнений (см. Глава 1), но может включать в себя также и экзогенные переменные, которые могут выступать в роли параметров управления. Размерность выходного вектора определяется количеством измеряемых фазовых переменных, т.е. тех, для которых доступны тренировочные данные (чаще всего их количество совпадает с размерностью входного вектора). Порядок нелинейности может быть выбран произвольно в соответствии с особенностями решаемой задачи, или определен с учетом построенных оценок точности, описанных в Главе 2.

Разработанная полиномиальная нейронная сеть относится к классу рекуррентных и осуществляет преобразование входного вектора в выходной по следующему правилу:

$$\boldsymbol{X}(t_{i+1}) = \text{bias} + \sum_{j=1}^{N} R^j \boldsymbol{X}(t_i)^{[j]}, \tag{3.1}$$

где $\boldsymbol{X}(t_i)$ – значения входного вектора, измеренные в момент времени t_i, $\boldsymbol{X}(t_{i+1})$ – прогнозируемые значения выходного вектора на момент времени t_{i+1}, bias – свободный коэффициент (по желанию может быть включен или исключен из состава обучаемых параметров), N – порядок нелинейности, $[j]$ – порядок нелинейности для преобразования входного вектора (может быть преобразован с повторениями – пункт 3.1.1 или без повторений – пункт 3.1.2), R^j $(j = 1..N)$ – весовые матрицы нейронной сети. Предусмотрен механизм инициализации R^j, матрицами, построенными на основе алгоритма, приведенного в Главе 2. При отсутствии инициализирующих матриц по умолчанию все нелинейные порядки инициализируются нулями, а матрица линейного члена разложения – единичной матрицей. Однако, разумеется, в этом случае существенно увеличивается время обучения сети и ухудшаются прогнозные свойства нейросетевой модели.

Для реализации полиномиальной нейронной сети были использованы библиотеки TensorFlow (TF) [39] и Keras для языка Python, имеющая большое количество встроенных инструментов, упрощающих построение и обучение нейросетевых моделей. Кроме того, TensorFlow позволяет использовать graphics processing unit (GPU) для увеличения скорости обучения нейронной сети.

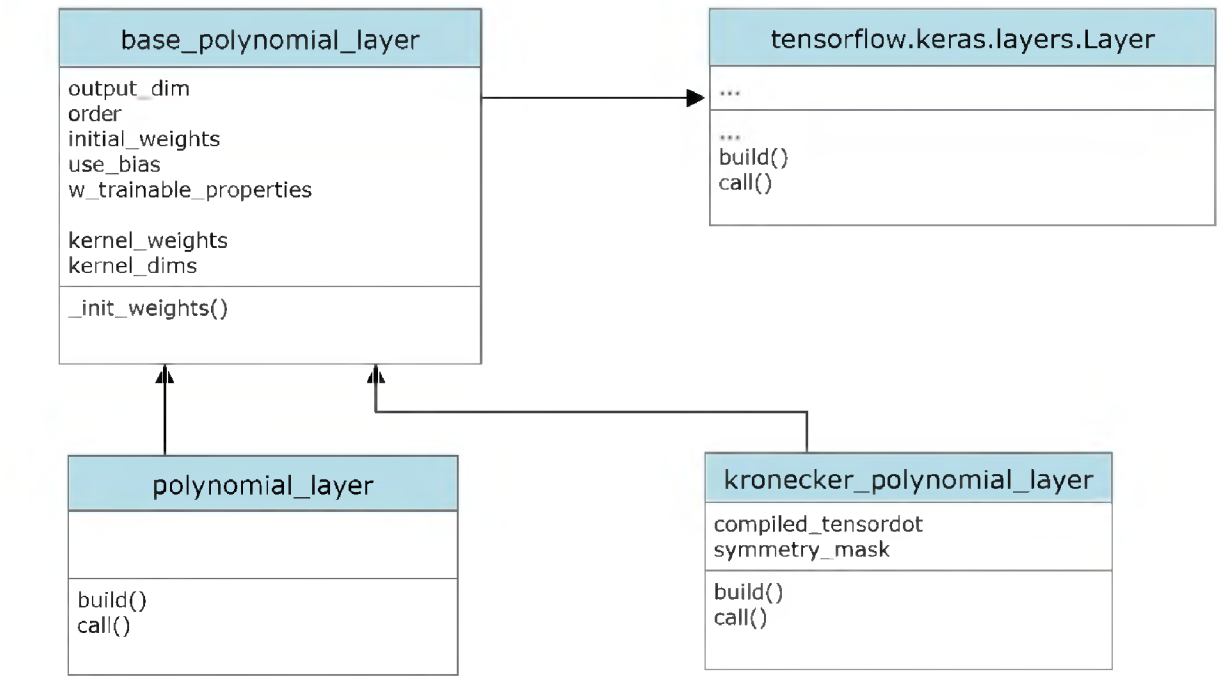

Рисунок 3.1 – Схема наследования классов, имплементирующих полиномиальные слои

3.1.1 Полиномиальный слой

Полиномиальный слой (polynomial_layer) является наследником класса base_polynomial_layer. base_polynomial_layer является потомком базового класса tensorflow.keras.layers.Layer (схема наследования показана на рисунке 3.1). Таким образом, полиномиальный слой переопределяет методы для построения слоя (build) и получения выходного вектора по входному (call) из tensorflow.keras.layers.Layer.

(build) При построении полиномиального слоя происходит вычисление размерностей матриц весовых коэффициентов для каждого порядка нелинейности и инициализация матриц весовых коэффициентов. В данном случае размерность *j*-ой матрицы определяется как $[n^j, m], j = 1..N$, где n – размерность входного вектора, m - размерность выходного вектора.

Первая размерность матриц весовых коэффициентов соответствует количеству комбинаций компонент входного вектора, для образования *j*-ого порядка нелинейности (с учетом повторений).

(call) В соответствии с формулой (3.1) для получения выходного вектора необходимо предварительно расширить входное пространство с помощью полиномиального нелинейного преобразования входного вектора. Данный шаг был реализован с помощью рекуррентного вычисления тензорного произведения по формуле

$$\boldsymbol{X}^k = \boldsymbol{X}^{k-1} \otimes \boldsymbol{X} \tag{3.2}$$

с использованием функция *einsum* из библиотеки TensorFlow, применяющей соглашение Эйнштейна о суммировании индексов своих аргументов.

Вычисление значений выходного вектора описывается следующим алгоритмом:

1. Инициализация $\overline{\boldsymbol{X}} = bias$;
2. В цикле (*j*) от 1 до *n*:
    a. Перемножить матрицу R^j на вектор $\boldsymbol{X}^j(t_{i-1})$ и прибавить результат к вектору $\overline{\boldsymbol{X}}$;
    b. Возвести вектор $\boldsymbol{X}(t_{i-1})$ в степень *j +1* по формуле (3.2);

Стоит отметить, что реализация полиномиального слоя не предполагает использование кронекеровских степеней вектора (с сокращением размерности). Например, для входного вектора размерности 2 при возведении во вторую степень получим:

$$\boldsymbol{Y} = R^2 \cdot \boldsymbol{X}^2 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{pmatrix} \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_2 x_1 \\ x_2^2 \end{pmatrix} \tag{3.3}$$

Как видно из (3.3), при умножении матрицы R^2 на вектор $\boldsymbol{X}^2$ второй и третий элементы вектора $\boldsymbol{Y}$ получаются одинаковыми, что предполагает возможность сокращения размерности вектора $\boldsymbol{Y}$ на единицу. Аналогичная ситуация возникает и для входных векторов большей размерности и использовании более высоких степеней. Для реализации возможности удаления повторяющихся нелинейных членов и сокращения размерности матриц весовых коэффициентов реализован кронекеровский слой, особенности функционирования которого рассмотрены ниже.

3.1.2 Кронекеровский слой

Кронекеровский слой (Kronecker_layer) фактически также является наследником класса tensorflow.keras.layers.Layer и реализует два его абстрактных метода для построения слоя (build) и получения выходного вектора по входному (call) (см. рисунок 3.1). Главной особенностью этого слоя является реализация кронекеровских операций возведения в степень, сопряженных с удалением дублирующихся (с точностью до перестановки сомножителей) элементов в полиномиально преобразованном входном векторе.

(*build*) При построении кронекеровского слоя происходит вычисление размерностей матриц весовых коэффициентов для каждого порядка нелинейности и инициализация матриц весовых коэффициентов. В данном случае размерность j-ой матрицы определяется как $[C_j^{n+j-1}, m], j = 1..N$, где n – размерность входного вектора, m - размерность выходного вектора, $C_j^{n+j-1} = \frac{(n+j-1)!}{j!(n-1)!}$.

Первая размерность матриц весовых коэффициентов соответствует количеству различных (с точностью до перестановки сомножителей) комбинаций компонент входного вектора, для образования j-ого порядка нелинейности.

Дополнительно происходит вычисление набора бинарных масок, которые будут применяться для удаления одинаковых элементов к результату возведения в степень, полученному по формуле (3.2).

(*call*) Аналогично методу call полиномиального слоя с помощью полиномиального нелинейного преобразования входного вектора происходит расширение входного пространства и удаление дубликатов. Например, вычисление второй степени Кронекера для входного вектора размерности два производится по формуле

$$\boldsymbol{X}^{[2]} = \boldsymbol{X}^{\boldsymbol{2}} \odot^* mask = \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_2 x_1 \\ x_2^2 \end{pmatrix} \odot^* \begin{pmatrix} True \\ True \\ False \\ True \end{pmatrix} = \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix}.$$

Здесь $\odot^*$ - произведение Адамара с сокращением размерности получаемого вектора (удаляются все элементы вектора, у которых в булевой маске на соответствующем месте стоит False). В этом случае формула (3.3) примет вид

$$\boldsymbol{Y} = \hat{R}^2 \cdot \boldsymbol{X}^{[2]} = \begin{pmatrix} \hat{r}_{11} & \hat{r}_{12} & \hat{r}_{13} \\ \hat{r}_{21} & \hat{r}_{22} & \hat{r}_{23} \\ \hat{r}_{31} & \hat{r}_{32} & \hat{r}_{33} \end{pmatrix} \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix}$$

Матрица $\hat{R}^2$ отличается от матрицы R^2 сокращением размерности.

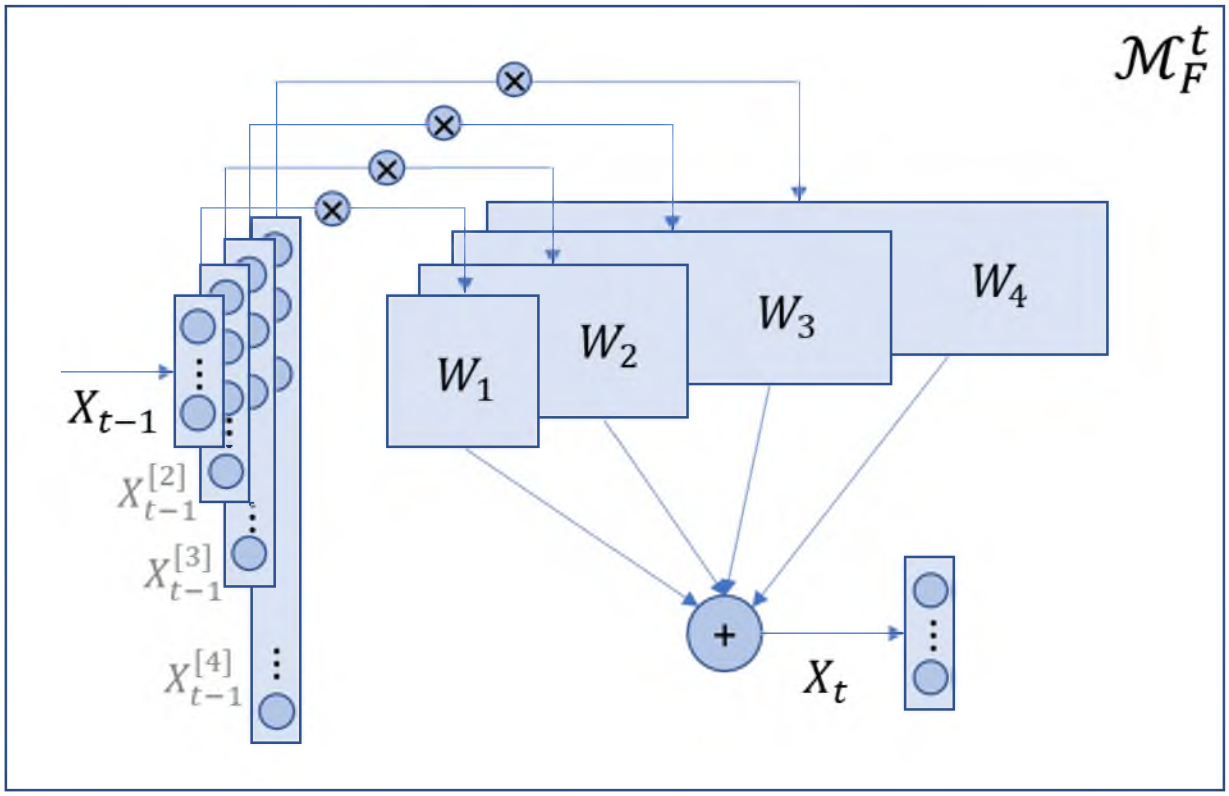

Рисунок 3.2 – Схема работы кронекеровского слоя в соответствии с формулой (3.1)

3.2 Полиномиальные модели

На основе описанных выше типов полиномиальных слоев (с/без повторений нелинейных компонент) можно стоить модели, подходящие для различных типов задач. В силу того, что Кронекеровский слой обеспечивает снижение размерности матриц весовых коэффициентов, а следовательно, сокращает количество свободных параметров для оптимизатора и увеличивает скорость вывода и обучения слоя, он является приоритетным строительным элементом для полиномиальных моделей, рассмотренных ниже.

3.2.1 Полиномиальная модель с одним выходом (Single output model)

Модель с одним выходом может строиться для стационарных и нестационарных систем. В стационарном случае матрицы весовых коэффициентов зависят только от дискретности временного ряда и не зависят от конкретного момента времени, поэтому такая модель содержит единственный кронекеровский слой (рисунок 3.2 а)). В случае нестационарной

системы можно попытаться привести систему к стационарному виду с помощью введения в состав фазового вектора экзогенных переменных, зависящих от времени, или построить многослойную модель, соответствующую, например, периодичности данных (рисунок 3.2 б)).

Для обучения модели с одним выходом применяется следующий алгоритм.

I подготовка данных

Алгоритм формирования набора данных для обучения полиномиальной нейронной сети с одним выходом.

1. Известны данные, полученные в результате измерения на k траекториях моделируемой динамической системы:

$$\boldsymbol{X}_1(t_1), \dots, \boldsymbol{X}_1(t_{p_1}), \boldsymbol{X}_2(t_{p_1+1}), \dots, \boldsymbol{X}_2(t_{p_1+p_2}), \dots, \boldsymbol{X}_k(t_{p_1+\dots+p_{k-1}+1}), \dots, \boldsymbol{X}_k(t_{p_1+\dots+p_k}),$$

Здесь $t_{p_1+\dots+p_i+J} < t_{p_1+\dots+p_i+J+1}, \forall j < p_{i+1}$, p_i – количество векторов состояния системы, измеренных на i-й траектории с начальным вектором $\boldsymbol{X}_i(t_{p_1+\dots+p_{i-1}+1})$

2. Формируется множество обучающих пар $\left\{\left(\boldsymbol{X}_{i+1}(t_{p_1+\dots+p_i+J}), \boldsymbol{X}_{i+1}(t_{p_1+\dots+p_i+J+1})\right), \forall j < p_{i+1}, i = \overline{1,k}\right\}$, где первый вектор пары – обучающий входной вектор, второй вектор пары – обучающий выходной вектор

Далее, первый вектор обучающей пары будем обозначать $\boldsymbol{X}(t_0)$, второй - $\boldsymbol{X}(t_M)$, где M – количество кронекеровских слоев в модели (1 для стационарной системы).

II обучение

Алгоритм работы рекуррентной полиномиальной модели с одним выходом:

1. Вектор $\boldsymbol{X}(t_0)$ подается на вход в кронекеровский слой, вычисляется вектор $\boldsymbol{X}(t_M)$;
2. Полученный вектор $\boldsymbol{X}(t_M)$ подается на вход в кронекеровский слой, вычисляется вектор $\boldsymbol{X}(t_{2M})$;
3. Данный процесс повторяется заданное число раз.

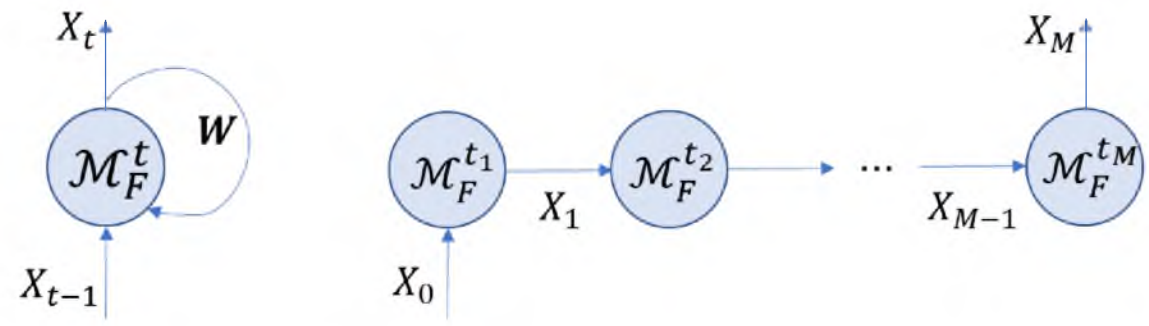

Рисунок 3.3 – Полиномиальная модель с одним выходом: а) стационарная система б) нестационарная система

3.2.2 Полиномиальная модель с несколькими выходами (Multiple output model)

Полиномиальная модель с несколькими выходами состоит из последовательного набора кронекеровских слоев. В случае стационарной модели матрицы весовых коэффициентов фиксируются между слоями и обновляются в процессе обучения синхронно. Для нестационарной модели общая архитектура не меняется, но выход формируется из каждого скрытого слоя модели (рисунок 3.3. б)) [40].

Число скрытых кронекеровских слоев чаще всего определяется или длительностью обучающих данных (в случае, если они заданы набором отдельных траекторий при различных параметрах динамической системы) или периодичностью/сезонностью временного ряда, используемого для обучения.

На рисунке 3.4 представлена модель с несколькими выходами для стационарной системы. В данном случае на вход подается вектор $\boldsymbol{X}(t_0)$, а на выходе получается массив векторов $\boldsymbol{X}(t_1)$, $\mathbf{X}(\mathrm{t}_2)$,…, $\mathbf{X}(\mathrm{t}_q)$. Соответствующим образом происходит подготовка множества тренировочных данных для такой модели.

I подготовка данных

Алгоритм формирования набора данных для обучения полиномиальной нейронной сети с несколькими выходами.

1. Известны данные, полученные в результате измерения на k траекториях моделируемой динамической системы:

$$\boldsymbol{X}_1(t_1), \dots, \boldsymbol{X}_1(t_{p_1}), \boldsymbol{X}_2(t_{p_1+1}), \dots, \boldsymbol{X}_2(t_{p_1+p_2}), \dots, \boldsymbol{X}_k(t_{p_1+\dots+p_{k-1}+1}), \dots, \boldsymbol{X}_k(t_{p_1+\dots+p_k}),$$

Здесь $t_{p_1+\dots+p_i+J} < t_{p_1+\dots+p_i+J+1}, \forall j < p_{i+1}$, p_i – количество векторов состояния системы, измеренных на i-й траектории с начальным вектором $\boldsymbol{X}_{\boldsymbol{i}}(t_{p_1+\dots+p_{i-1}+1})$

2. Формируется множество обучающих пар $\{(\boldsymbol{X}_{\boldsymbol{i}+1}(t_{p_1+\dots+p_i+1}), [\boldsymbol{X}_{\boldsymbol{i}+1}(t_{p_1+\dots+p_i+2}), \dots, \boldsymbol{X}_{\boldsymbol{i}+1}(t_{p_1+\dots+p_i+p_{i+1}})])$, $i = \overline{0, k-1}\}$, где первый вектор пары – обучающий входной вектор, второй вектор пары – список, содержащий набор выходных векторов в каждый момент времени.

Далее, первый вектор обучающей пары будем обозначать $\boldsymbol{X}(t_0)$, второй - $[\boldsymbol{X}(t_1), \dots, \boldsymbol{X}(t_q)]$, где q – количество скрытых кронекеровских слоев в модели (рисунок 3.4).

II обучение

Алгоритм работы рекуррентной полиномиальной модели с несколькими выходами:

1. Вектор $\boldsymbol{X}(t_0)$ подается на вход в кронекеровский слой, вычисляется набор векторов $[\boldsymbol{X}(t_1), \dots, \boldsymbol{X}(t_q)]$;

2. Полученный вектор $\boldsymbol{X}(t_q)$ подается на вход в кронекеровский слой, вычисляется вектор $[\boldsymbol{X}(t_{q+1}), \dots, \boldsymbol{X}(t_{2q})]$;
3. Данный процесс повторяется заданное число раз.

Обучение модели с несколькими выходами позволяет добиться динамической согласованности прогнозов, так как при вычислении градиентов учитывается рассогласование выходов на всей длине траектории динамической системы, а не только между парами соседних точек.

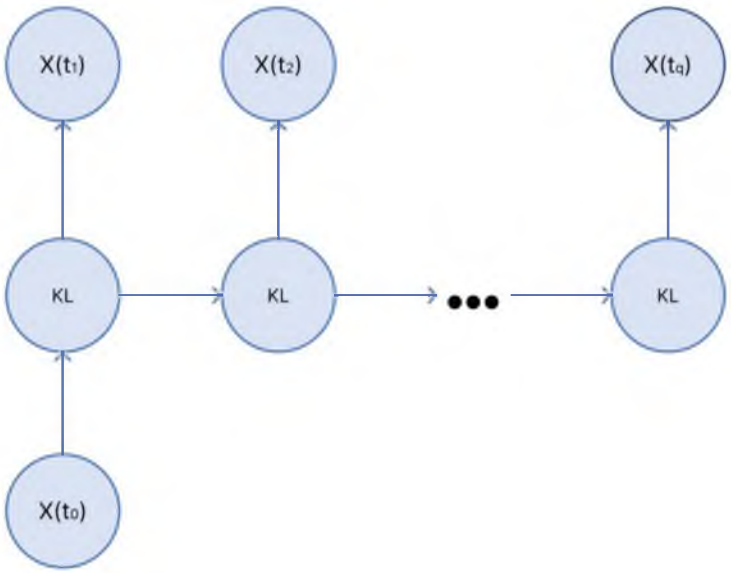

Рисунок 3.4 – Полиномиальная модель с несколькими выходами

3.3 Нормализация данных и инициализирующих весов

Для улучшения результатов обучения полиномиальной нейронной сети матрицы весов (если они были проинициализированы в соответствии с алгоритмами Главы 2) и данные нормализуются. Нормализующий коэффициент вычисляется с использованием нормы L_∞ для вектора начальных данных динамической системы

$$\|\boldsymbol{X}\|_\propto = max|X_i|. \quad (3.4)$$

При этом для нормированных данных должно выполняться соотношение, аналогичное (3.1)

$$\tilde{\boldsymbol{X}}(t_{i+1}) = \tilde{\text{bias}} + \sum_{j=1}^{N} \tilde{R}^j \tilde{\boldsymbol{X}}(t_i)^{[j]},$$

где $\tilde{\cdot}$ – означает нормировку.

Алгоритм нормализации матриц весов следующий

1) Вычисление нормы по формуле (3.4)
2) Умножаем все элементы матрицы R^i на коэффициент $\alpha_i = (\|X\|_\propto)^{i-1}$.

3.4 Особенности обучения

Обучение полиномиальных нейронных сетей может быть реализовано двумя способами:

1) обучение сразу всех весовых матриц:

   Данный подход предполагает обучение всех матриц весовых коэффициентов одновременно, т.е. градиент вычисляется для всех весовых коэффициентов сразу, и на каждом шаге обучения изменяются все матрицы весов. При таком подходе количество свободных коэффициентов может быть велико, что усложняет сходимость оптимизационного алгоритма (особенно при отсутствии грамотной инициализации) и потенциально может привести к физически некорректным прогнозам.

2) последовательное обучение матриц, начиная с bias и матрицы линейного порядка R^1 до матрицы R^N, соответствующей N-ому порядку нелинейности:

   Одновременное обучение матриц весов не учитывает важную особенность полиномиальной модели, вклад матриц весовых коэффициентов в результат (см. формулу (3.1)) различен. Наибольшее влияние оказывает линейная часть R^1, далее при увеличении порядка нелинейности вклад уменьшается. В связи с этим предложен механизм последовательного обучения.

   - строится линейная модель с матрицей R^1, которая обучается до момента, когда перестает уменьшаться функция потерь

   - строится модель второго порядка, линейная часть которой инициализируется обученной матрицей R^1, выгруженной из предыдущей модели. R^1 фиксируется, и обучается только матрица R^2 и т.д.

Для моделей с одним выходом применяется базовый подход к обучению, который предполагает, что на вход нейронной сети подается вектор $\tilde{\boldsymbol{X}}(t_0)$, вычисляется вектор $\tilde{\boldsymbol{X}}(t_1)$, который сравнивается с эталонным вектором $\bar{\boldsymbol{X}}(t_1)$. Затем вычисляется значение функции ошибки $\mathcal{L}$, в качестве которой могут использоваться, например MSE (mean square error), RMSE (root mean square error), RMSLE (Root Mean Squared Log Error):

$$MSE(\tilde{\boldsymbol{X}}(t_1), \bar{\boldsymbol{X}}(t_1)) = \frac{1}{n}\sum_{i=1}^{n}(\tilde{x}_i - \bar{x}_i)^2 \tag{3.5}$$

Здесь $\tilde{x}_i$ – элементы вектора $\tilde{\boldsymbol{X}}(t_1)$, $\bar{x}_i$ – элементы вектора $\bar{\boldsymbol{X}}(t_1)$. Далее вычисляется градиент функции ошибки, использующийся для изменения матриц весов. Одним из основных достоинств такого подхода можно назвать высокую скорость обучения. К недостаткам можно отнести потенциальное нарушение физичности получаемых матриц,

т.е. в процессе обучения не контролируются положения ненулевых коэффициентов, однако это можно исправить, используя различные методы регуляризации.

Для обучения полиномиальных моделей с несколькими выходами применяется следующий подход.

Алгоритм обучения моделей с несколькими выходами:

1) Принимается на вход вектор $\widetilde{\boldsymbol{X}}(t_0)$ и эталонные значения выходных векторов $\overline{\boldsymbol{X}}(t_1)$, $\overline{\boldsymbol{X}}(\mathbf{t_2})$,…, $\overline{\boldsymbol{X}}(\mathrm{t}_q)$;
2) Вычисляется массив векторов $\widetilde{\boldsymbol{X}}(t_1)$, $\widetilde{\mathbf{X}}(\mathrm{t}_2)$,…, $\widetilde{\mathbf{X}}(\mathrm{t}_q)$;
3) Для пары векторов $\widetilde{\boldsymbol{X}}(t_i)$ и $\overline{\boldsymbol{X}}(t_i)$ и вычисляется значение функции ошибки $\mathcal{L}$ для каждой пары при одинаковых t_i;
4) Вычисляется среднее значение ошибки, для этого можно вычислить среднее или средневзвешенное значение ошибки:
   a) $error = \frac{1}{Q-1}\sum_{i=1}^{Q}\mathcal{L}(\widetilde{\boldsymbol{X}}(t_i), \overline{\boldsymbol{X}}(t_i))$,
   b) $error = \frac{1}{Q-1}\sum_{i=1}^{Q}\alpha_i * \mathcal{L}(\widetilde{\boldsymbol{X}}(t_i), \overline{\boldsymbol{X}}(t_i))$,
5) Вычисляется градиент функции ошибки используя функции автоматического дифференцирования библиотеки TensorFlow;
6) Применение градиента в оптимизации весов.

Рассматриваемые дальше подходы применимы как для моделей с одним выходом, так и для моделей с несколькими выходами

3.4.1 Регуляризация весов, маскирование градиентов

Использование масок в процессе обучение связано с требованием сохранять физичность для матриц R^i в задачах, которые описывают физический процесс, а также с необходимость сокращения времени обучения полиномиальной нейронной сети. Данная возможность может быть реализована за счёт обнуления и фиксации во время обучения малых элементов матрицы весов. При этом количество настраиваемых (обучаемых) параметров (весов) модели уменьшается за счёт исключения из процесса обучения нулевых элементов матриц весов.

Для получения маски мы должны знать какие из коэффициентов являются не нулевыми. Это возможно только в том случае если получены значения матриц R^i на этапе инициализации. В этом случае маски R_m^i, размерность которых совпадает с размерностью матриц R^i, строятся так, что на месте ненулевых элементов начальных весов стоят единицы, а на месте нулевых (или близких к нулю) – нули.

Если матрицы R^i не были инициализированы при построении модели, возможно использовать регуляризацию весов L_0 которая будет добавлять к функции потерь $\mathcal{L}$ величину

$$\beta = \lambda \sum g(r_{jk}^i) \tag{3.6}$$

Здесь λ – задаваемый параметр, $g(x) = \begin{cases} 1, x \neq 0 \\ 0, x = 0 \end{cases}$, r_{jk}^i – элементы матриц R^i. График функции (3.6) приведен на рисунке 3.5. Норма L_0 штрафует количество ненулевых элементов матриц весовых коэффициентов и, таким образом, повышает разреженность матриц R.

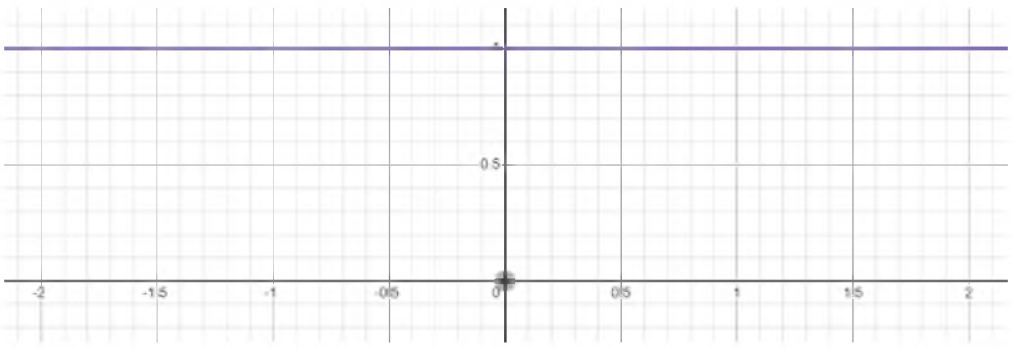

Рисунок 3.5 - L_0-норма

Комбинаторный характер функции (3.6) и ее не дифференцируемость делают оптимизацию с использованием базовых алгоритмов затруднительным, поэтому предлагаются различные способы ослабления дискретного характера штрафа [41–43], для эффективного применения методов оптимизации гладких функций.

3.5 Алгоритмы оптимизации весов

Библиотека TensorFlow предоставляет большой выбор алгоритмов оптимизации (оптимизаторов). Формулы для алгоритмов градиентного спуска выглядят следующим образом

$$\Delta\theta(t+1) = -\eta(t+1)\nabla_\theta J(\theta(t)),$$
$$\theta(t+1) = \theta(t) + \Delta\theta(t+1) = \theta(t) - \eta(t+1)\nabla_\theta J(\theta(t)).$$

Здесь t – номер шага обучения, $\theta(t)$ – набор всех весовых коэффициентов сети, $J(\theta(t))$ – функция ошибки, $\nabla_\theta J(\theta(t))$ – градиент функции ошибки, $\eta(t)$ – коэффициент (шаг) обучения.

В результате численных экспериментов можно сделать вывод, что при обучении полиномиальных нейронных сетей оптимизатор AdaGrad (adaptive gradient algorithm) [44] показывает лучшие результаты, т.к. его алгоритм разработан для улучшения обучения моделей машинного обучения с квадратичными функциями ошибки. Метод AdaGrad возводит в квадрат значения градиентов на предыдущих шагах обучения и суммирует их.

Для уменьшения эффекта быстрого накопления градиентов алгоритм рассчитывает скорость обучения посредством деления накопленных квадратов градиентов на коэффициент с большим значением. За счет этого алгоритм адаптивно уменьшает скорость обучения на крутых уклонах поверхности отклика (функции ошибки).

Формулы для алгоритма AdaGrad выглядят следующим образом:

$$G(\mathrm{t}+1) = \mathrm{G}(\mathrm{t}) + \nabla_\theta J(\theta(t)) \times \nabla_\theta J(\theta(t))$$

$$\theta(\mathrm{t}+1) = \theta(\mathrm{t}) - \frac{\eta(t+1)}{\sqrt{\mathrm{G}(\mathrm{t}+1) + e}} \mathrm{G}(\mathrm{t}+1)$$

Здесь $\times$ - операция поэлементного умножения векторов, e – сглаживающий параметр близкий к нулю, необходимый, чтобы избежать деления на 0.

4 ДЕМОНСТРАЦИЯ РАЗРАБОТАННЫХ АЛГОРИТМОВ НА ПРИМЕРАХ

4.1 Модель динамики заряженных частиц в цилиндрическом дефлекторе

В данном разделе рассмотрим методику предлагаемого метода идентификации нестационарных ОДУ на примере тестовой задачи моделирования динамики заряженных частиц в цилиндрическом дефлекторе [23]. Динамика заряженных частиц в электромагнитных полях может быть описана системой обыкновенных дифференциальных уравнений сложной нелинейной формы. Для простоты, рассмотрим приближение движения заряженной частицы в цилиндрическом дефлекторе

$$\begin{cases} \dot{x} = y \\ \dot{y} = -2x + x^2/p(t) \end{cases} \tag{4.1}$$

где p(t) - радиус поворота равновесной частицы, x - отклонение от этого радиуса, а $\dot{x} = y$ - производная по углу поворота. Предположим, что $p(t)$ – кусочно-постоянная функция:

$$p(t) = \begin{cases} p_1, \text{если } t \in [0, t_1], \\ p_2, \text{если } t \in [t_1, t_2], \\ p_3, \text{если } t \in [t_2, t_3] \end{cases}$$

Решение системы (4.1) предполагается непрерывным, т.е. в каждой точке переключения выполняется условие $\left(x_i(t_i), y_i(t_i)\right) = \left(x_{i+1}(t_i), y_{i+1}(t_i)\right), i = 1..2$.

Выберем набор значений p_i и начальное условие для генерации обучающих данных, с использованием численного решения (4.1) любым стандартным методом интегрирования ОДУ. Рассмотрим промежуток времени $T = [0;\ 3], t_i = [1;\ 2;\ 3], p_i = [10;\ 50;\ 100]$ и $(x_0;\ y_0) = (-2;\ 4)$. Следует отметить, что эта информация используется только для формирования обучающих данных для идентификации неизвестных параметров системы.

Следуя процедуре идентификации правой части ОДУ, описанной в Главе 1, приближенно находим неизвестные матрицы P^{1i} для каждого из интервалов переключений:

$$P_1^{11} = \begin{pmatrix} 4.07 \cdot 10^{-6} & 0.99 \\ -1.99 & 4.38 \cdot 10^{-6} \end{pmatrix} P_1^{12} = \begin{pmatrix} 1.24 \cdot 10^{-7} & 3.28 \cdot 10^{-6} & 2.00 \cdot 10^{-9} \\ 0.09 & -9.71 \cdot 10^{-8} & 3.34 \cdot 10^{-6} \end{pmatrix}$$

$$P_2^{11} = \begin{pmatrix} 2.4 \cdot 10^{-5} & 0.99 \\ -1.99 & 1.35 \cdot 10^{-5} \end{pmatrix} P_2^{12} = \begin{pmatrix} 1.68 \cdot 10^{-7} & 6.70 \cdot 10^{-7} & -4.07 \cdot 10^{-9} \\ 0.02 & -3.56 \cdot 10^{-8} & 6.95 \cdot 10^{-7} \end{pmatrix}$$

$$P_3^{11} = \begin{pmatrix} -4.32 \cdot 10^{-5} & 0.99 \\ -2.00 & 1.45 \cdot 10^{-5} \end{pmatrix} P_3^{12} = \begin{pmatrix} -1.44 \cdot 10^{-7} & 3.91 \cdot 10^{-7} & 6.08 \cdot 10^{-9} \\ 9.99 \cdot 10^{-3} & 1.03 \cdot 10^{-7} & 3.40 \cdot 10^{-7} \end{pmatrix}$$

Максимальный порядок нелинейности в правой части (4.1) равен двум, поэтому следует строить полиномиальную нейронную сеть как минимум второго порядка нелинейности.

Уравнение (4.1) является кусочно-стационарным, поэтому матрицы R^{1i} являются общими для временных шагов в пределах каждого интервала переключения.

Сначала воспользуемся процедурой инициализации весов, описанной в Главе 2. Начнем с восстановленных матриц системы обыкновенных дифференциальных уравнений, представленных выше. Для проверки способности сети к обобщению рассматриваются два тестовых начальных значения: [-2,5; 5] и [-1; 3]. Полиномиальная модель должна восстановить траектории системы, исходящие из этих значений. Результаты сравнивались с нейронной сетью LSTM из-за ее внутренней связи с динамической системой с дискретным временем.

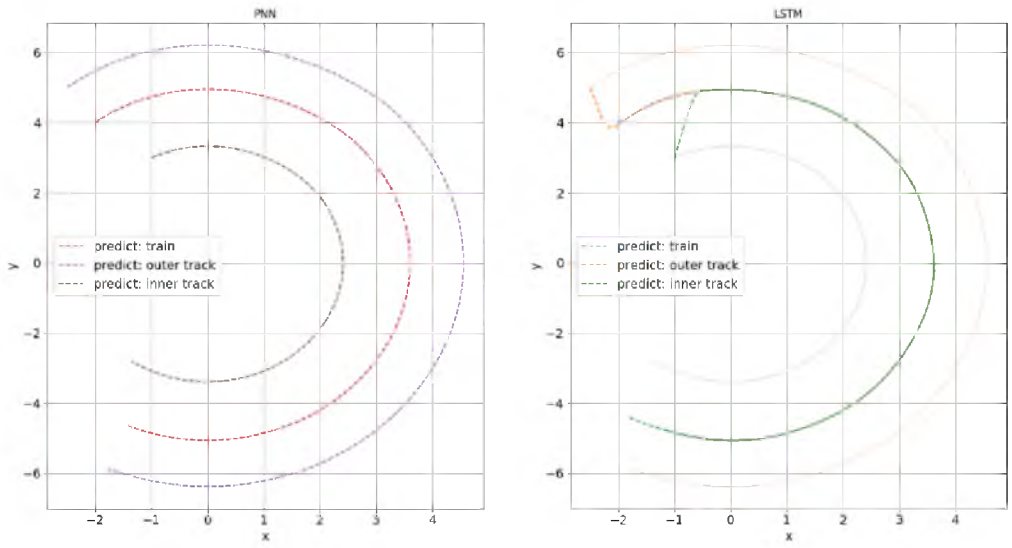

Рисунок 4.1 – Фазовый портрет реконструированной системы с использованием полиномиальной нейронной сети и сети LSTM

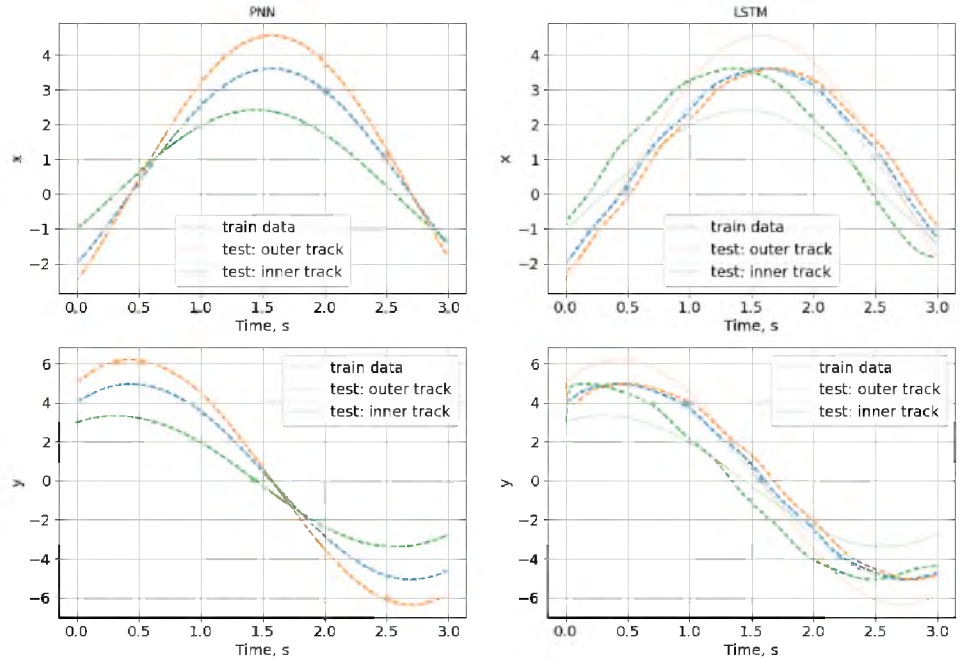

Рисунок 4.2. – Траектории реконструированной системы с использованием полиномиальной нейронной сети (с инициализацией) и сети LSTM

На рисунках 4.1 и 4.2 показано сравнение фазового портрета и траекторий системы соответственно для идентифицированной системы, предсказанных для тренировочных и тестовых начальных значений.

Из рисунков 4.1 и 4.2 видно, что LSTM не может восстановить динамику системы, просто запоминая траекторию, используемую для обучения. Даже для различных начальных условий он может предсказывать только ранее наблюдаемое поведение системы, в то время как полиномиальная нейронная сеть корректно восстанавливает динамику и показывает большую способность к обобщению для начального значения, лежащего за пределами диапазона, используемого для обучения.

Теперь сравним возможности полиномиальной сети и LSTM для других значений параметров переключения, не обеспечивающих гладкости траекторий системы: $p_i = [2; 50; 5]$. Обучение полиномиальной нейронной сети будем проводить без предварительной инициализации матриц весовых коэффициентов. Результаты представлены на рисунке 4.3.

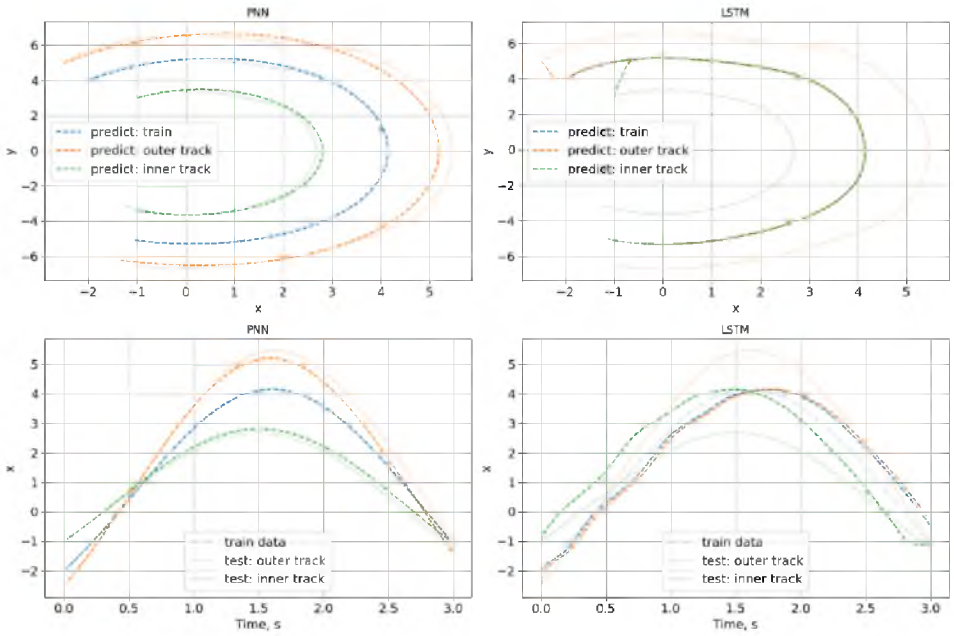

Рисунок 4.3 – Фазовый портрет и *x(t),* сгенерированный обученной полиномиальной нейронной сетью (без инициализации) и сети LSTM

Рисунок 4.3 показывает, что даже при обучении полиномиальной сеть с нулевых начальных весов, можно правильно восстановить динамику системы. Конечно, присутствуют расхождения между прогнозируемой и истинной траекторией для тестовых начальных значений в окрестности точек переключения, но тем не менее в целом поведение системы улавливается корректно. В то же время LSTM снова просто запоминает обучающие данные.

4.2 Модель Лотки-Вольтерры

В данном разделе рассматриваются обобщенные уравнения Лотки-Вольтерра, представляющие динамику популяции в экосистеме двух хищников и одной жертвы [24]. Эта система демонстрирует хаотическое поведение при определенном наборе параметров *a*, *b*, *c*. Таким образом, динамика системы очень чувствительна к начальным условиям [45]. Рассматриваемая модель Лотки-Вольтерры описывается следующей системой ОДУ:

$$\begin{aligned} \dot{x_1} &= x_1 - x_1x_2 + cx_1^2 - ax_1^2x_3, \\ \dot{x_2} &= -x_2 + x_1x_2 \\ \dot{x_3} &= -bx_3 + ax_1^2x_3, \end{aligned} \tag{4.2}$$

где a,b,c — постоянные положительные параметры системы, а $x_i, i = 1,2,3$ — фазовые переменные.

Существование стационарного решения и асимптотические особенности решения рассматриваются, например, в [46]. Параметры *a, b, c* выбраны таким образом, чтобы плотности популяций в модели сходились к стационарному состоянию вымирания.

Прямая задача

Прямая задача, сформулированная для ОДУ, означает нахождение ее численного решения при наличии полной информации о системе (структура, параметры, начальные условия). Система (4.2) демонстрирует хаотическое поведение при следующем наборе значений параметров $a = 2{,}9851, b = 3, c = 2$. Начальные значения обобщенной хаотической системы Лотки-Вольтерра (4.2) принимаются $x_1(0) = 16, x_2(0) = 24,$ $x_3(0) = 18$.

Покажем способность представленного в Главе 2 метода аппроксимировать решение (4.2). Мы будем сравнивать численные решения, полученные с помощью предложенного метода и традиционных решателей *lsoda* на временном интервале *T=[0;2]*. Необходимо построить полиномиальное решение, содержащее не менее трех матриц $R^{1i}, i = \overline{1..3},$ так как правая часть (4.2) имеет максимальный порядок нелинейности, равный трем.

Согласно результатам численного моделирования, точность решения незначительно улучшается с увеличением порядка нелинейности для этой системы. Таким образом, теперь мы сосредоточимся на иллюстрации зависимости точности от шага дискретизации.

На рисунке 4.4 а) показаны траектории системы (4.2) $x_1\,(t), x_2\,(t), x_3\,(t)$, рассчитанные с помощью традиционного решателя *lsoda* и введенного матричного подхода для шага по времени 0,001 и третьего порядка нелинейности. Матрицы R^{1i}, $i = \overline{1..3}$, вычисленные в соответствии с предложенным алгоритмом, описываются формулами (4.3). Рисунок 4.4 б) иллюстрирует разницу между численными решениями, полученными с использованием двух рассмотренных методов.

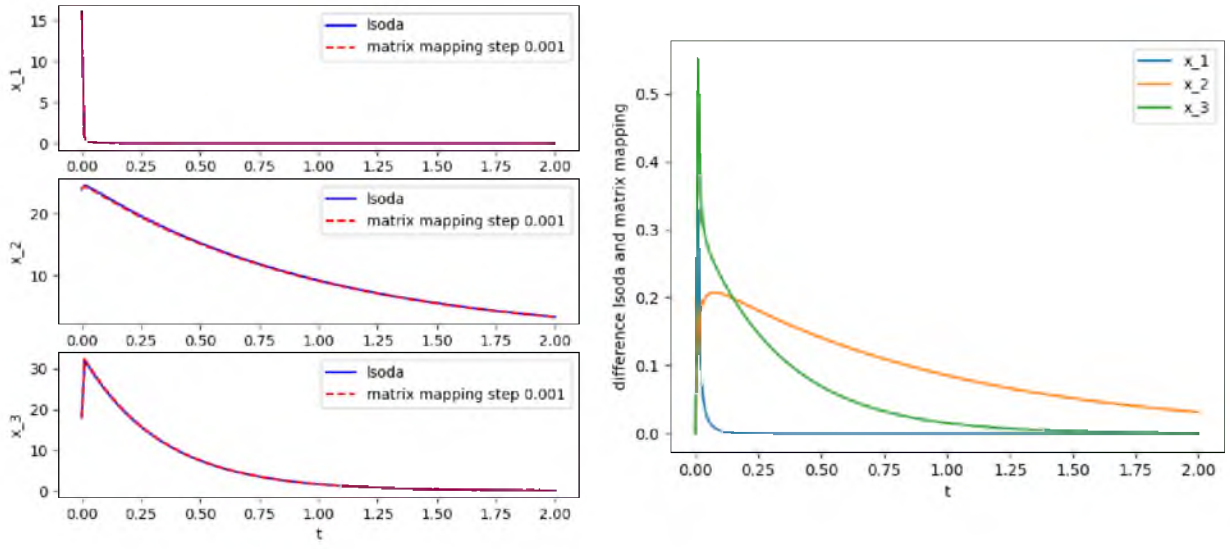

Рисунок 4.4. – а) Решение обобщенного уравнения Лотки-Вольтерра с помощью метода *lsoda* и метода матричных отображений б) Абсолютное отклонение численного решения с помощью метода *lsoda* и метода матричных отображений (шаг дискретизации 0,001 с)

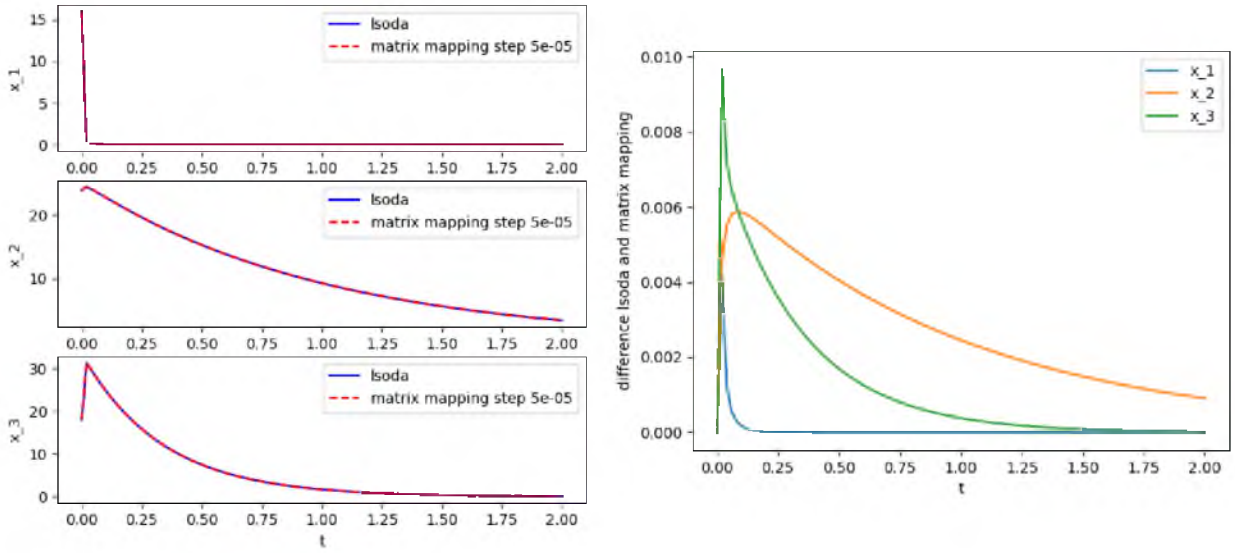

Рисунок 4.5. – а) Решение обобщенного уравнения Лотки-Вольтерра с помощью метода *lsoda* и метода матричных отображений б) Абсолютное отклонение численного решения с помощью метода *lsoda* и метода матричных отображений (шаг дискретизации $5 \cdot 10^{-5}$ с)

$$R^{11} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.999 & 0 \\ 0 & 0 & 0.997 \end{pmatrix}$$
$$R^{12} = \begin{pmatrix} 0.002 & -0.001 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (4.3)$$
$$R^{13} = \begin{pmatrix} 4 \cdot 10^{-6} & -3.5 \cdot 10^{-6} & -0.003 & 5 \cdot 10^{-7} & \dots & 0 \\ 0 & 1.5 \cdot 10^{-6} & 0 & -5 \cdot 10^{-7} & \dots & 0 \\ 0 & 0 & 0.003 & 0 & \dots & 0 \end{pmatrix}$$

Сравнивая графики абсолютных отклонений траекторий системы, построенных методом матричных отображений и решателем *lsoda,* можно сделать вывод, что отклонения принимают наибольшие значения при максимальных модулях производной решения. *Lsoda* — это адаптивный метод численного интегрирования, использующий начальный шаг интегрирования 10^{-5} с.

Проиллюстрируем сходимость решения матричного отображения, уменьшив шаг дискретизации. На рисунке 4.5 показаны траектории системы (4.2) $x_1(t), x_2(t), x_3(t)$ и абсолютное отклонение от численного решения *lsoda* для шага по времени $5 \cdot 10^{-5}$ с и третьего порядка нелинейности.

Сравнение графиков на рисунках 4.4 б) и 4.5 б) показывает, что максимальное расхождение между решениями, полученными традиционным и предложенным методом, уменьшается в 50 раз при уменьшении шага по времени в 20 раз. Это означает, что при необходимости возможно получить более точную аппроксимацию ОДУ с помощью полиномиальной нейронной сети, инициализированной матрицами (4.3), которые не зависят от начальных значений ОДУ и, таким образом, могут отражать динамику системы во всем диапазоне входных изменений.

Обратная задача

Проиллюстрируем способность полиномиальной нейронной сети решать обратную задачу для заданного ОДУ (4.2), заключающейся в восстановлении динамической модели по данным. Мы рассматриваем два случая:

- известна структура системы, но точные значения параметров a, b, c недоступны;
- информации о системе нет.

В качестве набора обучающих данных был взят синтетический временной ряд, сгенерированный, как численное решение (4.2) с параметрами $a = 10, b = 11, c = 5.$ и начальным условием $X_0 = [16, 10, 20]^T$. Дискретность по времени составляет 5 мс. Цель состоит в том, чтобы на основе этих данных обучить полиномиальную нейронную сеть для точного представления лежащего в основе динамического процесса. Затем обученная сеть может быть встроена как блок в более сложные нейросетевые архитектуры.

А. Обучение с предварительной информацией о системе

В первом случае мы рассматриваем обучение при неточный информации о значениях параметров в системе, однако структура (4.2) предполагается известной. Предположим, что значения параметров равны $a = 2{,}9851, b = 3, c = 2$. Тогда мы можем представить решение системы (4.2) с использованием преобразования Ли до 3-го порядка нелинейности. Матрицы $R^{1i}, i = 1..3$ находятся по формулам (4.3), уже найденным при решении прямой задачи, они же используются для инициализации весов полиномиальной

нейронной сети. В процессе обучения, весовые коэффициенты изменяются, чтобы соответствовать реальным значениям *a, b, c*.

Для проверки обученной полиномиальной сети обобщать прогнозы на новые начальные условия был сгенерирован набор тестовых данных путем численного решения системы (4.2) с различными начальными условиями ($X_0 = [6,2,4]^T$ и $X_0 = [10,6,8]^T$.

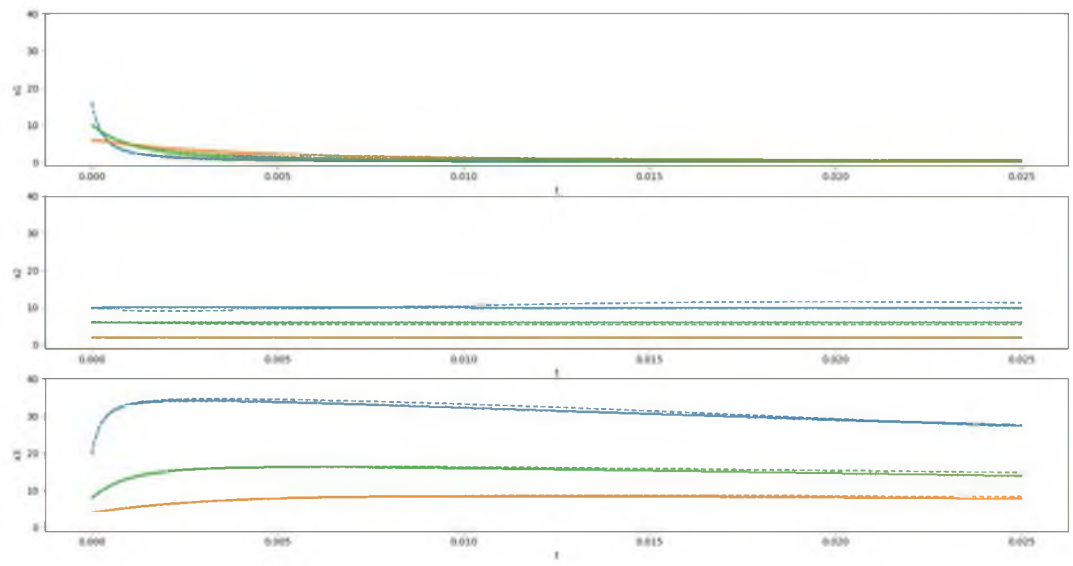

Рисунок 4.6 – Восстановленная динамика модели хищник-жертва (пунктирные линии) с помощью проинициализированной полиномиальной нейронной сети с одним выходом (синяя пунктирная линия — обучающие данные (500 обучающих пар, 100 эпох), зеленые и оранжевые пунктирные линии — тестовые данные)

На рисунке 4.6 показано, что даже когда наше первоначальное предположение о значениях параметров *a, b, c* довольно далеко от их истинных значений, полиномиальная нейронная сеть с одним выходом смогла адаптироваться на 500 обучающих парах точек за 100 эпох обучения.

Б. Обучение без использования предварительной информации о системе

Второй случай предполагает, что мы ничего не знаем об уравнениях системы, но все же хотим построить модель динамики на основе полиномиальной нейронной сети, используя те же обучающие данные, что и для случая А. На рисунке 4.7 показаны прогнозы, обученной полиномиальной нейронной сети для набора тренировочных и тестовых начальных значений. По сравнению с результатами обучения с использованием информации о структуре ОДУ, обучение без знаний о системе требует больше обучающих данных для сходимости прогнозируемой динамики к фактической.

Тем не менее из рисунка 4.7 видно, что при прогнозировании динамики второй популяции (x_2) наблюдаются отклонения прогнозных значений от истинных. Таким образом, остается пространство для дальнейшего улучшения алгоритмов обучения полиномиальной нейронной сети без инициализации. Одной из возможностей является

введение регуляризации, направленной на минимизацию количества ненулевых коэффициентов в весовых матрицах во время обучения (см. пункт 3.4.1).

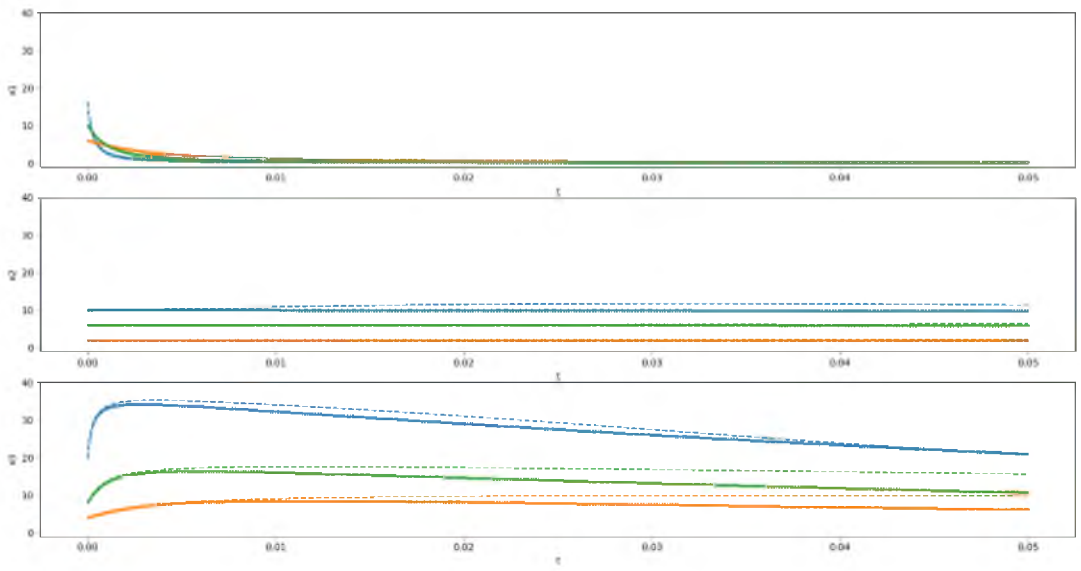

Рисунок 4.7 – Восстановленная динамика модели хищник-жертва (пунктирные линии) с помощью полиномиальной нейронной сети с одним выходом без инициализации (синяя пунктирная линия — обучающие данные (1000 обучающих пар, 100 эпох), зеленые и оранжевые пунктирные линии — тестовые данные)

Результаты сравнения

В качестве функции потерь при обучении использовалась стандартная среднеквадратическая ошибка, дополнительно оценивались средние абсолютная (MAE) и абсолютная процентная ошибки (MAPE). В таблице 4.1 приводятся их значения для случаев A и B во время обучения и проверки. На рисунке 4.8 сравниваются кривые обучения для полиномиальной нейронной сети при обучении по 500 и 1000 пар точек с и без предварительной инициализации.

Таблица 4.1 – Сравнение метрик на тренировочной и тестовой выборке

Experiment	**MSE**	**MAE**	**MAPE**
case A/500/Train	9.7814e-04	0.0166	2.2911
case A/500/Val 1	3.2742e-06	0.0012	0.0430
case A/500/Val 2	3.1238e-05	0.0031	0.1343
case A/1000/Train	4.1810e-04	0.0095	2.1466
case A/1000/Val 1	1.5368e-06	8.0159e-04	0.0268
case A/1000/Val 2	7.8331e-06	0.0014	0.1053
case B/500/Train	0.0013	0.0209	2.7056
case B/500/Val 1	5.8122e-06	0.0016	0.0319
case B/500/Val 2	4.4473e-05	0.0034	0.0918
case B/1000/Train	4.5774e-04	0.0097	2.0666

Как и ожидалось, обучение с использованием предварительной информации о системе в результате дает более низкие значения функции потерь и метрик точности, а также требует меньше данных для обучения. Процесс обучения с нулевых весов при тех же условиях сходится хуже, но, тем не менее, полиномиальная нейронная сеть способна воспроизводить реальную системную динамику даже для начальных условий, не представленных в обучающей выборке.

Рисунок 4.8. – Кривые обучения полиномиальной нейронной сети с одним выходом (с и без инициализации весовых коэффициентов) и использованием 500 и 1000 тренировочных пар точек

4.3 Прогнозирование трафика в сети

В данном разделе рассматривается алгоритм реконструкции динамической системы по временным рядам, а также пример применения этого алгоритма на наборе данных «Abilene» [26].

Рассматриваемый для примера набор данных «Abilene» описывает передачу трафика между 12 узлами телекоммуникационной сети, расположение которых представлено на рисунке 4.9.

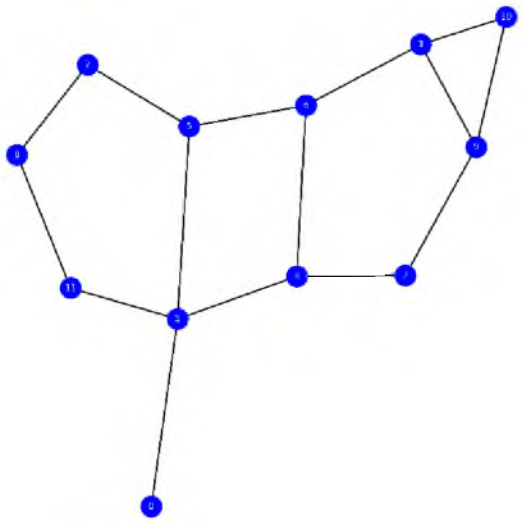

Рисунок 4.9 – Сеть Abilene

Каждое ребро означает передачу трафика между узлами, которые оно соединяет. Измерения величины трафика проводились для каждого ребра с 29.12.2003 00:00:00 по 13.06.2004 23:55:00 каждые 5 минут. Таким образом, этот набор данных – это таблица, состоящая из 30 столбцов и 48384 строк.

Декомпозиция данных

В качестве метода декомпозиции временного ряда был использован метод сингулярного спектрального анализа SSA. Этот метод основан на преобразовании одномерного временного ряда в многомерный ряд (по сути, представляющий собой матрицу, содержащую фрагменты временного ряда, полученные с некоторым сдвигом) с последующим сингулярным разложением полученной матрицы. В результате получается разложение из L слагаемых, которое пользователь должен самостоятельно сгруппировать, используя корреляционные матрицы, и получить основной сигнал и статистическую составляющую.

Например, рассмотрим временной ряд ребра (6, 3) из набора данных (см. рисунок 4.10).

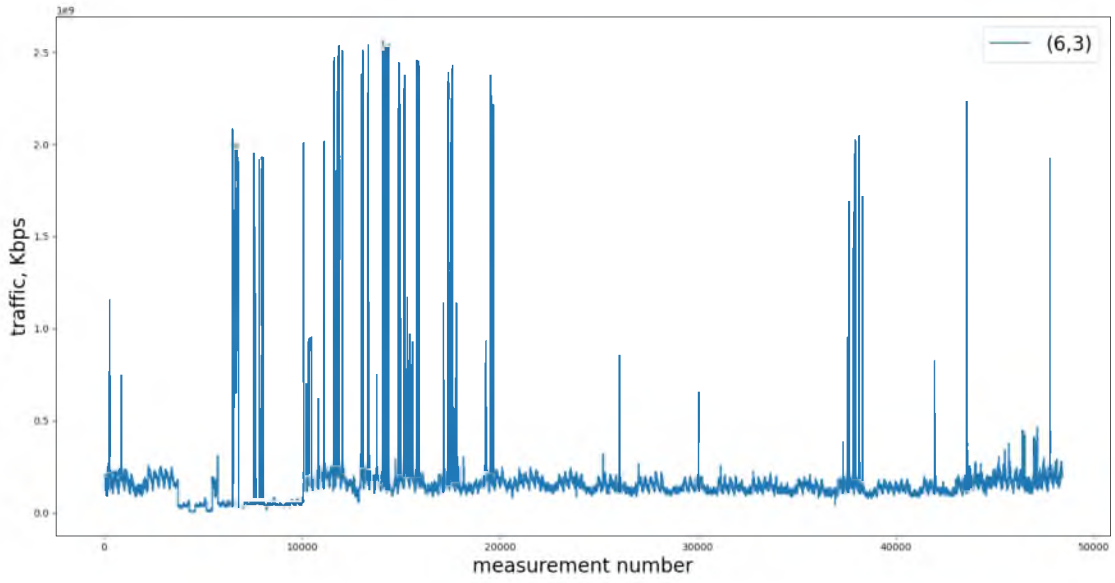

Рисунок 4.10 – Временной ряд трафика на ребре (6, 3)

Как можно заметить на графике, временной ряд имеет промежутки времени, характеризующиеся разной динамикой трафика, проходящего через ребро. Для иллюстрации описываемого метода рассмотрим измерения на таком временном промежутке, на котором эту динамику можно охарактеризовать как в той или иной степени однородную, например, измерения с 0 до 1000.

Выбирая параметр L для метода SSA, было выявлено, что чем он больше, тем более гладким и с меньшими колебаниями получается выделенный основной сигнал, и тем лучше

с ним работает алгоритм идентификации, однако тем меньше оказывается точность модели. Было решено выбрать L = 500. Используя корреляционную матрицу элементов разложения, было решено выделить сумму первых трех элементов как основной сигнал, а сумму всех остальных – как статистическую составляющую. Полученная в итоге декомпозиция представлена на рисунке 4.11.

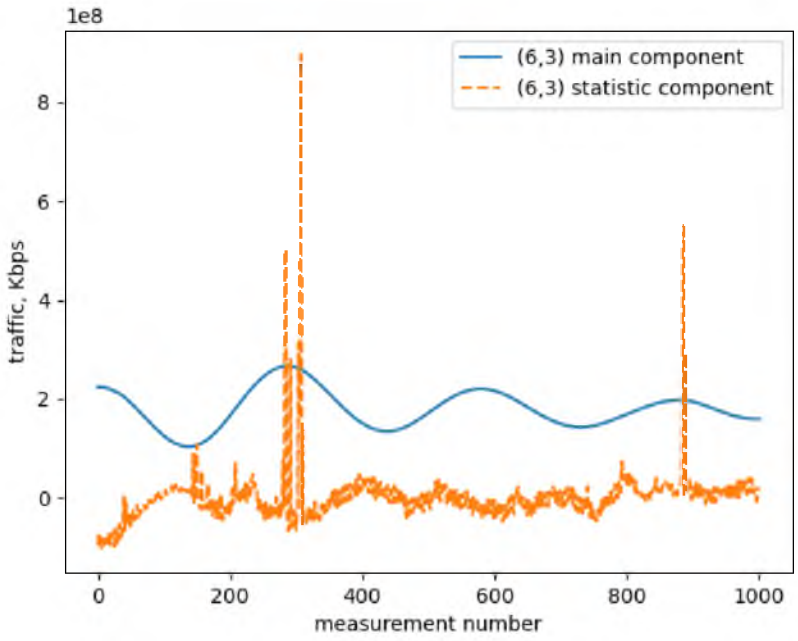

Рисунок 4.11 – Декомпозиция временного ряда ребра (6, 3)

Логично предположить, что на трафик, проходящий через одно ребро, влияет трафик, проходящий через соседние с ним ребра, причем с некоторой задержкой, связанной с тем, что трафик перемещается из одного узла в другой не моментально, а за какое-то время. Из этого предположения и появилась идея описать всю сеть динамической системой с запаздыванием.

Для того, чтобы понять, какие ребра влияют на какие в большей или меньшей степени, была построена корреляционная матрица для первых 1000 измерений временных рядов (см. рисунок 4.12).

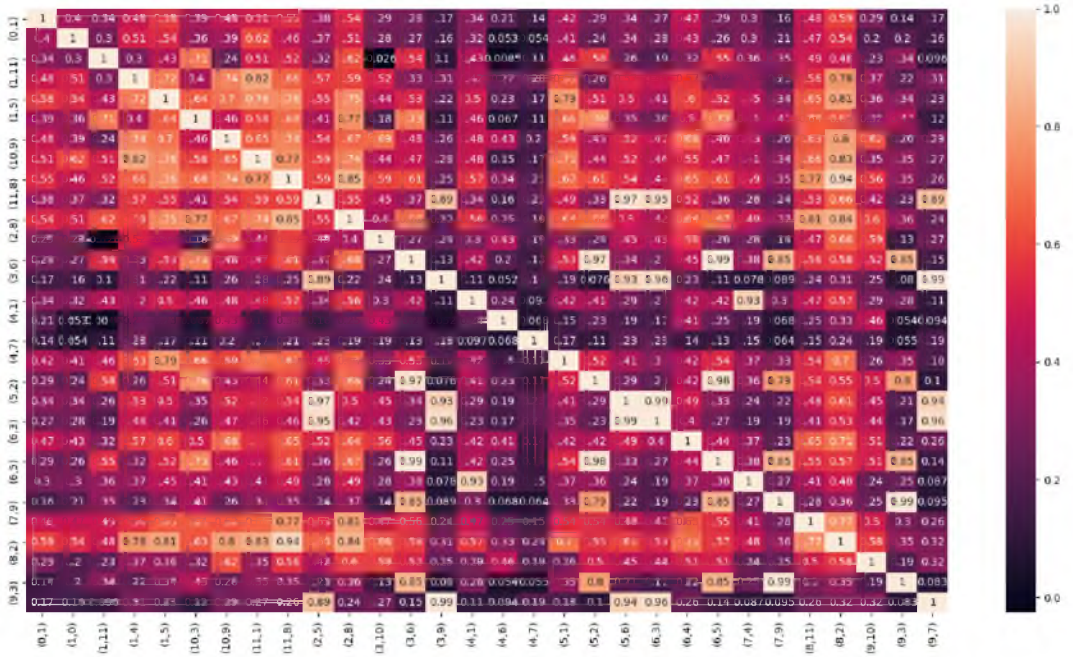

Рисунок 4.12 – Декомпозиция временного ряда ребра (6, 3)

Рассматривая коэффициенты корреляции для различных пар ребер, выяснилось, что, как и предполагалось, больше всего коррелируют между собой временные ряды соседних ребер. Таким образом, теперь можно описать всю сеть моделью, для начала связывая между собой динамической системой несколько последовательных ребер.

Для примера рассмотрим ребро (6, 3) и предшествующие ему последовательные ребра (2, 5) и (5, 6). Коэффициент корреляции между ребрами (2, 5) и (5, 6) равен 0,97, а между ребрами (5, 6) и (6, 3) – 0,99.

После применения декомпозиции к временным рядам этих ребер получаются основные сигналы и статистические составляющие, представленные на рисунке 4.13.

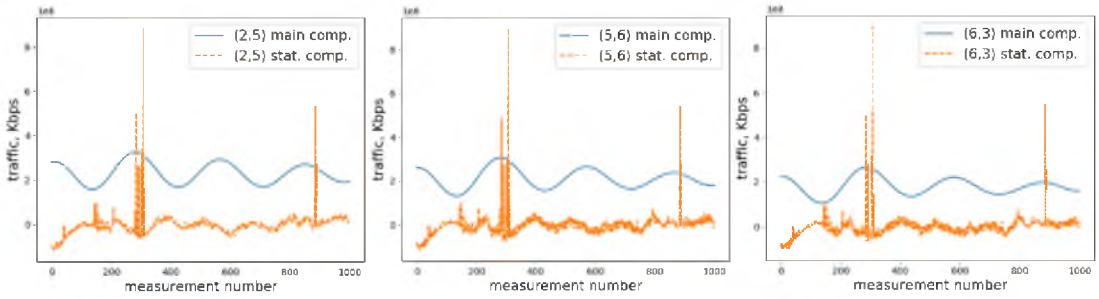

Рисунок 4.13 – Декомпозиция временных рядов ребер (2, 5), (5, 6), (6, 3)

Применяя алгоритм идентификации к временным рядам основных сигналов рассматриваемых ребер, получаем следующие функции, описывающие изменения основного сигнала трафика на этих ребрах (см. правый график на рисунке 4.14). В качестве тренировочных данных (показанных на левом графике на рисунке 4.14) на вход алгоритму

идентификации подавались первые 600 измерений из рассматриваемых 1000. Также на этом шаге с помощью перебора находятся величины задержек, с которыми трафик на ребре (6, 3) зависит от трафика на ребре (5, 6) и трафик на ребре (5, 6) зависит от трафика на ребре (2, 5), то есть такие величины задержек, при которых восстановленная модель получается наиболее точной. В качестве меры близости восстановленной системы и тренировочных данных использовалась величина MAE, которая для оптимальных величин задержек составила 0,238 Гбит (для тренировочных данных). После нахождения оптимальных параметров системы и оптимальных задержек можно прогнозировать поведение основных сигналов, например, в течение следующих после 600-го измерения 5 часов. На рисунке 4.14 вертикальная черта разграничивает тренировочные данные и спрогнозированные данные.

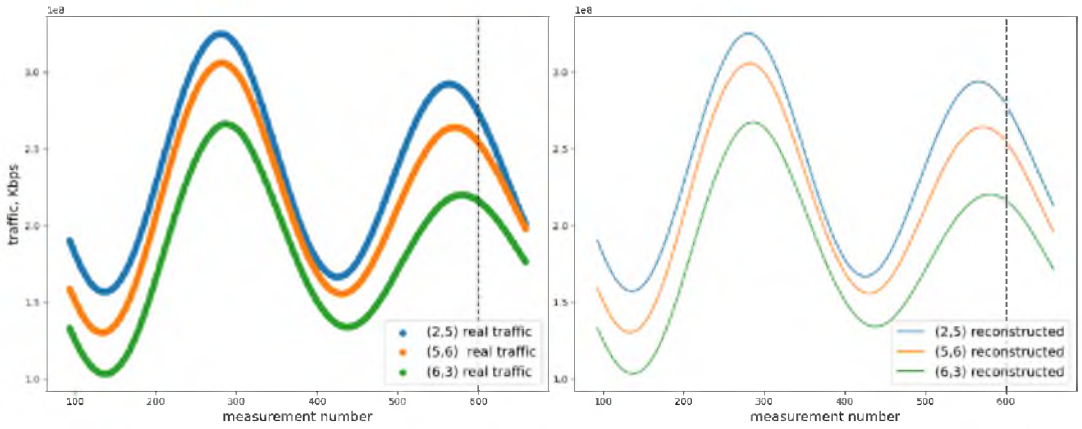

Рисунок 4.14 – Функции, описывающие изменения основного сигнала на рассматриваемых ребрах

Вид динамической системы, описывающей изменение трафика на ребрах (2, 5), (5, 6) и (6, 3), полученной в результате алгоритма идентификации, может быть представлен в виде $\boldsymbol{X'}^T = \boldsymbol{P}\boldsymbol{X}^T$, где $\boldsymbol{X'}, \boldsymbol{P}, \boldsymbol{X}$ – матрицы, представленные на рисунке 4.15. Функции $x(t), y(t), z(t)$ описывают изменение трафика на ребрах (2, 5), (5, 6) и (6, 3) соответственно. Величины a, b имеют значения 18 и 93 соответственно.

```
P = [[ 9.5e-23,  1.6e-13, -7.6e-13,  5.4e-13, -4.7e-13,  1.0e-08,  9.7e-09, -2.0e-09,  2.2e-09, -2.0e-08,
       4.3e-19, -7.2e-17,  6.0e-17, -6.1e-19, -1.1e-18,  1.4e-18,  2.0e-16, -1.1e-17, -1.9e-16,  1.0e-17],
     [ 1.5e-22,  2.4e-13, -1.2e-12,  8.3e-13,  1.9e-10,  1.4e-08,  1.6e-08, -3.8e-09,  4.0e-09, -3.1e-08,
      -5.8e-20,  1.5e-17, -6.8e-17,  6.2e-18, -8.7e-18, -3.5e-17, -9.6e-17, -5.8e-17,  1.5e-16,  9.2e-17],
     [ 1.0e-22,  1.7e-13, -8.4e-13,  6.0e-13,  1.5e-10,  1.0e-08,  1.1e-08, -3.7e-09,  4.2e-09, -2.2e-08,
       4.2e-19, -6.8e-17,  3.9e-17,  1.5e-18, -4.9e-18,  3.0e-18,  1.7e-16, -1.8e-17, -1.4e-16,  1.5e-17]]

X = [1, x(t-b), y(t-a), z, x(t-b)^2, y(t-a)^2, z^2, x(t-b)y(t-a),  x(t-b)z, y(t-a)z, x(t-b)^3, y(t-a)^3, z^3,
     (x(t-b)^2)y(t-a), (x(t-b)^2)z, x(t-b)(y(t-a)^2), z(y(t-a)^2), x(t-b)(z^2), y(t-a)(z^2), x(t-b)y(t-a)z]

X' = [x'(t), y'(t), z'(t)]
```

Рисунок 4.15 – Матрицы P, X, X'

В ходе исследования также выяснилось, что количество ребер в системе влияет на точность восстановления системы. Например, на рисунке 4.16 приведены результаты идентификации системы, состоящей из ребер (5, 6) и (6,3). MAE составляет 10,7 Гбит.

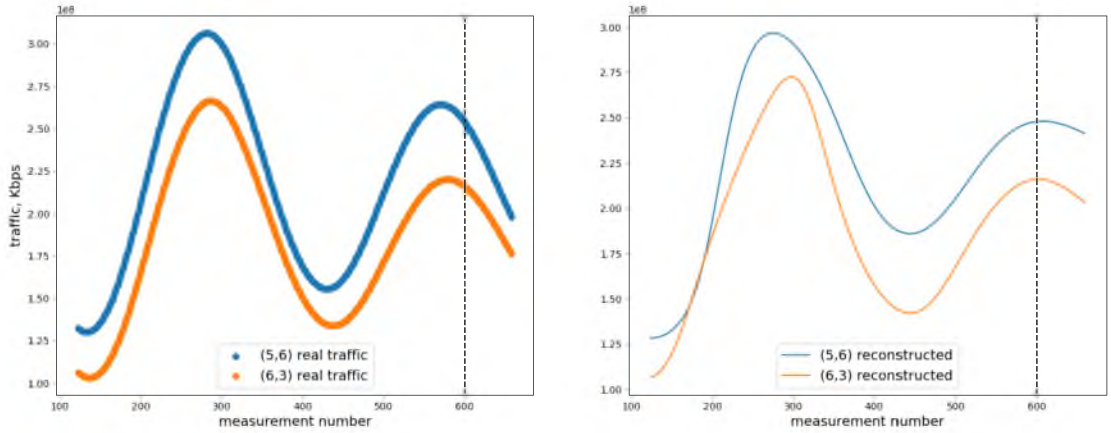

Рисунок 4.16 – Прогноз и истинное поведение трафика на рассматриваемых ребрах

В таблице 4.2 приведены величины погрешностей, полученные при идентификации системы, состоящей из разного количества ребер в системе (2 ребра (5, 6) и (6, 3) или 3 ребра (2, 5), (5, 6) и (6, 3)), при одинаковых или различных величинах запаздывания между ребрами в случае системы из трех ребер, и при различной длине временного промежутка, на котором проводится прогнозирование. Погрешности считались двумя способами: средняя абсолютная ошибка MAE и средняя абсолютная ошибка в процентах MAPE.

Таблица 4.2 – Сравнительная таблица погрешностей

Характеристики системы			**MAE (Гбит)**	**MAPE**
Количество ребер	**Величины запаздывания**	**Длительность прогноза (ч.)**		
2	-	2	10,558	0,054
3	Одинаковые	2	0,821	0,003
3	Разные	2	0,334	0,001
2	-	5	11,442	0,057
3	Одинаковые	5	1,532	0,006
3	Разные	5	0,628	0,002
2	-	10	15,160	0,074
3	Одинаковые	10	3,270	0,015
3	Разные	10	1,220	0,006

2	-	24	16,570	0,080
3	Одинаковые	24	4,629	0,022
3	Разные	24	2,807	0,014

Моделирование статистической компоненты

Для аппроксимации и прогнозирования статистических составляющих временных рядов была использована статистическая модель ARIMA.

Пусть, например, нужно спрогнозировать поведение статистической составляющей трафика на ребре (6, 3) в следующие 5 часов (принимая за текущий момент времени момент под номером 600). Тогда, используя вышеупомянутую модель, получаем следующий прогноз, выделенный зеленым цветом на рисунке 4.17.

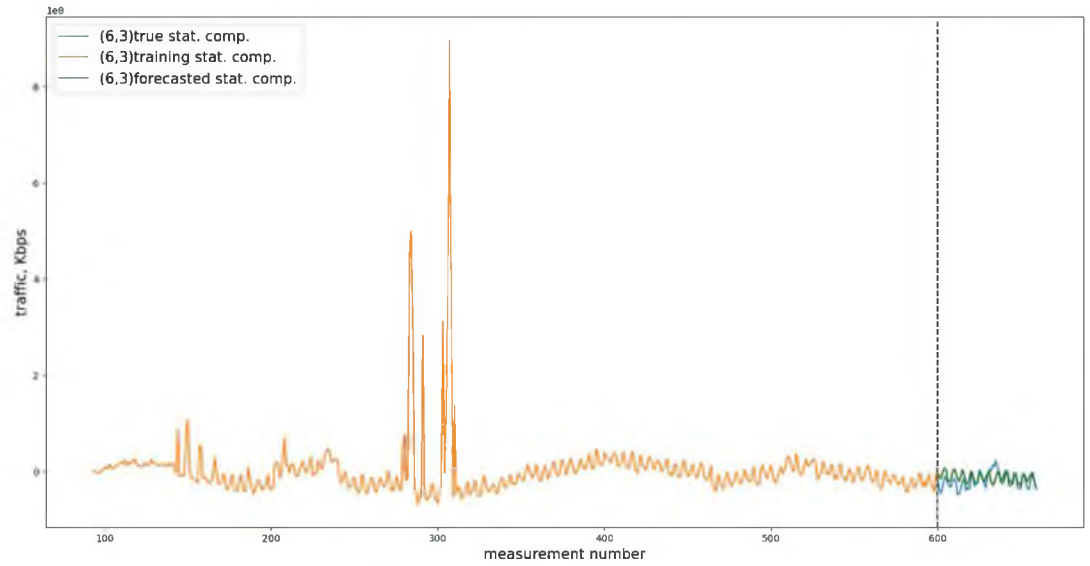

Рисунок 4.17 – Прогнозирование статистической составляющей на ребре (6, 3)

В таблице 4.3 приведены величины погрешностей, полученные при прогнозировании статистической составляющей на ребре (6, 3) при различной длине временного промежутка, на котором проводится прогнозирование.

Таблица 4.3 – Сравнительная таблица погрешностей статистической составляющей для ребра (6, 3)

Длительность прогноза (ч.)	**MAE (Гбит)**	**MAPE**
2	16,707	0,711
5	12,379	1,773
10	12,865	1,919

24	24,456	2,354

Суммирование основной и статистической компоненты. Прогнозирование

Продолжим рассматривать трафик на ребрах (2,5), (5, 6) и (6,3). Чтобы спрогнозировать его поведение, например, в следующие 5 часов, необходимо суммировать в каждый момент времени с номером от 0 до 660 значения функций, описывающих основные сигналы, и значения статистической составляющей, которые по отдельности представлены на рисунке 4.18.

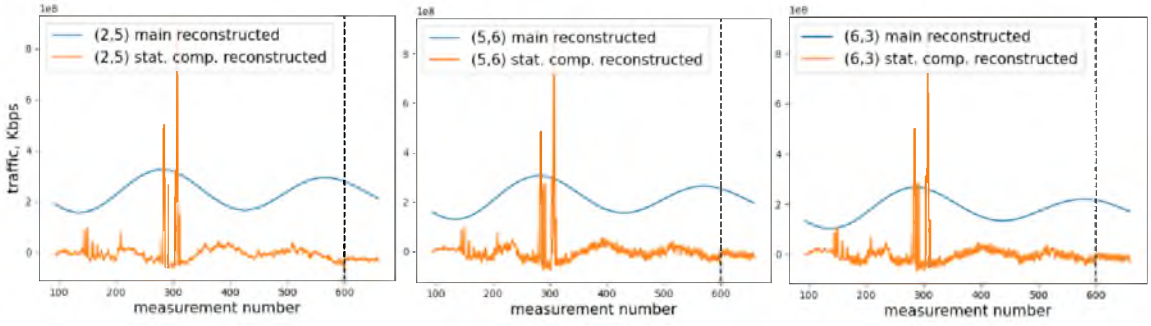

Рисунок 4.18 – Функции основных сигналов и статистических составляющих

Результат, полученный после сложения соответствующих значений представлен на рисунке 4.19. Также на рисунке 4.19 для сравнения приведены истинные значения трафика на всем временном промежутке. На рисунке 4.20 более детально представлен промежуток, на котором проводилось прогнозирование. MAE полученного прогноза на всем временном промежутке составляет 1.56 Гбит. MAPE составляет 0.00765.

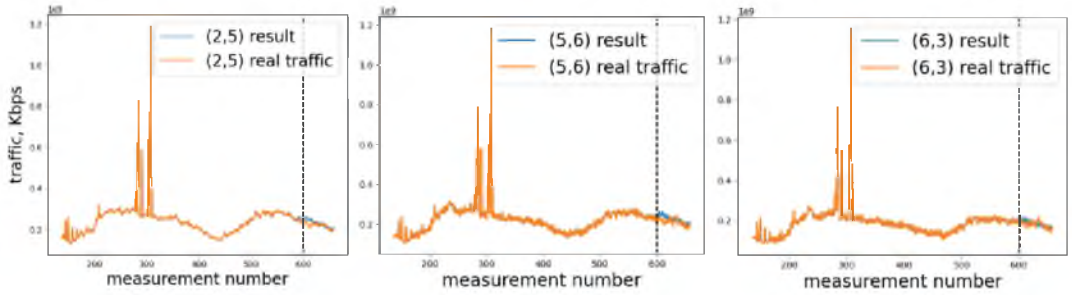

Рисунок 4.19 – Прогноз и истинное поведение трафика на рассматриваемых ребрах

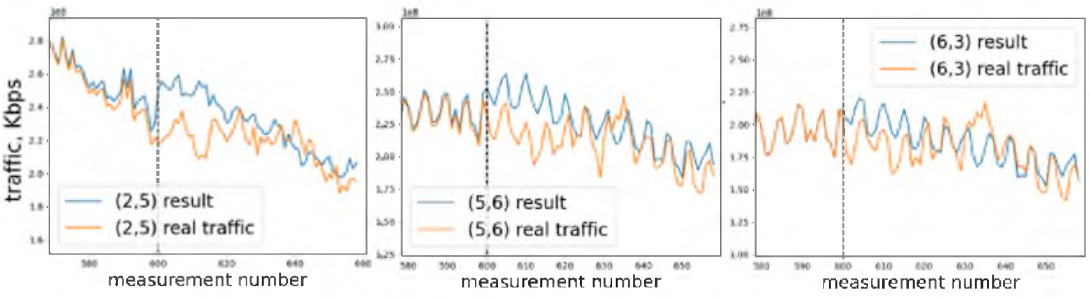

Рисунок 4.20 – Прогноз и истинное поведение трафика на рассматриваемых ребрах

В таблице 4.4 приведены величины погрешностей, полученные после суммирования значений основного сигнала и статистической составляющей при разном количестве ребер в системе, при одинаковых или различных величинах запаздывания между ребрами в случае системы из трех ребер, и при различной длине временного промежутка, на котором проводится прогнозирование.

Таблица 4.4 – Сравнительная таблица погрешностей результатов

Характеристики системы			**MAE (Гбит)**	**MAPE**
Кол-во ребер	**Величины запаздывания**	**Длительность прогноза (ч.)**		
2	-	2	11,354	0,062
3	Одинаковые	2	1,578	0,007
3	Разные	2	1,026	0,004
2	-	5	12,081	0,066
3	Одинаковые	5	2,084	0,010
3	Разные	5	1,569	0,007
2	-	10	14,346	0,081
3	Одинаковые	10	2,796	0,014
3	Разные	10	3,258	0,017
2	-	24	20,123	0,109
3	Одинаковые	24	9,682	0,042
3	Разные	24	9,975	0,043

Способ прогнозирования, описываемый в этом разделе, был сравнен со способом, основанном на обучении с использованием нейронной сети GRU (Gated Recurrent Units) [47]. В этом исследовании в качестве тренировочных данных используются данные временного ряда одного из ребер сети, соответствующие 6 часам измерений, а прогнозирование проводится на 1 час вперед. MAE при использовании этого способа составляет 7.4. Гбит.

Проведя прогнозирование на аналогичном количестве измерений для временного ряда ребра (6, 3), используя восстановленную по временным рядам ребер (2, 5), (5, 6) и (6, 3) динамическую систему, MAE описываемого в этой главе способа составила 0.9 Гбит.

Таким образом, пользуясь описанным способом, можно, группируя ребра с помощью корреляционной матрицы, восстанавливать динамическую систему, описывающую трафик на этих ребрах. Разделив все ребра системы на группы и связав каждую группу ребер динамической системой, можно получить систему, описывающую трафик во всей сети.

ЗАКЛЮЧЕНИЕ

В результате выполнения второго этапа НИР была разработана архитектура нейронной сети на основе нелинейного матричного преобразования и исследована ее применимость для идентификации и построения нестационарных моделей динамических систем.

Описаны теоретические основы математического аппарата на основе преобразования Ли и его связь с обыкновенными дифференциальными уравнениями. Предложен алгоритм построения и обучения рекуррентной полиномиальной нейронной сети, соответствующей системе неавтономных дифференциальных уравнений. Алгоритм обучения состоит из двух этапов: инициализация весовых коэффициентов нейронной сети и дообучение на данных. В случае, когда доступно малое количество данных, дообучение проводится с использованием представленных регуляризационных методов, позволяющих контролировать сложность нейросетевой модели и избегать переобучения коэффициентов. Кроме того, разработан алгоритм идентификации неавтономных дифференциальных уравнений на основе измерений, работающий в том числе при нерегулярно распределенных обучающих данных. Таким образом, в рамках первого этапа научно-исследовательской работы помимо создания новых нейросетевых архитектур развиваются подходы, снижающие требования к объему и качеству данных, необходимых для обучения модели.

Предложенные алгоритмы продемонстрированы на нескольких примерах, которые часто используются для тестирования новых методов моделирования. При проведении численных экспериментов в том числе осуществлялась оценка точности и качества полученных моделей.

Таким образом, в результате выполнения второго этапа НИР все поставленные задачи были выполнены в полном объеме:

1. Усовершенствована реализация разработанных численных алгоритмов в рамках первого этапа НИР для случая произвольной размерности вектора фазовых переменных.
2. Разработана архитектура полиномиальной нейронной сети на основе нелинейного матричного преобразования оператора Ли для случая произвольных нелинейных неавтономных систем с полиномиальными нелинейностями.
3. Разработан алгоритм инициализации весовых коэффициентов нейронной сети (п. 2) с использованием результатов, полученных в рамках первого этапа НИР
4. Определены требования к количеству, дискретности данных и порядку нелинейности для обеспечения требуемой точности прогнозирования с использованием полиномиальной нейронной сети.

5. Разработан алгоритм реконструкции дифференциального уравнения по данным временных рядов (в случае, если вид уравнения заранее неизвестен, а в данных присутствует шум и различные статистические возмущения).
6. Построена оценка точности нейросетевой аппроксимации
7. Разработан алгоритм обучения построенной нейронной сети на малых данных
8. Разработанные алгоритмы продемонстрированы на классических модельных задачах из теории нелинейных систем: нелинейный дефлектор, система Лотки-Вольтерры, а также на реальных данных.
9. Описанные алгоритмы реализованы в виде библиотеки с открытым исходным кодом с использованием набора библиотек TensorFlow.

Значимость полученных результатов обуславливается возможностью их применения для решения задач моделирования, прогнозной аналитики и оптимального управления промышленными объектами, для которых влияние нелинейных эффектов является существенным. В отличие от существующих моделей искусственного интеллекта и машинного обучения, которые не гарантируют математической строгости предсказания, имеют нерешенные проблемы с устойчивостью и интерпретируемостью, развиваемые в рамках научно-исследовательской работы новые подходы позволят преодолеть эти барьеры. Это позволит построить интеллектуальные системы управления индустриальными процессами, где критически важным является соблюдение физических законов, технологических ограничений, и возможность адаптации к изменяющимся условиям внешней среды.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Kumar S., Tiwari P., Zymbler M. Internet of Things is a revolutionary approach for future technology enhancement: a review // J Big Data. 2019. Vol. 6, № 1. P. 111.
2. Rehman H.U., Asif M., Ahmad M. Future applications and research challenges of IOT // 2017 International Conference on Information and Communication Technologies (ICICT). Karachi: IEEE, 2017. P. 68–74.
3. Lee S., Bae M., Kim H. Future of IoT Networks: A Survey // Applied Sciences. 2017. Vol. 7, № 10. P. 1072.
4. Mathur S. et al. AIIOT: Emerging IoT with AI Technologies // A Fusion of Artificial Intelligence and Internet of Things for Emerging Cyber Systems / ed. Kumar P. et al. Cham: Springer International Publishing, 2022. Vol. 210. P. 269–291.
5. Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead // Nat Mach Intell. 2019. Vol. 1, № 5. P. 206–215.
6. Chen R.T.Q. et al. Neural Ordinary Differential Equations // Advances in Neural Information Processing Systems. Curran Associates, Inc., 2018. Vol. 31.
7. Greydanus S., Dzamba M., Yosinski J. Hamiltonian Neural Networks // Advances in Neural Information Processing Systems. Curran Associates, Inc., 2019. Vol. 32.
8. Han C.-D. et al. Adaptable Hamiltonian neural networks // Phys. Rev. Research. 2021. Vol. 3, № 2. P. 023156.
9. Hamiltonian-based Neural ODE Networks on the SE(3) Manifold For Dynamics Learning and Control [Electronic resource]. URL: https://thaipduong.github.io/SE3HamDL/ (accessed: 12.09.2022).
10. Lutter M., Ritter C., Peters J. Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning. 2018.
11. Roehrl M.A. et al. Modeling System Dynamics with Physics-Informed Neural Networks Based on Lagrangian Mechanics**This work was sponsored by the German Federal Ministry of Education and Research (ID: 01 IS 18049 A). // IFAC-PapersOnLine. 2020. Vol. 53, № 2. P. 9195–9200.
12. [PDF] Taylor-Lagrange Neural Ordinary Differential Equations: Toward Fast Training and Evaluation of Neural ODEs | Semantic Scholar [Electronic resource]. URL: https://www.semanticscholar.org/reader/0c3170c199f4a6173f6b07302ffac559e4924586 (accessed: 10.09.2022).
13. Goyal P., Benner P. Learning Dynamics from Noisy Measurements using Deep Learning with a Runge-Kutta Constraint. 2021.

14. Wang Y.-J., Lin C.-T. Runge-Kutta neural network for identification of dynamical systems in high accuracy // IEEE Transactions on Neural Networks. 1998. Vol. 9, № 2. P. 294–307.
15. Kim B. et al. Robust Neural Networks Inspired by Strong Stability Preserving Runge-Kutta Methods // Computer Vision – ECCV 2020 / ed. Vedaldi A. et al. Cham: Springer International Publishing, 2020. Vol. 12354. P. 416–432.
16. Cuomo S. et al. Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next: arXiv:2201.05624. arXiv, 2022.
17. Chiu P.-H. et al. CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method // Computer Methods in Applied Mechanics and Engineering. 2022. Vol. 395. P. 114909.
18. Djeumou F. et al. Neural Networks with Physics-Informed Architectures and Constraints for Dynamical Systems Modeling // undefined. 2022.
19. Karniadakis G.E. et al. Physics-informed machine learning: 6 // Nat Rev Phys. Nature Publishing Group, 2021. Vol. 3, № 6. P. 422–440.
20. Physics-enhanced Neural Networks in the Small Data Regime [Electronic resource] // DeepAI. 2021. URL: https://deepai.org/publication/physics-enhanced-neural-networks-in-the-small-data-regime (accessed: 12.09.2022).
21. Ганаева Д., Головкина А. Метод реконструкции нелинейных динамических систем по временным рядам // Процессы управления и устойчивость. Vol. 9, № 1. P. 197–201.
22. Golovkina A., Kozynchenko V., Kulabukhova N. RECONSTRUCTION OF ORDINARY DIFFERENTIAL EQUATIONS FROM IRREGULARLY DISTRIBUTED TIME- SERIES DATA // 9th International Conference "Distributed Computing and Grid Technologies in Science and Education." Crossref, 2021. P. 342–347.
23. Golovkina A., Kozynchenko V. Parametric Identification of a Dynamical System with Switching // Computational Science and Its Applications – ICCSA 2022 Workshops / ed. Gervasi O. et al. Cham: Springer International Publishing, 2022. P. 557–569.
24. Golovkina A., Kozynchenko V. Neural Network Representation for Ordinary Differential Equations // Artificial Intelligence in Models, Methods and Applications. Springer International Publishing, 2022. Vol. 457.
25. PNN-Lab/tmflow: Python. PNN-Lab, 2022.
26. Набор данных «Abilene».
27. Ljung L. System identification: theory for the user. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999. 609 p.

28. Brunton S.L., Proctor J.L., Kutz J.N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems // Proc. Natl. Acad. Sci. U.S.A. 2016. Vol. 113, № 15. P. 3932–3937.
29. Lguensat R. et al. The Analog Data Assimilation // Mon. Wea. Rev. 2017. Vol. 145, № 10. P. 4093–4107.
30. Pathak J. et al. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data // Chaos. 2017. Vol. 27, № 12. P. 121102.
31. Pathak J. et al. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach // Phys. Rev. Lett. 2018. Vol. 120, № 2. P. 024102.
32. Ayed I. et al. Learning Dynamical Systems from Partial Observations: arXiv:1902.11136. arXiv, 2019.
33. Фихтенгольц Г.М. Курс дифференциального и интегрального исчисления. 8th ed. Москва: Физматлит, 2003. Vol. 1. 276 p.
34. Traore O.I. et al. Structure analysis and denoising using Singular Spectrum Analysis: Application to acoustic emission signals from nuclear safety experiments // Measurement. 2017. Vol. 104. P. 78–88.
35. Box G.E.P., Jenkins G.M. Time series analysis: forecasting and control. Rev. ed. San Francisco: Holden-Day, 1976. 575 p.
36. Findley D.F. et al. New Capabilities and Methods of the X-12-ARIMA Seasonal-Adjustment Program // Journal of Business & Economic Statistics. 1998. Vol. 16, № 2. P. 127.
37. Голяндина Н.Э. Метод “Гусеница”-SSA: анализ временных рядов. Санкт-Петербург: СПбГУ, 2004. 76 p.
38. Polynomial and spline approximation: theory and applications: proceedings of the NATO Advanced Study Institute held at Calgary, Canada, August 26- September 2, 1978 / ed. Sahney B.N. Dordrecht ; Boston: D. Reidel Pub. Co, 1979. 321 p.
39. Abadi M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015.
40. Chollet F. Deep Learning with Python. New York: Manning Publications Co, 2018.
41. Louizos C., Welling M., Kingma D.P. Learning Sparse Neural Networks through L_0 Regularization. 2022.
42. Louizos C., Welling M., Kingma D.P. Learning Sparse Neural Networks through L_0 Regularization: arXiv:1712.01312. arXiv, 2018.
43. Liu J. et al. Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers. 2020.

44. Duchi J., Hazan E., Singer Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. P. 39.

45. Vaidyanathan S. ADAPTIVE CONTROL AND SYNCHRONIZATION OF A GENERALIZED LOTKA-VOLTERRA SYSTEM. 2011. P. 12.

46. Elsadany A.A. et al. Dynamical analysis, linear feedback control and synchronization of a generalized Lotka-Volterra system // International Journal of Dynamics and Control. 2018. Vol. 6, № 1. P. 328–338.

47. Troia S. et al. Deep Learning-Based Traffic Prediction for Network Optimization // 2018 20th International Conference on Transparent Optical Networks (ICTON). Bucharest: IEEE, 2018. P. 1–4.