



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(национальный исследовательский университет)»

---

Институт (Филиал) № 8 «Компьютерные науки и прикладная математика» Кафедра 806  
Группа М8О-407Б-19 Направление подготовки 01.03.02 «Прикладная математика и  
информатика»  
Профиль Информатика  
Квалификация: бакалавр

---

## ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему: Разреженная идентификация нелинейных динамических систем

Автор ВКРБ:	Бирюков Виктор Владимирович	(_____)
Руководитель:	Ревизников Дмитрий Леонидович	(_____)
Консультант:	—	(_____)
Консультант:	—	(_____)
Рецензент:	—	(_____)

### К защите допустить

Заведующий кафедрой № 806                    Крылов Сергей Сергеевич                    (\_\_\_\_\_)  
\_\_\_\_\_ мая 2023 года

## **РЕФЕРАТ**

Выпускная квалификационная работа бакалавра состоит из 50 страниц, 35 рисунков, 24 использованных источников, 1 приложения.

**ДИНАМИЧЕСКИЕ СИСТЕМЫ, ИДЕНТИФИКАЦИЯ СИСТЕМ,  
РАЗРЕЖЕННАЯ РЕГРЕССИЯ, ЧИСЛЕННОЕ ДИФФЕРЕНЦИРОВАНИЕ,  
ОБЫКНОВЕННЫЕ ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ**

Объектом исследования в данной работе являются нелинейные динамические системы.

Цель работы — идентификация систем обыкновенных дифференциальных уравнений первого порядка на основе потенциально шумных данных.

Для достижения поставленной цели были проведены исследования в области алгоритмов идентификации, а также разреженной регрессии и численного дифференцирования.

Основное содержание работы состояло в разработке алгоритма идентификации и его составных частей: алгоритмов разреженной регрессии и устойчивого численного дифференцирования.

Основными результатами работы, полученными в процессе разработки, являются алгоритм идентификации, рекомендации по его использованию, а также рекомендации по использованию методов дифференцирования, которые для данной задачи используются впервые.

Результаты разработки предназначены для извлечения математических закономерностей из данных — структурного анализа.

Использование результатов данной работы позволяет существенно расширить возможности анализа данных, так как появляется возможность использовать соответствующий математический аппарат для анализа самой системы.

## СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ . . . . .	4
ВВЕДЕНИЕ . . . . .	5
1 ПОСТАНОВКА ЗАДАЧИ . . . . .	7
1.1 Актуальность работы . . . . .	7
1.2 Техническое задание . . . . .	8
2 РАЗРЕЖЕННАЯ ИДЕНТИФИКАЦИЯ НЕЛИНЕЙНЫХ ДИНАМИЧЕСКИХ СИСТЕМ . . . . .	9
2.1 Теоретическое описание . . . . .	9
2.1.1 Структурный анализ нелинейных динамических систем . .	9
2.1.2 Разреженная регрессия . . . . .	11
2.1.3 Численное дифференцирование . . . . .	14
2.2 Используемые технологии . . . . .	21
2.3 Описание программной разработки . . . . .	21
3 РЕЗУЛЬТАТЫ РАБОТЫ . . . . .	23
3.1 Результаты . . . . .	23
3.1.1 Анализ методов разреженной регрессии . . . . .	23
3.1.2 Анализ методов численного дифференцирования . . . .	25
3.1.3 Анализ алгоритма идентификации . . . . .	36
3.2 Перспективы дальнейшего развития алгоритма . . . . .	45
ЗАКЛЮЧЕНИЕ . . . . .	47
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ . . . . .	48
ПРИЛОЖЕНИЕ А Исходный код . . . . .	50

## **ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

В настоящей выпускной квалификационной работе бакалавра применяют следующие определения, обозначения и сокращения:

IoU — intersection over union

Lasso — least absolute shrinkage and selection operator

SINDy — sparse identification of nonlinear dynamics

STLSQ — sequential thresholded least-squares

TVR — total variation regularization

МНК — метод наименьших квадратов

ОДУ — обыкновенное дифференциальное уравнение

СЛАУ — система линейных алгебраических уравнений

Уровень шума — дисперсия белого гауссова шума с нулевым математическим ожиданием

## ВВЕДЕНИЕ

Актуальность темы данной работы связана с распространенностью нелинейных динамических систем для описания различных процессов. В случаях, когда существуют только дискретные данные, собранные, например, в результате эксперимента, возникает задача по идентификации системы, лежащей за этими данными. Такая задача может осложняться большим объемом данных, их сложностью или наличием в них посторонних шумов, что делает проблематичным ее решение в ручном режиме.

Таким образом, выполненная работа актуальна и с теоретической, и с практической точек зрения.

Цель работы — идентификация систем обыкновенных дифференциальных уравнений первого порядка на основе потенциально шумных данных.

Для достижения поставленной цели в работе были решены следующие задачи:

- реализация алгоритма идентификации;
- реализация алгоритмов дифференцирования шумных данных;
- реализация алгоритмов разреженной регрессии;
- тестирование алгоритма идентификации на известных системах ОДУ 1-ого порядка;
- сравнение различных методов дифференцирования и регрессии.

Работа основывалась на следующих инструментах и методах:

- язык программирования Python,
- библиотеки научных вычислений NumPy и SciPy,
- библиотека машинного обучения Scikit-learn,
- библиотеки построения графиков Matplotlib и Seaborn,
- среда разработки Jupyter.

Основными результатами, полученными в работе, являются:

- работающий алгоритм идентификации систем ОДУ по шумным данным;
- рекомендации по использованию алгоритма идентификации;
- реализация алгоритмов численного дифференцирования методом регуляризации полной вариации;
- рекомендации по подбору параметров и использованию алгоритмов

дифференцирования.

Результаты работы предназначены для использования в анализе данных.

Использование разработки позволяет выявлять в данных закономерности, которые могут быть описаны при помощи нелинейных динамических систем различного вида, например, систем ОДУ 1-ого порядка. Это существенно расширяет возможности анализа данных, так как позволяет использовать соответствующий математический аппарат для анализа самой системы.

# 1 ПОСТАНОВКА ЗАДАЧИ

## 1.1 Актуальность работы

Извлечение математических закономерностей из данных — структурный анализ — является важной задачей во многих научных областях. Существует много проблем, связанных с данными, таких как понимание когнитивных процессов на основе мозговой активности, изучение закономерностей климата, определение устойчивости финансовых рынков, прогнозирование и подавление распространения болезней. В связи с обилием данных, роль структурного анализа в этих сферах вероятно будет только расти.

Достижения в области машинного обучения и науке о данных привели к продвижению в анализе и понимании сложных данных, извлечение закономерностей из которых превышает возможности человека. Однако, несмотря на быстрое развитие инструментов на основе статистических отношений, замедлился прогресс в извлечении физических моделей динамических процессов. Это приводит к тому, что имеющиеся методы не могут экстраполировать результат за пределы аттрактора, в котором данные были получены.

Поэтому более перспективными кажутся методы, извлекающие закономерности в символьном виде. Первым прорывом в этой области стало использование символьной регрессии и генетического программирования для поиска нелинейных дифференциальных уравнений [1; 2]. В таком подходе балансируется сложность модели, измеряемая количеством членов, с точностью идентификации. Однако символьная регрессия вычислительно сложна, а также плохо масштабируется и склонна к переобучению.

Другим методом, который и рассматривается в данной работе, является решение задачи идентификации с точки зрения разреженной регрессии [3; 4]. При этом активно используется тот факт, что большинство физических систем имеют небольшое количество значимых членов, что делает их разреженными в пространстве нелинейных функций. Это позволяет избежать комбинаторного взрыва, который возникает при обычном переборе. В результате процесс идентификации обеспечивает естественное балансирование сложности модели, которая определяется разреженностью

правых частей уравнений, с точностью. Использование алгоритмов выпуклой оптимизации позволяет применять метод к задачам большого масштаба.

В таком подходе также можно усмотреть сходство с методом разложения по динамическим модам (dynamic mode decomposition) [5], который является линейной динамической регрессией. Этот метод опирается только на исходные данные, а не на знание уравнений динамики, и также связан с теорией операторов Купмана [6]. Однако при использовании этого метода все равно необходимы предположения о виде динамической системы, поскольку на данный момент нет методов по определению наблюдаемых функций. В отличие от этого, использование разреженной регрессии позволяет автоматически определять значимые члены в динамических системах.

## 1.2 Техническое задание

Таким образом, целью работы является реализация алгоритма идентификации нелинейных систем на основе данных.

Входные данные представляют из себя массив замеров некоторых величин, описывающих динамическую систему. Так как алгоритм нацелен на использование с реальными системами и данными, полученными экспериментальным путем, входные данные могут содержать некоторую шумовую компоненту. Поэтому разработанный алгоритм должен быть устойчив к этому шуму.

Задача идентификация ограничивается системами обыкновенных дифференциальных уравнений первого порядка. Системы ОДУ 1-ого порядка позволяют описать довольно много процессов реального мира и для их идентификации достаточно базовой версии алгоритма и производных первого порядка.

В качестве источника данных используются известные системы ОДУ. Данные получаются синтетическим путем.

## 2 РАЗРЕЖЕННАЯ ИДЕНТИФИКАЦИЯ НЕЛИНЕЙНЫХ ДИНАМИЧЕСКИХ СИСТЕМ

### 2.1 Теоретическое описание

#### 2.1.1 Структурный анализ нелинейных динамических систем

Алгоритм, описываемый в данной работе, позволяет производить структурную идентификацию динамических систем [3] вида:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)), \quad (1)$$

где вектор  $\mathbf{x}(t) = (x_1(t) \quad x_2(t) \quad \cdots \quad x_n(t))^T \in \mathbb{R}^n$  описывает состояние системы в момент времени  $t$ ;

$f(\mathbf{x}(t))$  — нелинейная функция, описывающая поведение самой системы.

Ключевое наблюдение, лежащее в основе алгоритма, состоит в том, что большое число таких функций представляют из себя линейную комбинацию небольшого количества значимых членов, другими словами, они являются разреженными в пространстве возможных функций.

Для проведения структурной идентификации необходимо собрать множество измерений состояний системы  $\mathbf{x}(t)$  и соответствующие им значения производной. Эти измерения образуют две матрицы:

$$X = \begin{pmatrix} \mathbf{x}^T(t_1) \\ \mathbf{x}^T(t_2) \\ \vdots \\ \mathbf{x}^T(t_m) \end{pmatrix} = \begin{pmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{pmatrix} \quad (2)$$

$$\dot{X} = \begin{pmatrix} \dot{\mathbf{x}}^T(t_1) \\ \dot{\mathbf{x}}^T(t_2) \\ \vdots \\ \dot{\mathbf{x}}^T(t_m) \end{pmatrix} = \begin{pmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \dot{x}_n(t_m) \end{pmatrix}. \quad (3)$$

В случае использования реальных экспериментальных данных (или синтезированных данных, которые призваны их заменить) значения производных будут неизвестны. В связи с этим возникает задача численного дифференцирования для получения данных значений. Такая задача может

быть осложнена наличием шума в данных, что требует специальных алгоритмов дифференцирования, способных справиться с подобными затруднениями.

Следующим этапом составляется матрица признаков  $\Theta(X)$ , которая содержит нелинейные функции от столбцов  $X$ , предположительно содержащиеся в искомой системе. Например, она может состоять из константной функции, полиномов и тригонометрических функций:

$$\Theta(X) = \begin{pmatrix} | & | & | & | & | & | \\ 1 & X & X^{P_2} & X^{P_3} & \dots & \sin(X) & \cos(X) & \sin(2X) & \cos(2X) & \dots \\ | & | & | & | & | & | \end{pmatrix}, \quad (4)$$

где  $X^{P_2}$ ,  $X^{P_3}$  обозначают полиномы второй и третьей степени соответственно, например:

$$X^{P_2} = \begin{pmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \dots & x_2^2(t_1) & x_2(t_1)x_3(t_1) & \dots & x_n^2(t_1) \\ x_1^2(t_2) & x_1(t_2)x_2(t_2) & \dots & x_2^2(t_2) & x_2(t_2)x_3(t_2) & \dots & x_n^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_1^2(t_m) & x_1(t_m)x_2(t_m) & \dots & x_2^2(t_m) & x_2(t_m)x_3(t_m) & \dots & x_n^2(t_m) \end{pmatrix}. \quad (5)$$

Теперь задача заключается в определении того, как именно надо скомбинировать столбцы матрицы признаков для получения векторов левых частей. Это есть ни что иное, как задача линейной регрессии, при этом, так как было сделано предположение, что искомая функция содержит небольшое число значимых членов, это конкретный подвид линейной регрессии — разреженная регрессия — при помощи которого можно получить разреженную матрицу коэффициентов  $W$ :

$$\dot{X} = \Theta(X)W. \quad (6)$$

Столбцы матрицы  $W$  определяют значимые члены правых частей каждого из уравнений системы и коэффициенты перед ними.

Таким образом, основными частями алгоритма являются численное дифференцирование и разреженная регрессия. Рассмотрим каждую из них подробно.

## 2.1.2 Разреженная регрессия

Задача линейной регрессии формулируется следующим образом. Имеется множество объектов и множество соответствующих им значений целевой переменной. Необходимо подобрать такие коэффициенты линейной комбинации признаков объектов, чтобы она лучше всего приближала значения целевой переменной в смысле некоторой функции потерь (обычно среднеквадратичного отклонения).

Таким образом, пусть имеется вектор значений целевой переменной  $y = (y_i)_{i=1}^N \in \mathbb{R}_N$  и матрица признаков  $X = (x_i)_{i=1}^N \in \mathbb{R}^{N \times D}$ ,  $x_i \in \mathbb{R}^D$ . Функция потерь имеет вид:

$$L(w) = \frac{1}{N} \|y - Xw\|_2^2 \rightarrow \min_w, \quad (7)$$

где  $w$  — искомые веса.

К решению такой задачи существует несколько подходов. Во-первых, можно получить аналитическое решение. Действительно, градиент функции потерь имеет вид:

$$\nabla L = \frac{2}{N} X^T (Xw - y), \quad (8)$$

а вторая производная:

$$L'' = \frac{2}{N} X^T X. \quad (9)$$

Функция потерь квадратичная, следовательно имеет один экстремум, который является точкой минимума, так как матрица  $\frac{2}{N} X^T X$  положительно определена (для любого  $X$  с линейно независимыми столбцами). Таким образом, по необходимому условию экстремума, искомое  $w$  получается как решение матричного уравнения:

$$X^T (Xw - y) = 0 \quad (10)$$

$$w = (X^T X)^{-1} X^T y. \quad (11)$$

Такое решение будет точным, однако, его вычисление включает в себя

операцию обращения матрицы, что вычислительно дорого и может быть неприемлемо для большого объема данных. Более того, можно столкнуться с проблемой плохой обусловленности матрицы, что приведет к неустойчивости решения.

Поэтому другой способ заключается в решении задачи оптимизации методом градиентного спуска. Зная градиент функции потерь, само решение можно вычислить следующим итеративным процессом:

$$w_{k+1} = w_k - \eta \cdot \nabla L(w_k), \quad (12)$$

где  $\eta$  — скорость обучения — контролирует величину шага в направлении антиградиента.

При помощи этого метода можно получать решение с некоторой точностью, но со значительно меньшим использованием вычислительных ресурсов.

Проблема описанного выше подхода (для любого метода решения) заключается в том, что на веса не накладывается никаких дополнительных ограничений. Это может привести, например, к их неограниченному росту при наличии коррелированных признаков, но, что более важно, каждому признаку будет приписан какой-то ненулевой вклад в целевую функцию. Этот вклад может быть очень маленьким, но его наличие не позволяет, например, просто откинуть наименее важные признаки, так как это потребует пересчета оставшихся весов и незначимые признаки могут возникнуть вновь.

Поэтому возникает потребность в алгоритмах разреженной регрессии, частью которых является отбор признаков и по результатам которых можно точно отделить признаки, не вносящие вклада в целевую функцию.

### 2.1.2.1 Lasso

Одним из способов решения задачи разреженной регрессии является использование регуляризации — штрафа на величину весов, который добавляется в функцию потерь. Наиболее распространены  $L_1$  и  $L_2$  регуляризации, в которых штраф имеет вид  $L_1$  нормы  $\|w\|_1 = \sum_i |w_i|$  и  $L_2$  нормы  $\|w\|_2 = \sqrt{\sum_i |w_i|^2}$  весов соответственно. В отличие от  $L_2$  регуляризации, которая приводит только к уменьшению величины весов,  $L_1$  регуляризация позволяет обнулять коэффициенты при самых незначимых

признаках. В результате получается алгоритм Lasso — Least absolute shrinkage and selection operator [4].

Вместе с добавленным штрафом, функция потерь выглядит следующим образом:

$$L(w) = \frac{1}{N} \|y - Xw\|_2^2 + \alpha \|w\|_1, \quad (13)$$

где  $\alpha$  — коэффициент регуляризации.

Технически,  $L_1$  норма недифференцируема. Однако недифференцируема она только в одной точке (в нуле). Применительно к машинному обучению, можно считать, что вероятность встречи точного нуля крайне мала, и пренебречь этим. Тогда производная  $L_1$  нормы — это знаковая функция sign, доопределенная в нуле (при помощи 1 или  $-1$ ). Таким образом градиент функции потерь:

$$\nabla L(w) = \frac{2}{N} X^T(Xw - y) + \alpha \text{sign}(w). \quad (14)$$

Для получения решения используем метод градиентного спуска.

### 2.1.2.2 STLSQ

Помимо классического Lasso авторы алгоритма идентификации предлагают собственный алгоритм разреженной регрессии STLSQ — Sequential Thresholded Least-Squares [3].

Алгоритм заключается в следующем. Найдем решение задачи регрессии при помощи МНК — метода наименьших квадратов — любым способом. Такое решение не будет разреженным, но некоторые коэффициенты в нем будут меньше других. Обнулим коэффициенты, меньшие некоторого порогового значения  $\alpha$ . После этого снова найдем решение МНК, но уже для оставшихся коэффициентов, и опять обнулим коэффициенты. Эта процедура уточнения и отбора коэффициенты повторяется, пока все ненулевые не станут больше  $\alpha$ . Если произошло обнуление вообще всех коэффициентов, значит порог  $\alpha$  выбран слишком большим и его надо уменьшить. Псевдокод алгоритма представлен на рисунке 1.

**Исходные параметры:** пороговое значение  $\alpha$

**Входные данные:** матрица признаков  $X$ , вектор целевой переменной  $y$

**Выходные данные:** вектор весов  $w$

```
1  $w = \text{solve}(X, y); //$  функция solve использует МНК
2 до тех пор, пока вектор  $w$  изменяется выполнять
3    $smallinds = |w| < \alpha;$ 
4    $w[smallinds] = 0;$ 
5    $biginds = \neg smallinds;$ 
6    $w[biginds] = \text{solve}(X[biginds], y);$ 
```

Рисунок 1 – Псевдокод алгоритма STLSQ

Приведенные выше алгоритмы описаны для одной целевой функции, однако, они легко обобщаются на случай нескольких целевых функций. Lasso, как и другие алгоритмы, основанные на градиентном спуске, практически не требует изменений, так как градиент можно вычислять для любой размерности. В случае STLSQ алгоритм надо выполнять отдельно для каждого вектора целевых функций и весов.

### 2.1.3 Численное дифференцирование

#### 2.1.3.1 Конечно-разностный метод

Простейшими методами численного дифференцирования является семейство методов, использующих конечные разности. В таких методах оператор дифференцирования аппроксимируется отношением конечных разностей, которое получается путем дифференцирования аппроксимирующего многочлена. Используя многочлены различных степеней, можно получить конечно-разностные формулы соответствующего порядка аппроксимации.

Пусть функция  $f(x)$  задана таблично  $f_i = f(t_i)$ ,  $i = \overline{1, n}$  на регулярной сетке с шагом  $x_{i+1} - x_i = h$ . Тогда для вычисления производной первого порядка в тех же точках  $f'_i = f'(t_i)$ ,  $i = \overline{1, n}$  будем использовать следующие конечно-разностные формулы:

$$f'_1 = \frac{-3f_1 + 4f_2 - f_3}{2h} \quad (15)$$

$$f'_i = \frac{f_{i+1} - f_{i-1}}{2h}, \quad i = \overline{2, n-1} \quad (16)$$

$$f'_n = \frac{f_{n-2} - 4f_{n-1} + 3f_n}{2h}. \quad (17)$$

Такие формулы обеспечивают второй порядок аппроксимации.

Значительным недостатком конечно-разностного метода является то, что при наличии шумовой составляющей в используемых данных  $x_i$ , шумовая составляющая усиливается, что приводит к некорректным значениям производной. Таким образом, с точки зрения практического использования этот метод имеет очень узкую область применимости.

### 2.1.3.2 Регуляризация полной вариации

Более устойчивый к шуму метод можно получить, используя регуляризацию самого процесса дифференцирования, а именно — регуляризацию полной вариации.

Полная вариация является обобщением понятия длины кривой, задаваемой произвольной функцией  $f : [a, b] \rightarrow \mathbb{R}^n$ . Если  $f \in C^1$ , то есть имеет непрерывную первую производную,  $f'$  является функцией ограниченной вариации, которая вычисляется по формуле:

$$\int_a^b ||f'(x)|| d(x), \quad (18)$$

где  $||\cdot||$  — некоторая норма в пространстве  $\mathbb{R}^n$ .

Задачу дифференцирования функции  $f' = u$  на отрезке  $[0, L]$  можно свести к интегральному уравнению Вольтерра первого рода:

$$\int_0^x u(s) d(s) = f(x) - f(0), \quad (19)$$

где  $x \in [0, L]$ .

Обозначим  $Au(x) = \int_0^x u$  — оператор интегрирования, также

предположим, что  $f(0) = 0$  (на практике это достигается вычитанием  $f(0)$  из  $f$ ):

$$Au = f. \quad (20)$$

Это уравнение относится к некорректным в силу высокой чувствительности решения к малым возмущениям правой части. Непосредственное решение данного уравнения не имеет смысла. Вместо этого будем решать задачу минимизации функционала:

$$F(u) = DF(Au - f) + \alpha R \longrightarrow \min_u, \quad (21)$$

где  $DF(Au - f)$  — компонента, отвечающая за точность решения;  
 $\alpha$  — коэффициент регуляризации;  
 $R$  — регуляризационная компонента.

Функционал  $DF$  обычно представляет из себя квадрат  $L^2$  нормы,  $DF = \int_0^L |\cdot|^2$ , так как предполагается, что  $f$  имеет в своем составе белый гауссов шум. Для шумов с другим распределением необходимо использовать другие функционалы.

С этого момента существует несколько подходов к решение этой задачи, связанных с разными формами регуляризации.

### **Первый метод**

Первый способ заключается в регуляризации — в виде квадрата  $L^2$  нормы производной функции  $f$  [7]:

$$R = \int_0^L |f'|^2. \quad (22)$$

Таким образом происходит регуляризация полной вариации  $f$  и функционал  $F$  принимает вид:

$$F(u) = \int_0^L |Au - f|^2 + \alpha \int_0^L |f'|. \quad (23)$$

Аппроксимируя операторы интегрирования и дифференцирования, можно переписать задачу в матрично-векторной форме (о конкретном виде дискретных аналогов операторов будет сказано позже):

$$F(u) = \|Au - f\|_2^2 + \alpha\|Df\|_2^2, \quad (24)$$

где  $u$  — искомый вектор производных,  
 $f$  — вектор приращений функции,  
 $A$  — матричный оператор интегрирования,  
 $D$  — матричный оператор дифференцирования,  
 $\|\cdot\|_2$  — евклидова норма вектора.

Минимизируемая функция квадратичная, следовательно задача минимизации может быть решена с использованием необходимого условия экстремума:

$$\nabla F(u) = 0. \quad (25)$$

В результате задача сводится к решению системы линейных алгебраических уравнений относительно вектора  $u$ :

$$Bu = b, \quad (26)$$

где  $B = A^T A + \alpha D^T D$ ,  
 $b = A^T u$ .

### **Второй метод**

Для второго способа вместо квадрата  $L^2$  нормы используем  $H_1^1$  норму искомой производной  $u$  [8]:

$$R = \int_0^L |u'|. \quad (27)$$

Тогда регуляризации подвергается полная вариация  $u$  и функционал  $F$  имеет вид:

$$F(u) = \int_0^L |Au - f|^2 + \alpha \int_0^L |u'|. \quad (28)$$

Такую задачу минимизации уже нельзя решить, используя только необходимое условие экстремума, и необходимо применять другие методы оптимизации. Простейшим способом будет использовать метод градиентного спуска. Для такой задачи этот метод сводится к эволюции дифференциального уравнения до стационарного состояния:

$$u_t = \alpha \frac{d}{dx} \frac{u'}{|u'|} - A^T(Au - f), \quad (29)$$

где  $A^T v(x) = \int_x^L v$  — сопряженный оператор к  $A$ .

Во избежании деления на ноль,  $|u'|$  в знаменателе можно заменить на  $\sqrt{(u')^2 + \varepsilon}$ , для некоторого малого  $\varepsilon > 0$ .

Проблема градиентного спуска состоит в медленной сходимости. Поэтому вместо него предлагается использовать метод запаздывающей диффузии (lagged diffusivity) [9; 10]. Основная идея этого метода заключается в замене нелинейного дифференциального оператора  $u \mapsto (d/dx)(u'/|u'|)$  линейным оператором  $u \mapsto (d/dx)(u'/|u'_k|)$ .

Для этого необходимо произвести дискретизацию. Предполагаем, что функция  $f$  задана на регулярной сетке с шагом  $h$  в точках  $\{x_i\}_0^n = \{0, h, 2h, \dots, L\}$ , ее производную  $u$  будем искать в точках  $\{x_i\}_0^n \cup \{x_{-1}\} = \{-h, 0, h, 2h, \dots, L\}$  (первая точка нужна только для работы алгоритма и в результат не входит). Производную  $u$  будем вычислять между точками сетки, как центральные разности:

$$Du(x_i + \frac{h}{2}) = \frac{u(x_{i+1}) - u(x_i)}{h}, \quad i = \overline{-1, n-1}. \quad (30)$$

Интеграл от  $u$  будем вычислять в точках  $\{x_i\}_0^n$ , используя метод трапеций:

$$Au(x_i) = \sum_{j=-1}^{i-1} \frac{u(x_j) + u(x_{j+1})}{2} h, \quad i = \overline{0, n}. \quad (31)$$

Таким образом, матрицы операторов интегрирования и

дифференцирования, обе размером  $n \times (n + 1)$ , выглядят следующим образом:

$$D_{n,n+1} = \frac{1}{h} \begin{pmatrix} -1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix} \quad (32)$$

$$A_{n,n+1} = h \begin{pmatrix} 0.5 & 0.5 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0.5 & 1 & 0.5 & 0 & \cdots & 0 & 0 & 0 \\ 0.5 & 1 & 1 & 0.5 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0.5 & 1 & 1 & 1 & \cdots & 1 & 0.5 & 0 \\ 0.5 & 1 & 1 & 1 & \cdots & 1 & 1 & 0.5 \end{pmatrix}. \quad (33)$$

Подобные матрицы и подход с дополнительной точкой можно также использовать и в первом методе.

Метод запаздывающей диффузии представляет из себя итеративный процесс. На каждой итерации  $k$  используются следующие матрицы:

$$E_k = \text{diag}((Du_k)^2 + \varepsilon)^{-1/2}, \quad (34)$$

$$L_k = h D^T E_k D, \quad (35)$$

$$H_k = A^T A + \alpha L_k, \quad (36)$$

где  $\text{diag}(v)$  — диагональная матрица со значениями вектора  $v$  на главной диагонали;

$H_k$  — аппроксимация гессиана  $F$  в точке  $u_k$ .

Переход к следующей итерации:

$$g_k = A^T(Au_k - f) + \alpha L_k u_k, \quad (37)$$

$$H_k s_k = g_k, \quad (38)$$

$$u_{k+1} = u_k + s_k, \quad (39)$$

где  $s_k$  — решение системы линейных алгебраических уравнений (38).

Начальным значением  $u$  может быть нулевой вектор или вектор производных, полученных методом конечных разностей. Завершение итеративного процесса можно произвести по достижению заданного числа итераций, или при незначительном изменении вектора  $u_k$ , относительно предыдущего значения. Псевдокод алгоритма представлен на рисунке 2.

	<b>Исходные параметры:</b> коэффициент регуляризации $\alpha$ , число итераций $K$
	<b>Входные данные:</b> вектор значений функции $f$ , шаг регулярной сетки $h$
	<b>Выходные данные:</b> вектор производных $u$
1	$n =$ размер вектора $f$ ;
2	$u = \underbrace{(0 \ 0 \ \cdots \ 0)}_{n+1}^T;$
3	<b>цикл</b> $k = 1$ <b>до</b> $K$ <b>выполнять</b>
4	$E = \text{diag}(((Du)^2 + \varepsilon)^{-1/2});$
5	$L = hD^T E D;$
6	$H = A^T A + \alpha L;$
7	$g = A^T(Au - f) + \alpha L u;$
8	$s = \text{solve}(H, g); //$ функция <code>solve</code> решает СЛАУ
9	$u = u + s;$
10	<b>возвратить</b> $(u_1 \ u_2 \ \cdots \ u_{n+1})^T$

Рисунок 2 – Псевдокод второго метода алгоритма дифференцирования

## 2.2 Используемые технологии

Основной и единственный язык реализации — Python [11]. Это очень удобный язык программирования для работы с данными и реализации алгоритмов машинным обучением.

Так как большинство алгоритмов реализовывались полностью с нуля, в них активно используются две основные библиотеки научных вычислений — NumPy [12] и SciPy [13].

При разработке использовался объектно-ориентированный подход на основе базовых классов библиотеки Scikit-learn [14]. Эта библиотека предоставляет крайне удобную инфраструктуру для алгоритмов машинного обучения.

При тестировании и анализе методов использовалась среда разработки Jupyter [15].

Для визуализации использовались библиотеки Matplotlib [16] и Seaborn [17].

## 2.3 Описание программной разработки

Разработанное решение реализовано в формате модуля языка Python. Структура модуля изображена на рисунке 3, служебные файлы (такие как `__init__.py`) опущены.

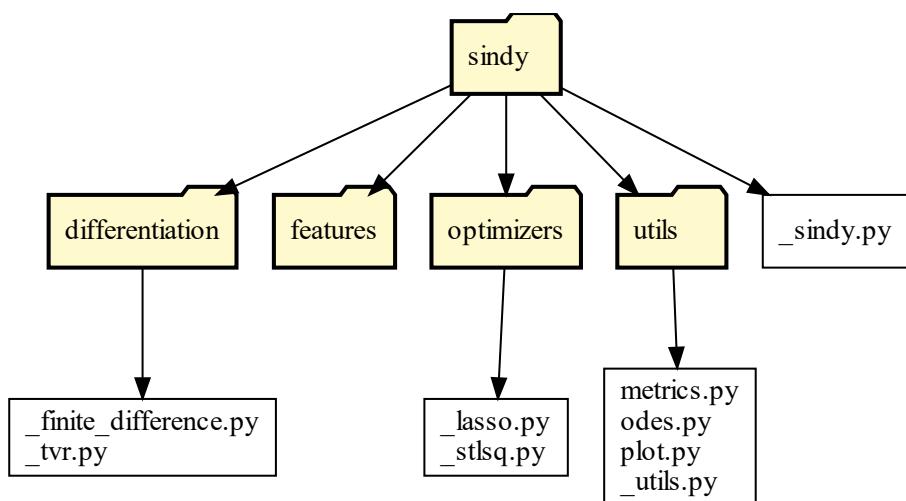


Рисунок 3 – Структура модуля

Для удобства разработки, основные части алгоритма реализованы в отдельных подмодулях.

Подмодуль `optimizers` содержит реализацию алгоритмов разреженной регрессии. В качестве Lasso используется `sklearn.linear_model.Lasso` с зафиксированным параметром `fit_intercept=False`, который убирает свободный член из формулы регрессии (считаем, что он задается в матрице признаков, если необходим). STLSQ реализован как `sklearn.base.RegressorMixin`.

Подмодуль `differentiation` содержит алгоритмы численного дифференцирования, `_finite_difference.py` — метод конечных разностей, `_tvr.py` — методы регуляризации полной вариации. Первый метод назван FastTVR, так как не является итеративным и содержит только одну процедуру решения СЛАУ, второй метод — просто TVR. При описании результатов в подразделе 3.1 методы указаны именно с этими именами. Все алгоритмы дифференцирования реализованы как `sklearn.base.TransformerMixin`, методы регуляризации полной вариации используют матрицы дискретных операторов в явном виде.

Подмодуль `features` не содержит никаких реализаций, поскольку в работе в итоге исследовались только системы с полиномиальными правыми частями, и для получения матрицы признаков использовался `sklearn.preprocessing.PolynomialFeatures`.

Подмодуль `utils` содержит различные вспомогательные классы и функции. В частности метрики, используемые системы ОДУ, функции для генерации и визуализации данных.

Алгоритм идентификации реализован в `_sindy.py` как `sklearn.base.BaseEstimator`. «Обученный» алгоритм может предсказывать значения левых частей системы, а также описать саму систему или составить экземпляр класса `sindy.utils.Equation`, который затем можно эволюционировать для получения траектории.

Исходный код программной разработки доступен в репозитории по ссылке в приложении А.

## 3 РЕЗУЛЬТАТЫ РАБОТЫ

### 3.1 Результаты

#### 3.1.1 Анализ методов разреженной регрессии

Для анализа алгоритмов линейной регрессии при помощи `sklearn.datasets.make_regression` сгенерирован датасет из 100 объектов, с одним значением целевой переменной. Каждый объект описывается 100 признаками, из которых значимыми являются только 10. К значениям признаков добавлен гауссовский шум с дисперсией 5. Визуализация истинных значений коэффициентов представлена на рисунке 4.

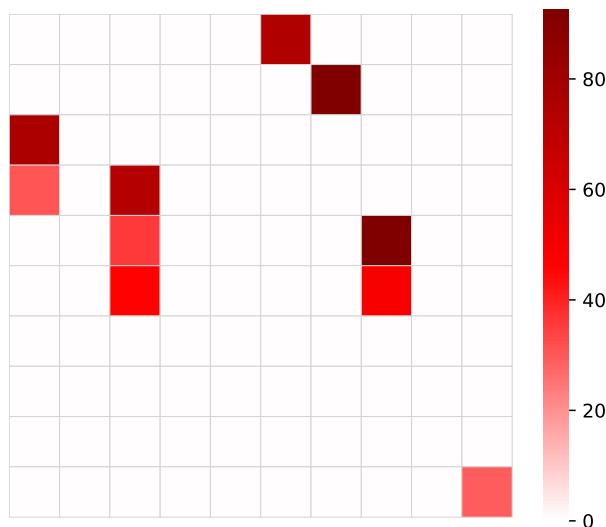


Рисунок 4 – Истинные коэффициенты

В качестве контрольного примера, на рисунке 5 приведены результаты работы обычной линейной регрессии без регуляризации. Для оценки результатов самой регрессии используются такие метрики, как максимальная ошибка, средняя абсолютная ошибка, среднеквадратичная ошибка, коэффициент детерминации. Как и ожидалось, такой метод хорошо предсказывает значения целевой переменной, но совершенно не справляется с отбором признаков. Хорошие результаты предсказаний также говорят о том, что этот метод подстраивается под шум.

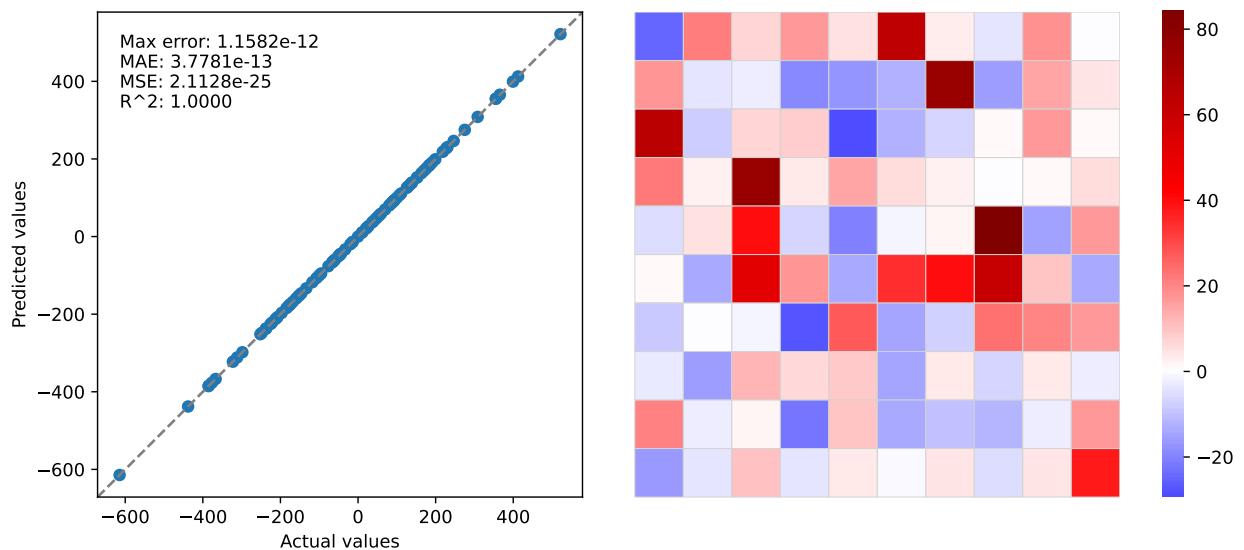


Рисунок 5 – Результаты работы линейной регрессии без регуляризации

На рисунке 6 приведены результаты работы Lasso — линейной регрессии с  $L_1$  регуляризацией. Коэффициент регуляризации  $\alpha = 1$ .

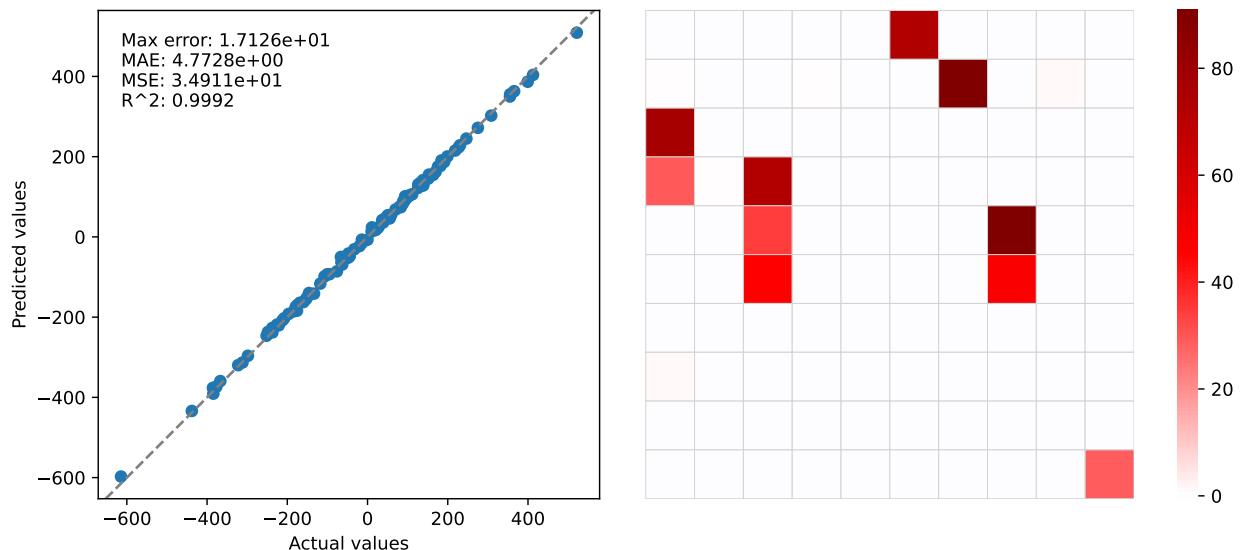


Рисунок 6 – Результаты работы Lasso

На рисунке 7 приведены результаты работы STLSQ. Пороговое значение  $\alpha = 2$ .

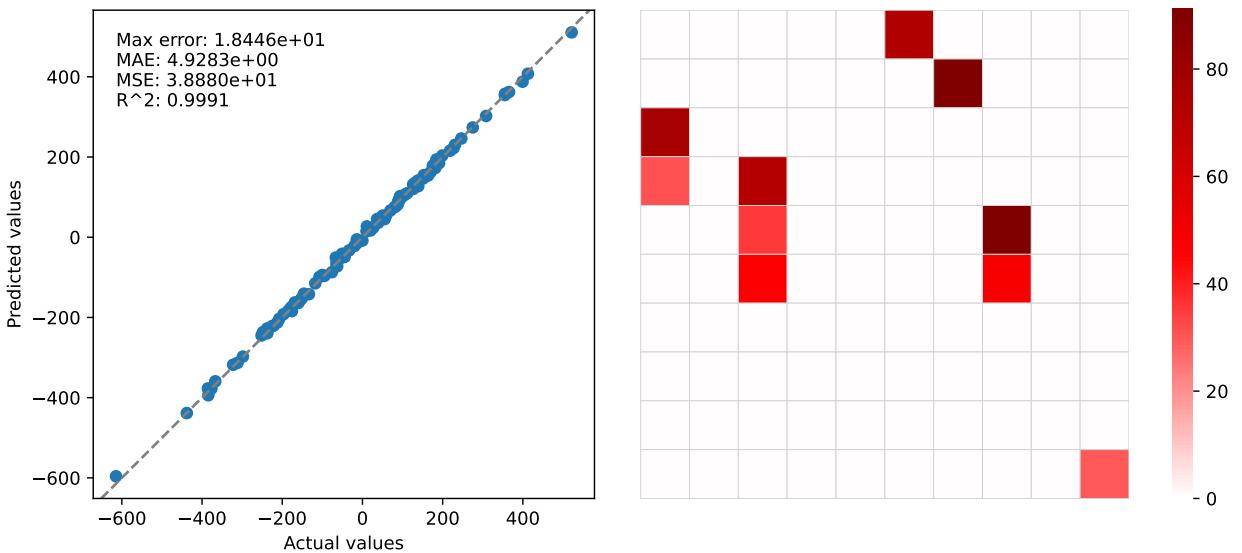


Рисунок 7 – Результаты работы STLSQ

В результате тестирования можно сказать, что методы разреженной регрессии действительно справляются с поставленной задачей. Ценой этому служит необходимость подбора параметров и уменьшение точности. Но, так как они используются в структурной идентификации, отбор признаков является более важным результатом, чем качество предсказаний.

### 3.1.2 Анализ методов численного дифференцирования

Использование методов численного дифференцирования, основанных на регуляризации полной вариации, осложняется наличием у этих методов параметров. Оба метода имеют параметр, отвечающий за степень регуляризации, второй метод, будучи итеративным, также содержит ограничение на число итераций. Поэтому в данном пункте будут проанализированы эти методы, проведено сравнение с методом конечных разностей и выявлены закономерности влияния параметров на конечный результат.

Параметр  $\alpha$  — коэффициент регуляризации — несет одинаковую функцию в обоих методах: определяет насколько сильно будет подвергаться регуляризации процесс дифференцирования. Так как речь идет о регуляризации полной вариации, его влияние на производную очевидно: при увеличении  $\alpha$ , полная вариация производной будет уменьшаться, приводя к «уплощению» графика, до тех пор, пока он не выродится в прямую линию. Это подтверждает пример с дифференцированием синусоиды (без

добавочного шума) на рисунке 8.

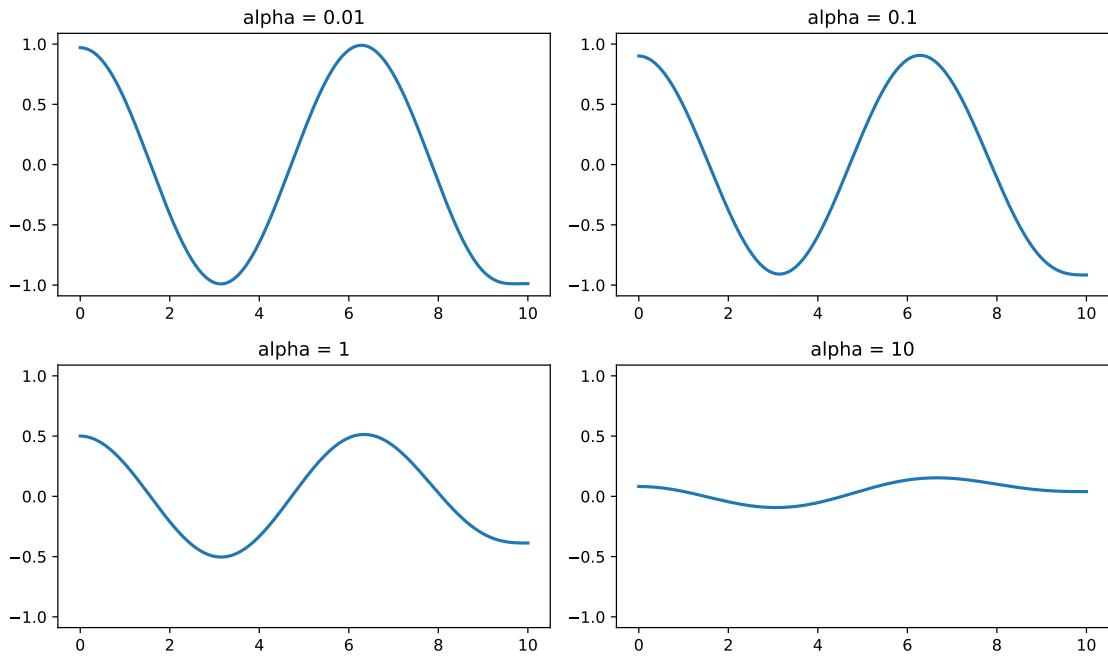


Рисунок 8 – Результаты дифференцирования при различных значениях параметра  $\alpha$

Для итеративного метода параметром также является число итераций. Итеративный процесс сходится к некоторому стационарному состоянию, пример значений производной той же самой синусоиды (уровень шума — 0.05) на разных итерациях показан на рисунке 9. Так как в этом случае речь идет о полной вариации полученной производной, на графиках видно увеличение ступенчатости — количества и размеров областей с постоянными значениями, которые имеют нулевую вариацию.

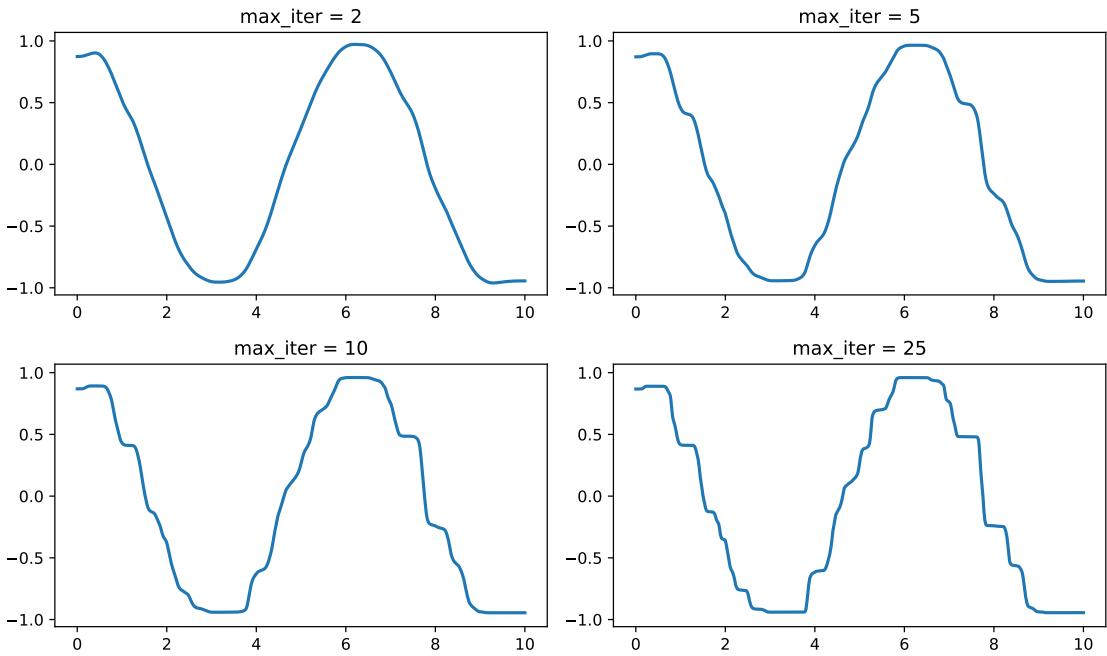


Рисунок 9 – Результаты дифференцирования на различных итерациях

Рассмотрим подробный анализ обоих методов для трех тестовых функций.

### 3.1.2.1 Простая функция с непрерывной производной

В качестве простой функции с непрерывной производной выбрана функция  $\sin(x)$  на отрезке  $[0, 10]$ . Она представлена на рисунке 10.

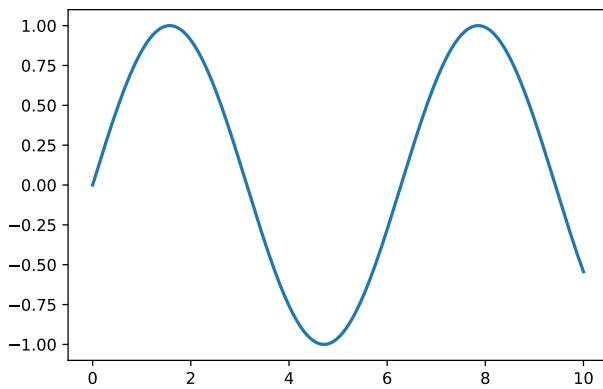


Рисунок 10 – Функция  $\sin(x)$

Для каждого метода рассмотрим среднюю абсолютную ошибку между результатом дифференцирования и точной производной для:

- различных разбиений отрезка и фиксированных параметрах,

- различных уровней шума и фиксированных параметрах,
- различных величин одного из параметров,
- комбинаций уровней шума и значений параметра.

На рисунке 11 представлено тестирование первого метода.

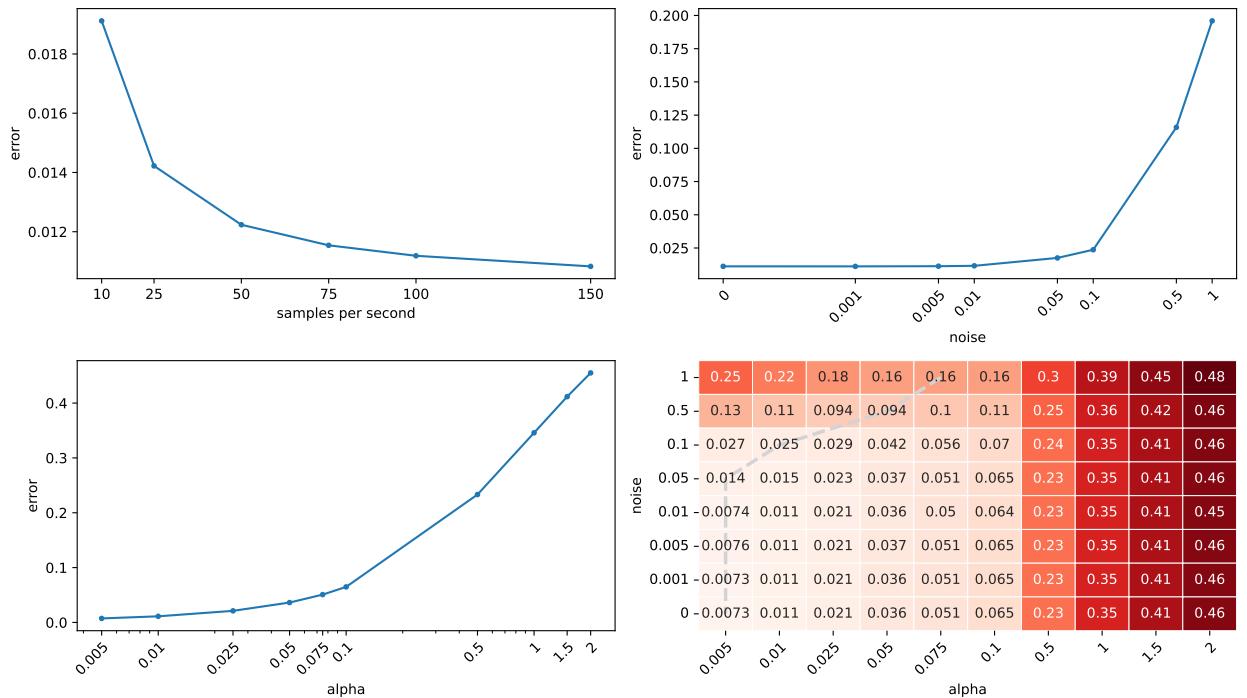


Рисунок 11 – Тестирование первого метода дифференцирования

Увеличение разбиения ожидаемо приводит к увеличению точности. Увеличение уровня шума при неизменных параметрах — к ухудшению результата, возникает задача подбора параметров. Самый большой интерес представляют результаты комбинаций уровня шума и коэффициента регуляризации (серой пунктирной линией обозначена траектория с минимальной ошибкой): большие значения приводят к плохим результатам из-за сильного сглаживания производной, в остальном, можно сказать, что коэффициент регуляризации должен иметь схожий порядок с уровнем шума.

Тестирование второго метода, показанного на рисунке 12, приводит к схожим результатам. Отдельно стоит отметить влияние на результат числа итераций. При низком уровне шума оно практически не влияет, при больших — ухудшает результат. Оптимальное число итераций — 2.

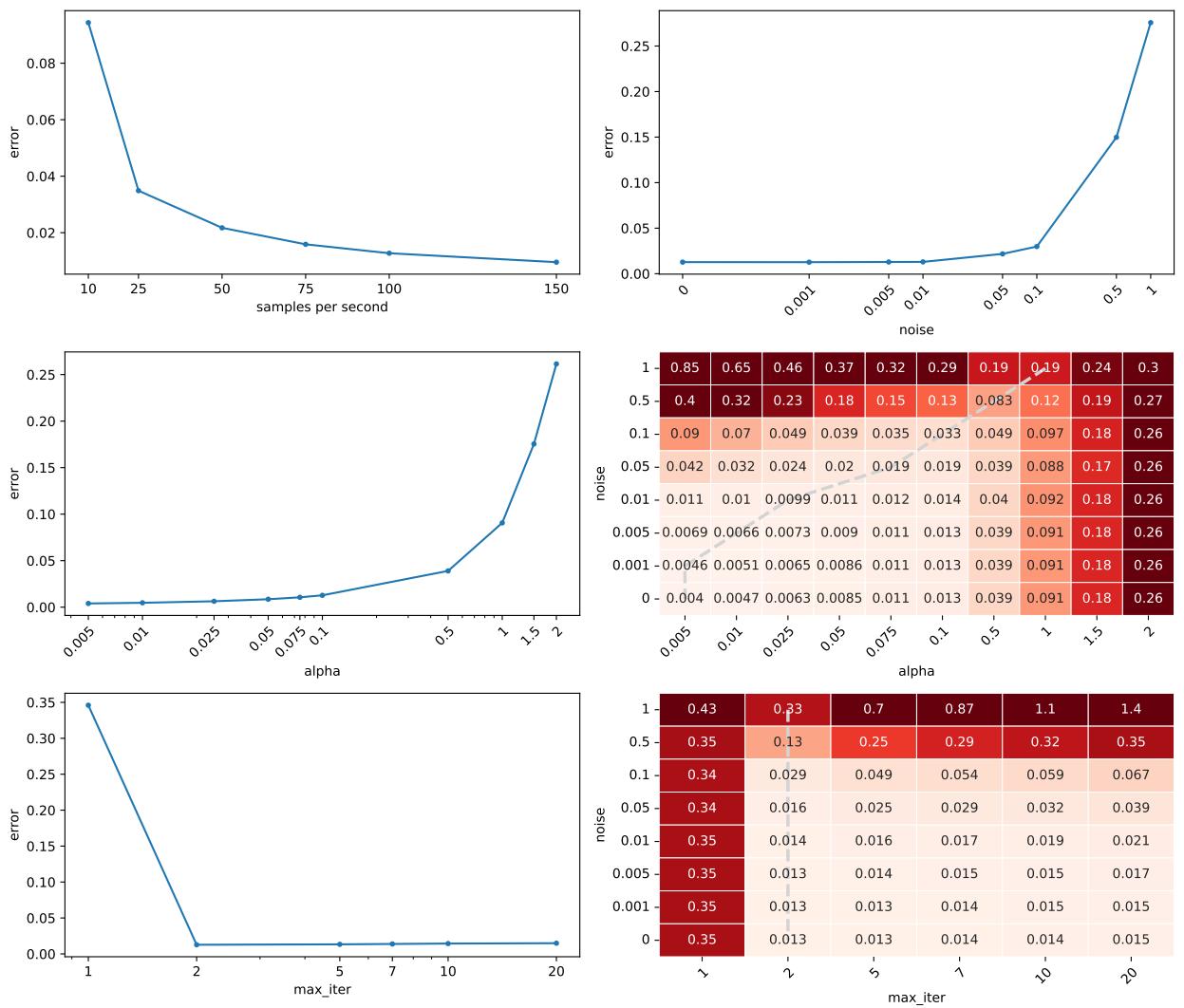


Рисунок 12 – Тестирование второго метода дифференцирования

Таким образом можно подобрать параметры для приемлемого дифференцирования данных с различным уровнем шума. Это показано на рисунке 13. Столбцы соответствуют различным уровням шума (первый столбец — точные значения функции и производной), строки — зашумленным данным, методу конечных разностей, первому и второму методам регуляризации полной вариации соответственно. Очень хорошо видно поведение метода конечных разностей, который перестает справляться с задачей при малейшем шуме.

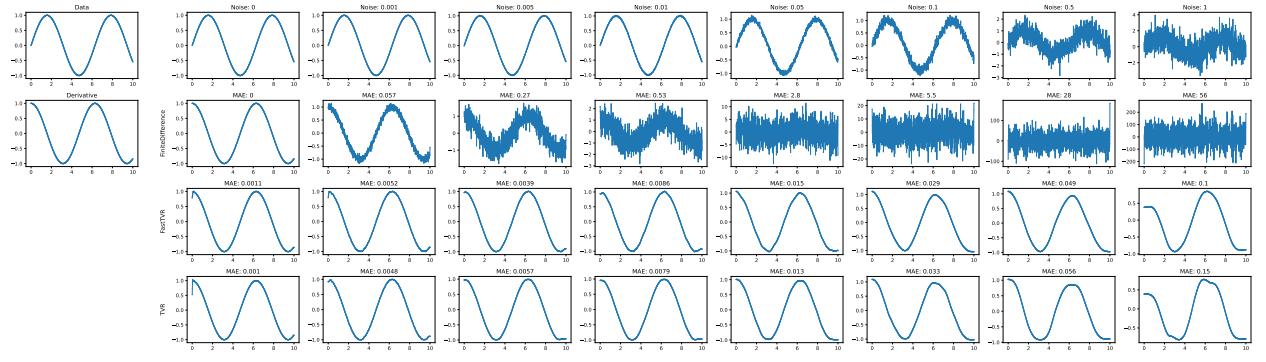


Рисунок 13 – Дифференцирование данных с различным уровнем шума

### 3.1.2.2 Простая функция с разрывной производной

В качестве функции с разрывной производной выбрана функция модуля  $|x|$  на отрезке  $[-1, 1]$ . Она изображена на рисунке 14.

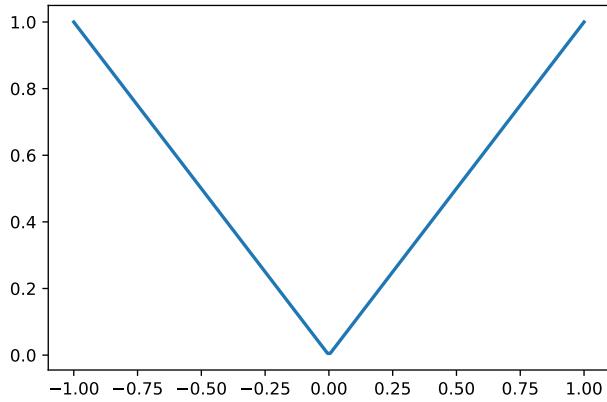


Рисунок 14 – Функция  $|x|$

На рисунках 15, 16 представлены результаты тестирования методов. В целом наблюдается довольно похожая картина, однако, более сложный характер функции несколько осложняет дифференцирование. Соотношение между уровнем шума и коэффициентом регуляризации тоже несколько поменялось, теперь коэффициент должен быть на один-два порядка меньше.

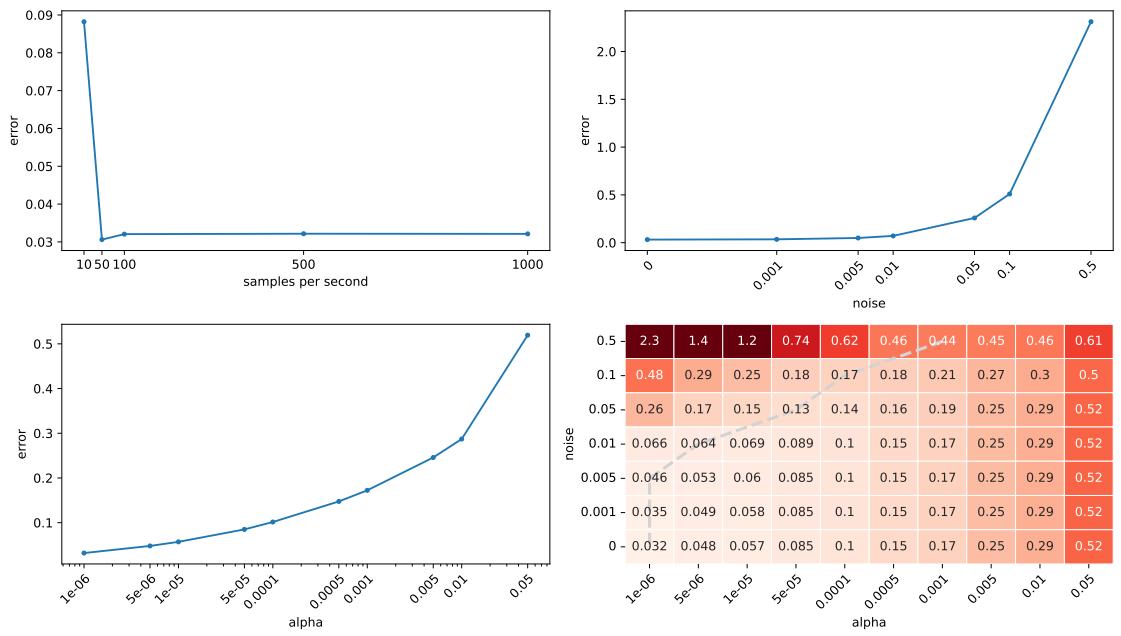


Рисунок 15 – Тестирование первого метода дифференцирования

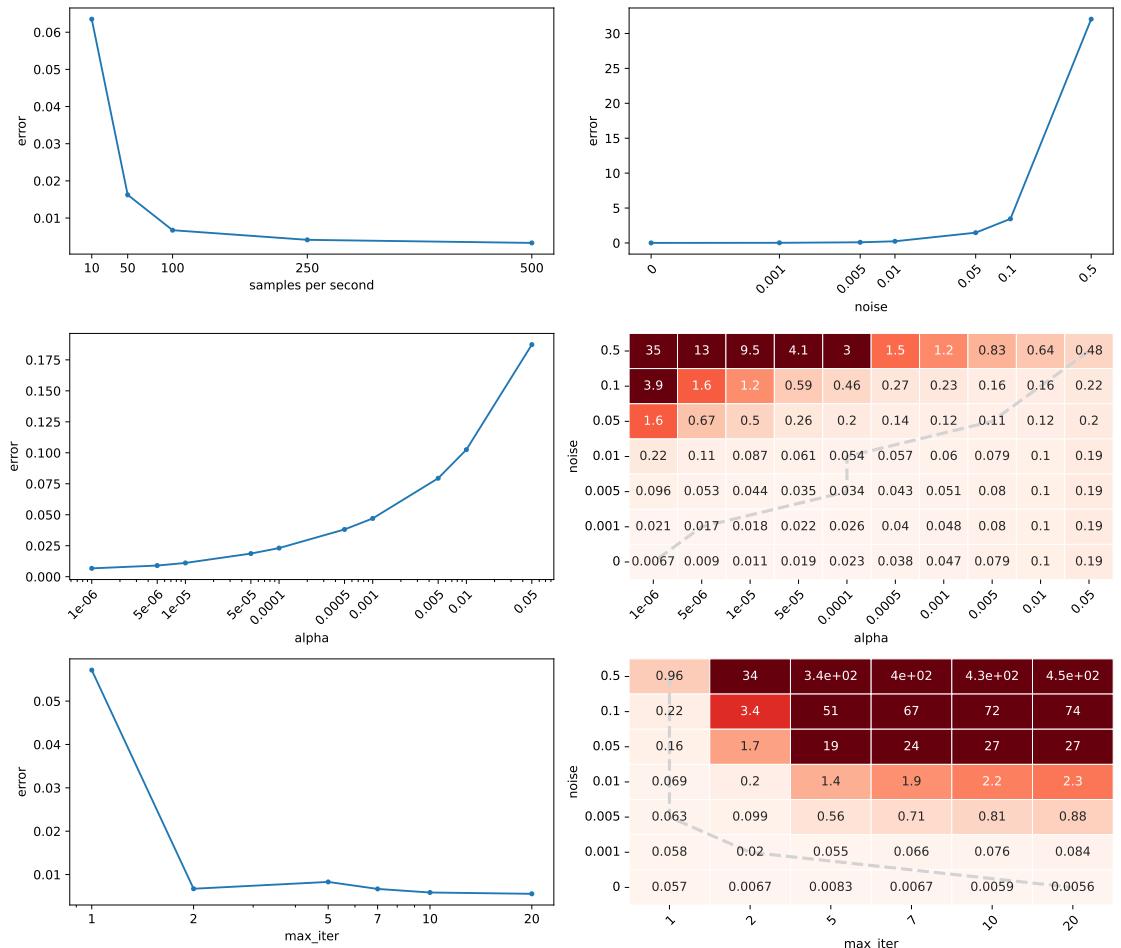


Рисунок 16 – Тестирование второго метода дифференцирования

На рисунке 17 показаны результаты дифференцирования функции с различным уровнем шума. Заметно сглаживание места разрыва с увеличением шума, однако, даже когда исходная функция слабо различима, основной характер производной все еще хорошо заметен.

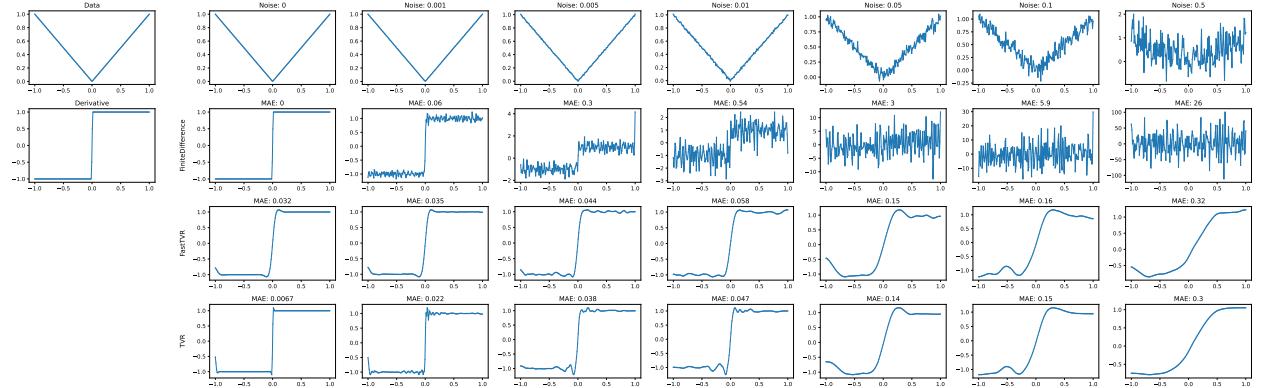


Рисунок 17 – Дифференцирование данных с различным уровнем шума

### 3.1.2.3 Сложная функция с непрерывной производной

В качестве более сложной функции чем синус, но все еще с непрерывной производной, выбран аттрактор Лоренца — решение системы Лоренца. Аттрактор изображен на рисунке 18.

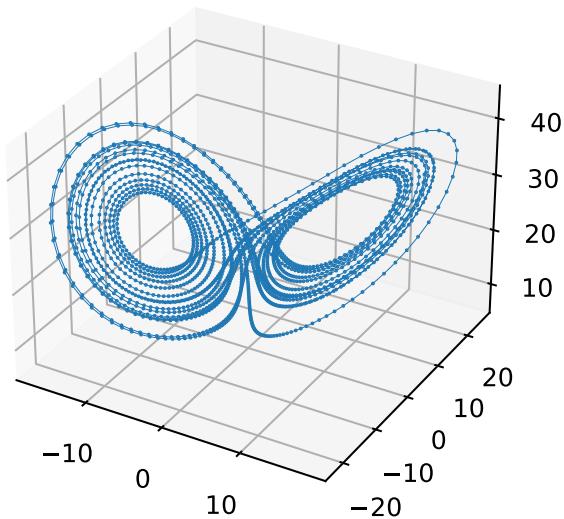


Рисунок 18 – Аттрактор Лоренца

На рисунках 19, 20 представлены результаты тестирования методов. Оптимальная величина параметра  $\alpha$  упала еще на несколько порядков.

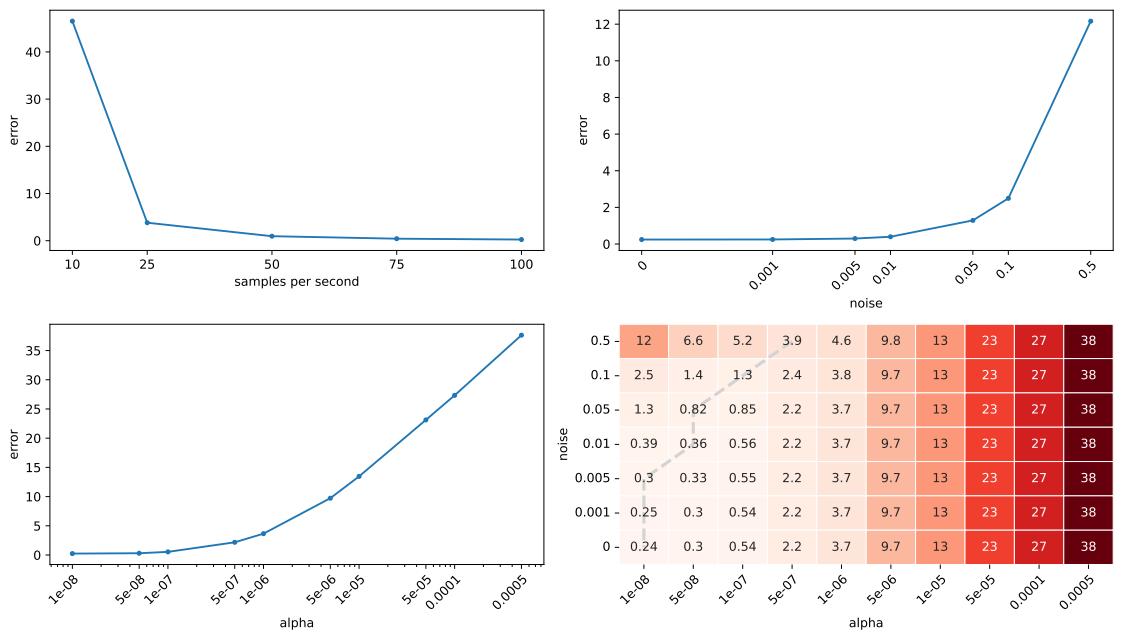


Рисунок 19 – Тестирование первого метода дифференцирования

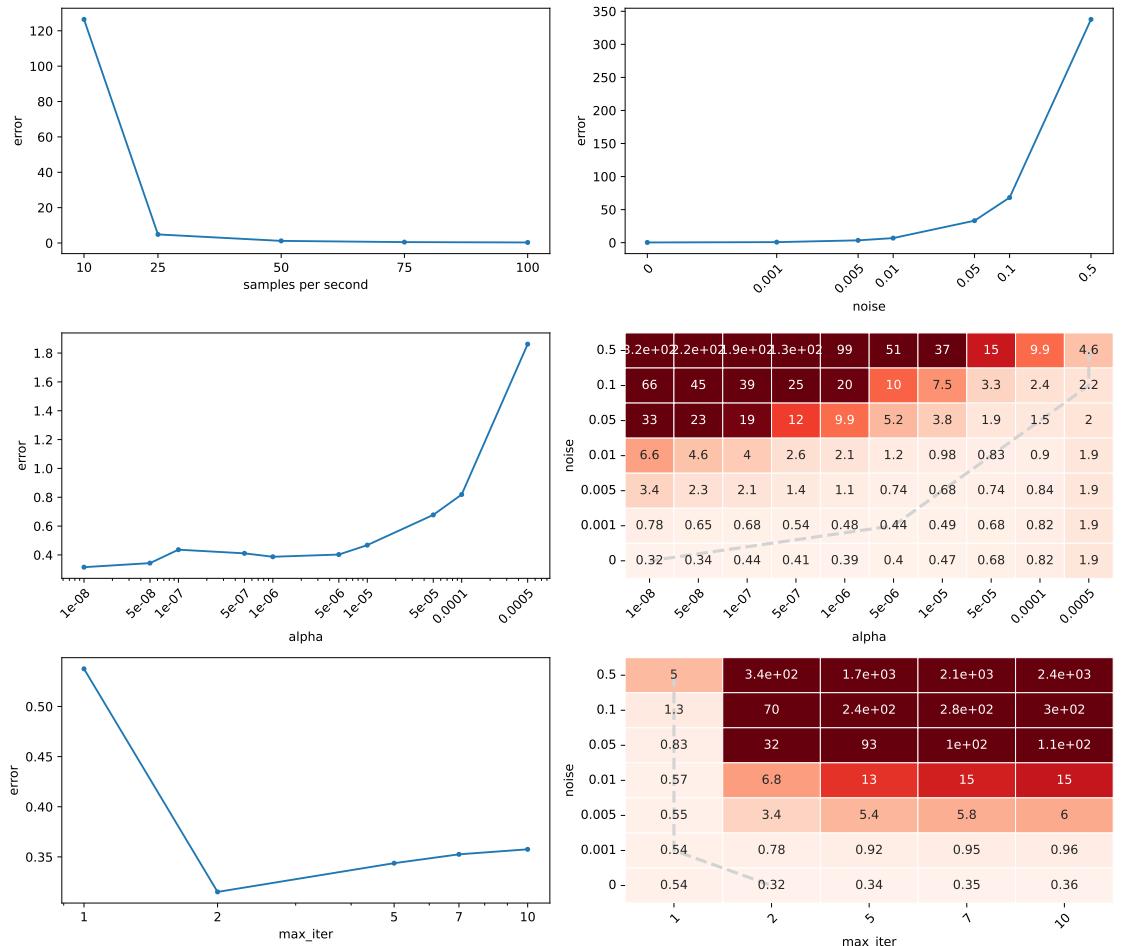


Рисунок 20 – Тестирование второго метода дифференцирования

Однако, как показано на рисунке 21, функция все еще неплохо дифференцируется, что вызывает уверенность в дальнейшей ее успешной идентификации.

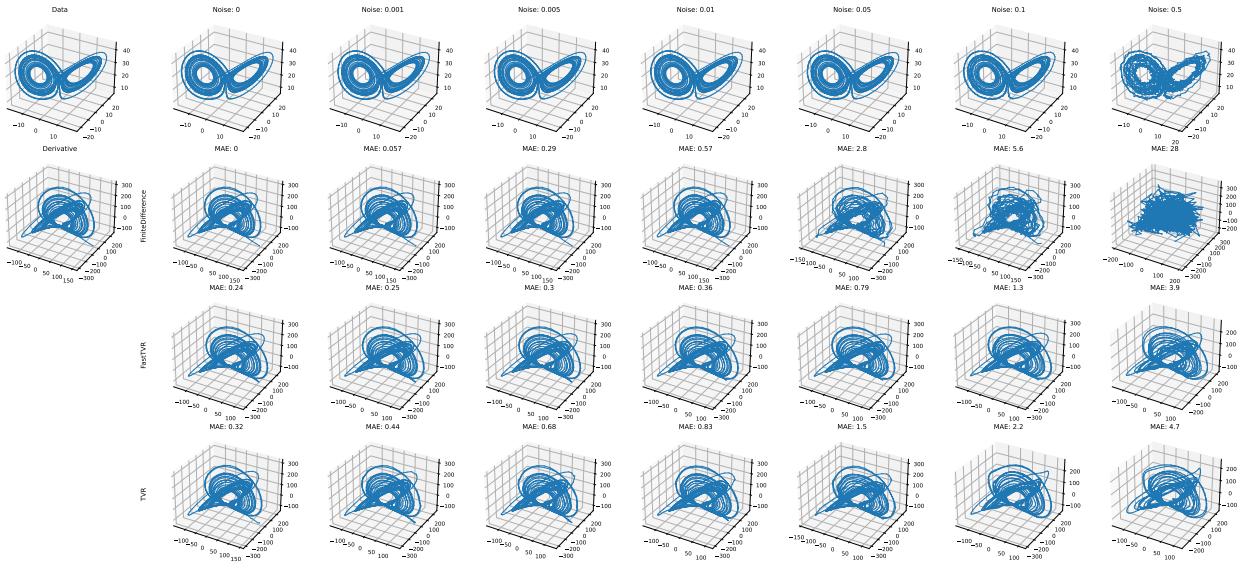


Рисунок 21 – Дифференцирование данных с различным уровнем шума

### 3.1.2.4 Общие выводы

По результатам тестирования можно сказать, что успешно дифференцировать зашумленные данные вполне реально. Анализ результатов позволяет сделать некоторое выводы о подборе параметров.

Во-первых, чем больше измерений значений функции в единицу времени дано, тем точнее будет полученная производная. Изменение шага сетки является способом контроля погрешности еще в методе конечных разностей и здесь не теряет свою актуальность. Во-вторых, для итеративного метода дифференцирования оптимальным числом итераций является 2. После первой итерации производная еще не достаточно приняла свою форму, а после третьей и далее она становится слишком ступенчатой. Наконец, более подробно остановимся на коэффициенте регуляризации — параметре, имеющемуся у обоих методов.

Из графиков видно, что порядок коэффициента приблизительно линейно связан с порядком уровня шума. Это подтверждает и рисунок 22, на котором изображены оптимальные траектории параметра  $\alpha$  в зависимости от уровня шума для нескольких test-функций (в основном систем ОДУ, которые будут использованы в следующем пункте), серой пунктирной линией

изображен примерный тренд.

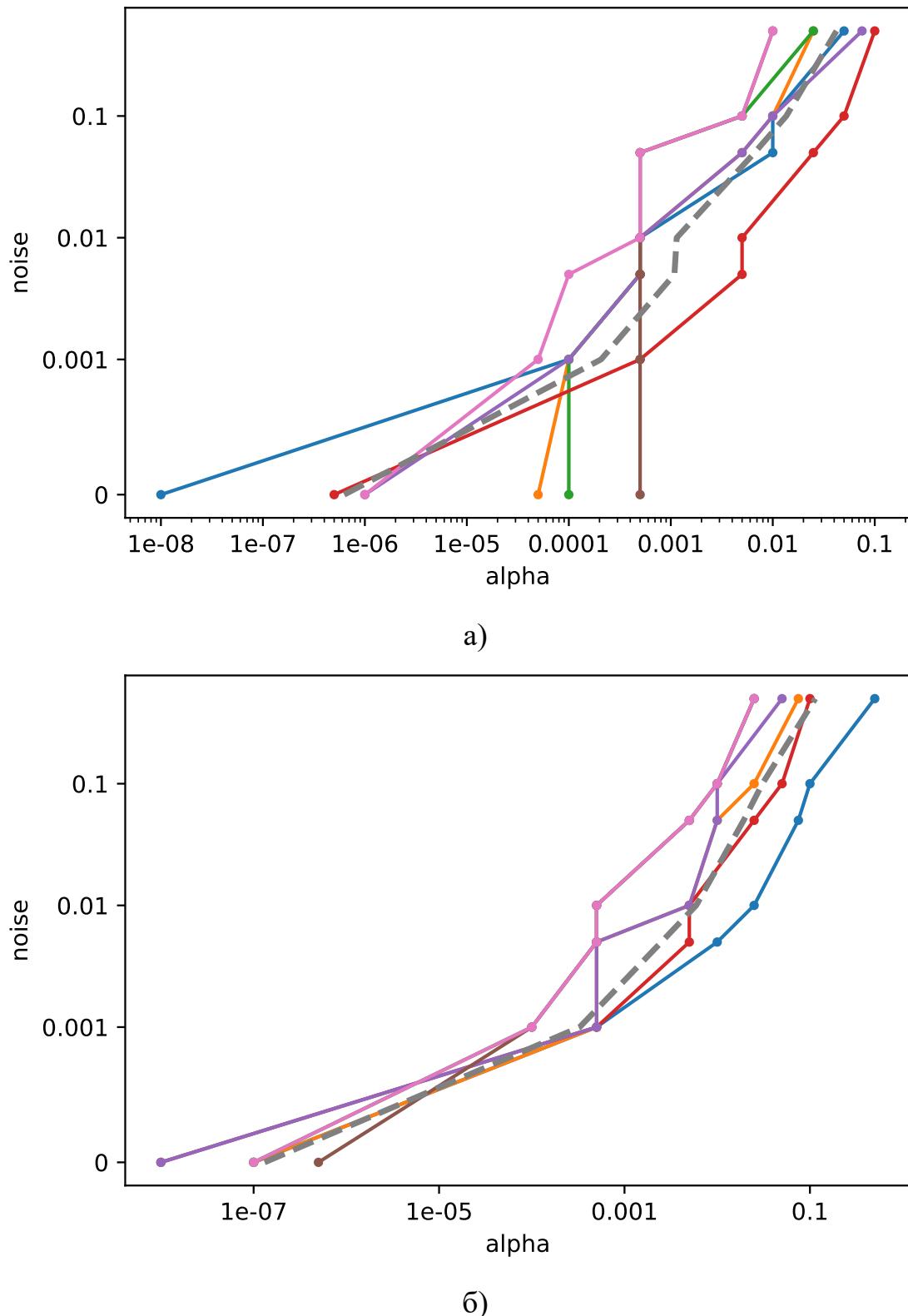


Рисунок 22 – Оптимальные значения параметра  $\alpha$ : а) для первого метода; б) для второго метода

При этом для различных функций порядок параметра может сильно различаться. В целом возникает субъективное ощущение, что чем «сложнее»

функция (с точками разрыва или просто более извилистая), тем меньше должен быть коэффициент регуляризации.

### 3.1.2.5 Замер времени

Как было сказано в описании разработки 2.3, оба метода реализованы с явным использованием матричных операторов дифференцирования и интегрирования. Поэтому от них можно ожидать полиномиальной сложности от квадратичной и выше. На рисунке 23 изображена зависимость времени дифференцирования от числа замеров в единицу времени (измерения проводились на процессоре AMD Ryzen 7 3700U). Действительно, при увеличении числа замеров на порядок, затраченное время увеличивается примерно на два порядка.

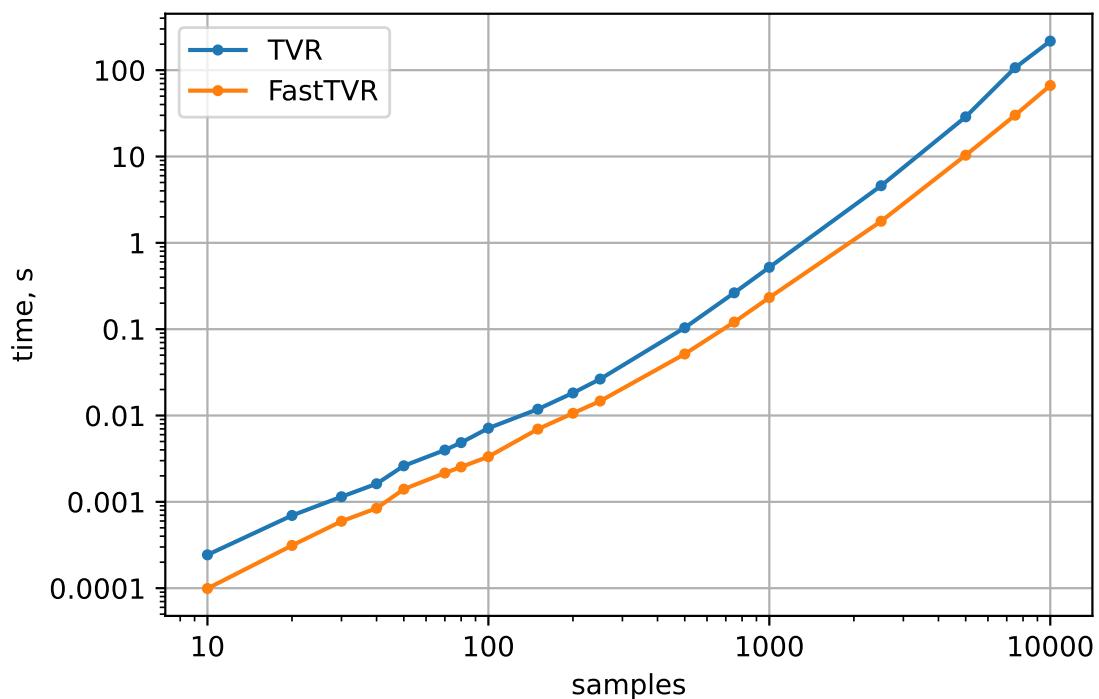


Рисунок 23 – Зависимость времени дифференцирования от количества данных

### 3.1.3 Анализ алгоритма идентификации

Для тестирования алгоритма идентификации использовались различные системы ОДУ 1-ого порядка, найденные в книге [18]. Эволюция систем производилась при помощи `scipy.integrate.solve_ivp`, в котором используется явный метод Рунге-Кутты четвертого порядка.

Оценка идентифицированной системы производилась при помощи нескольких метрик. В оценке задачи определения значимых членов важно учитывать и те из них, которые не идентифицировались, и те, которые идентифицировались ложно. Поэтому элементарной метрикой может являться само количество таких членов (метрика ME):

$$ME = (FN, FP), \quad (40)$$

где  $FN$  — false negative — число потерянных слагаемых,  
 $FP$  — false positive — число ложно определенных слагаемых.

Однако такая метрика может быть не очень удобной, и в качестве метрики, объединяющей эти два числа, выбрана IoU — intersection over union, также известная как коэффициент Жаккара. Эта метрика показывает какая часть из всех фигурирующих слагаемых (предсказанных и истинных) определена верно:

$$J = \frac{TP}{TP + FP + FN}, \quad (41)$$

где  $TP$  — true positive — число верно определенных слагаемых.

После выбора метрик, методы регрессии были протестированы снова, на этот раз применительно к задаче идентификации. Так, выяснилось, что алгоритм Lasso плохо для этого подходит. Если простые системы, например, простейшую систему линейных уравнений (42), еще получается идентифицировать нормально, как это показано на рисунке 24. Пример конкретной восстановленной системы изображен на рисунке 25.

$$\begin{aligned} \dot{x} &= -0.1x + 2y \\ \dot{y} &= -2x - 0.1y \end{aligned} \quad (42)$$

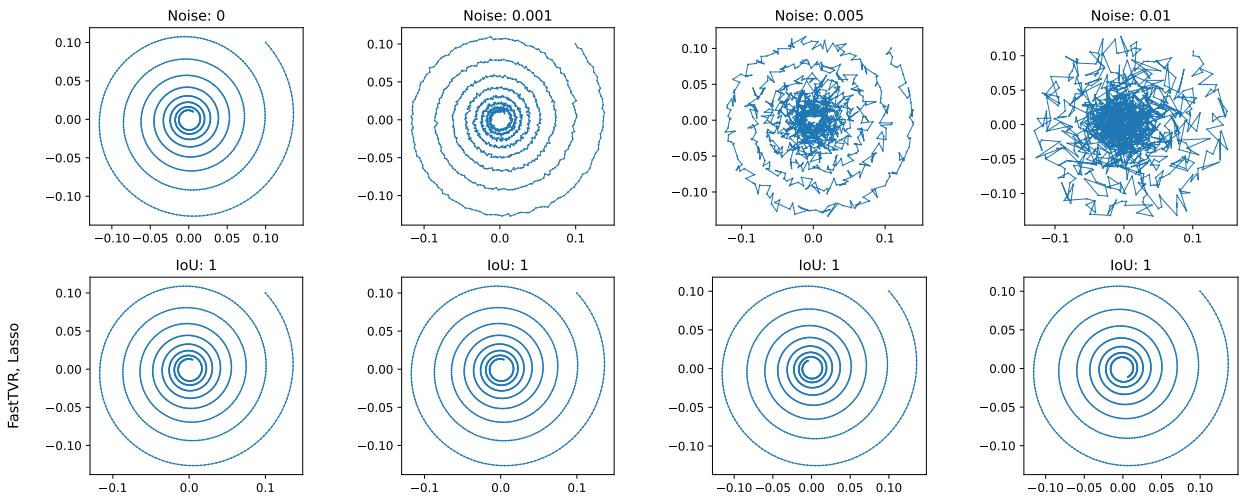


Рисунок 24 – Идентификация системы линейных уравнений

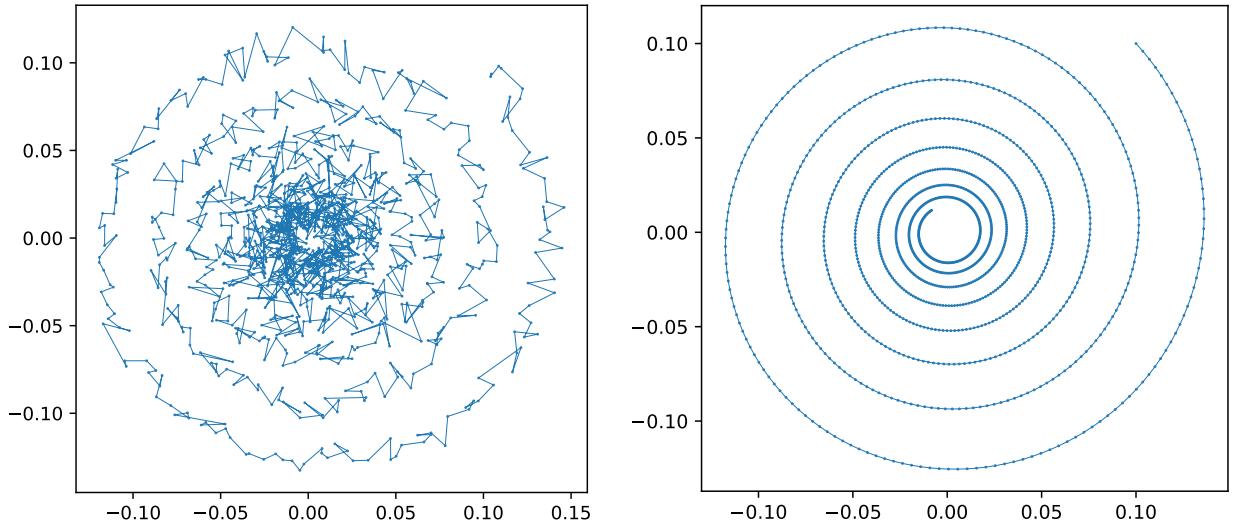


Рисунок 25 – Исходные данные (уровень шума 0.005) и график восстановленной системы уравнений (43)

$$\begin{aligned}\dot{x} &= -0.115x + 1.97y \\ \dot{y} &= -1.94x - 0.062y\end{aligned}\tag{43}$$

То с более сложными системами возникают проблемы. Это видно на рисунке 26, на котором изображена попытка подбора параметра — коэффициента регуляризации — для системы Лоренца. График показывает значения метрики IoU в зависимости от уровня шума и величины параметра, серые пунктирные линии ограничивают область с наибольшим IoU.

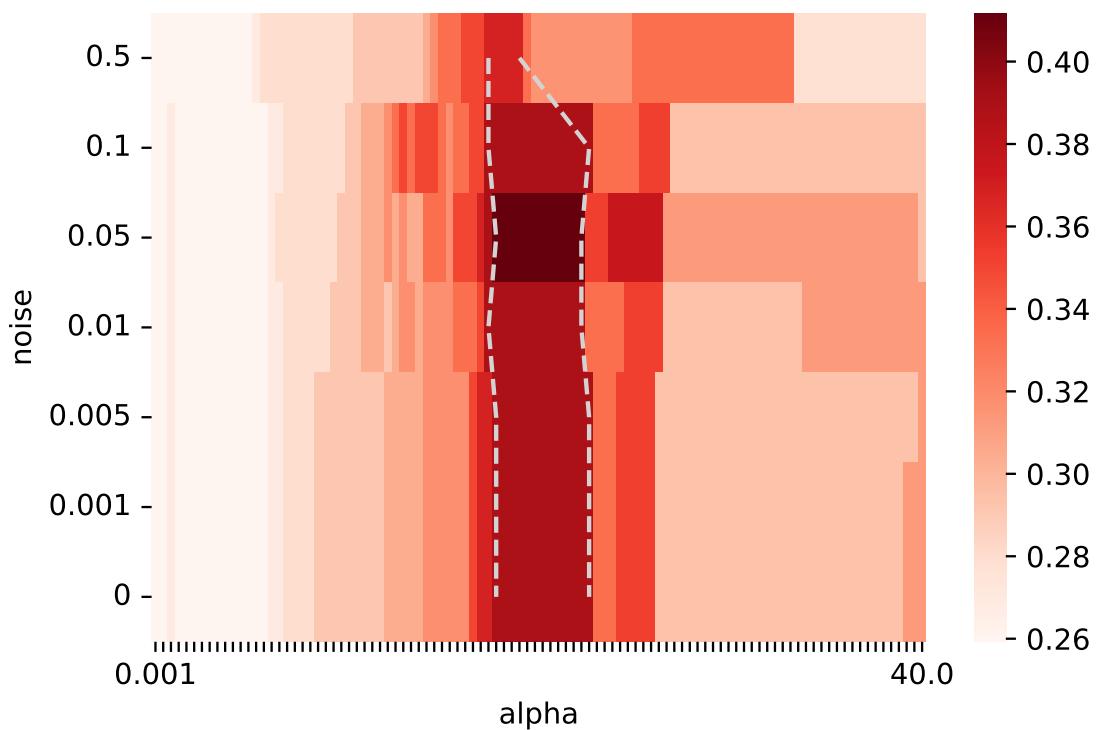


Рисунок 26 – Подбор параметра  $\alpha$  для системы Лоренца

Пример крайне неудачной идентификации показан на рисунке 27.

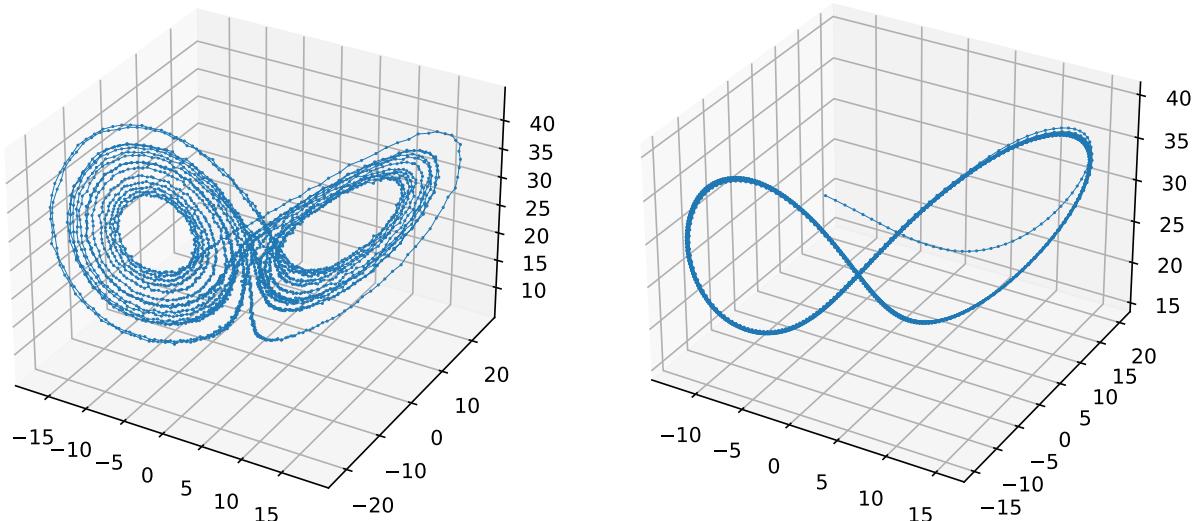


Рисунок 27 – Исходные данные (уровень шума 0.1) и график восстановленной системы уравнений (44)

$$\begin{aligned}\dot{x} &= 1.2y - 0.3xz + 5.1 \times 10^{-4}y^2 + 0.28yz + 0.0013z^2 \\ \dot{y} &= 0.47x + 12y + 0.0026x^2 - 0.26xz - 0.32yz - 0.004z^2 \\ \dot{z} &= -0.72z + 0.57x^2 + 0.57xy + 9 \times 10^{-5}xz + 0.035y^2 - 0.088z^2\end{aligned}\tag{44}$$

В связи с этим, далее алгоритм Lasso не используется, и в качестве метода регрессии рассматривается только STLSQ.

### 3.1.3.1 Система Лоренца

Систему линейных уравнений решено пропустить из-за ее излишней простоты. Поэтому переходим к аттрактору Лоренца — решению системы Лоренца (45) с начальными значениями  $x_0 = 8$ ,  $y_0 = 7$ ,  $z_0 = 27$  и в диапазоне  $t \in [0, 20]$ . Аттрактор изображен на рисунке 28.

$$\begin{aligned}\dot{x} &= 10(y - x) \\ \dot{y} &= x(28 - z) - y \\ \dot{z} &= xy - 8/3z\end{aligned}\tag{45}$$

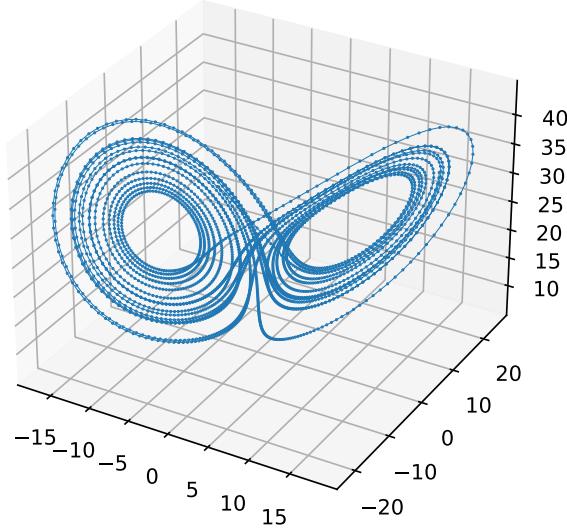


Рисунок 28 – Аттрактор Лоренца

Протестируем алгоритм STLSQ. Результаты тестирования изображены на рисунке 29. Здесь уже наблюдается диапазон значений параметра, при которых достигается приемлемый результат.

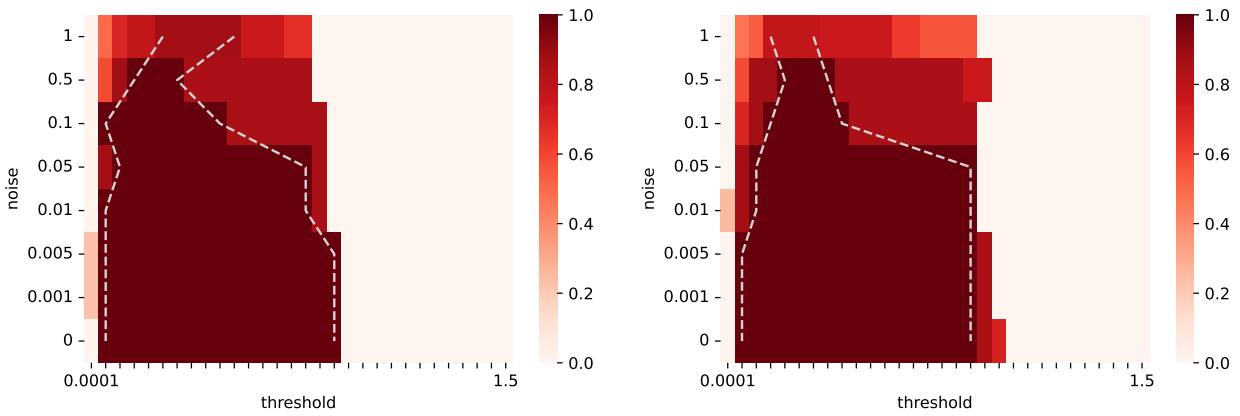


Рисунок 29 – Тестирование алгоритма STLSQ для обоих методов дифференцирования

Таким образом, имеется возможность подобрать параметры для идентификации системы. На рисунке 30 показаны результаты идентификации вместе со значениями метрики IoU, каждый столбец соответствует одному уровню шума, на первой строке показаны исходные данные.

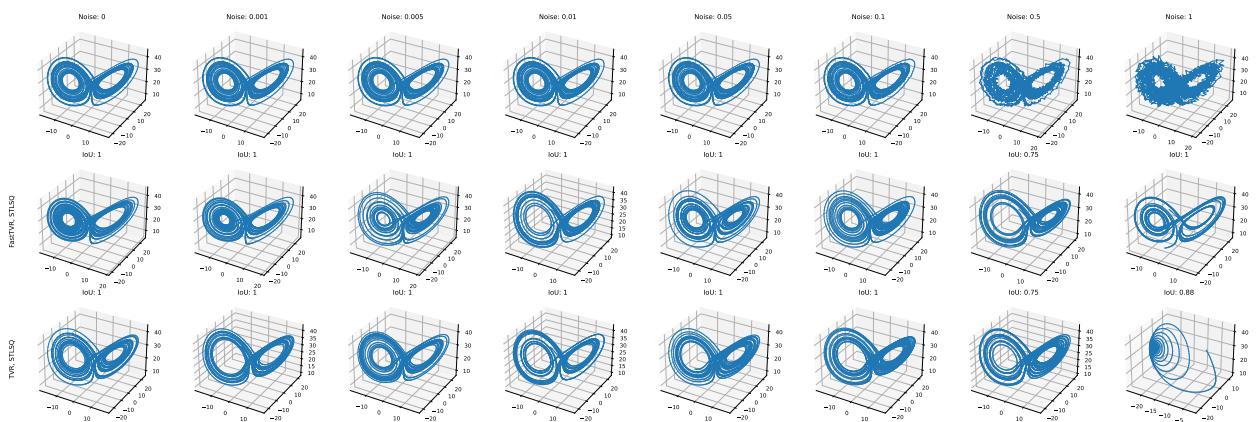


Рисунок 30 – Графики восстановленных систем Лоренца

Можно заметить, что восстановленные системы отличаются от оригинальной. Это связано с тем, что параметры системы восстанавливаются не очень точно. Восстановленные системы уравнений при использовании TVR и уровнях шума 0, 0.005, 0.05, 0.1, 0.5 и 1 имеют следующий вид:

$$\dot{x} = -10x + 10y$$

$$\dot{y} = 27.79x - 0.917y - 0.993xz$$

$$\dot{z} = -2.67z + 0.999xy$$

$$\dot{x} = -9.974x + 9.979y$$

$$\dot{y} = 27.78x - 0.919y - 0.993xz$$

$$\dot{z} = -2.667z + 0.999xy$$

$$\dot{x} = -9.97x + 9.978y$$

$$\dot{y} = 27.63x - 0.861y - 0.989xz$$

$$\dot{z} = -2.668z + 0.999xy$$

$$\dot{x} = -9.91x + 9.928y$$

$$\dot{y} = 27.34x - 0.833y - 0.982xz \quad (46)$$

$$\dot{z} = -2.647z + 0.989xy$$

$$\dot{x} = -9.605x + 9.683y$$

$$\dot{y} = 25.55x - 0.939xz$$

$$\dot{z} = 1.2 - 2.658z + 0.977xy$$

$$\dot{x} = 2.741 - 0.438x + 4.448y$$

$$\dot{y} = 23.03x + 0.327y - 0.877xz$$

$$\dot{z} = -2.541z + 0.955xy$$

На рисунке 31 показаны изменения метрик при использовании TVR: средней ошибки между значениями величин, метрик ME и IoU.

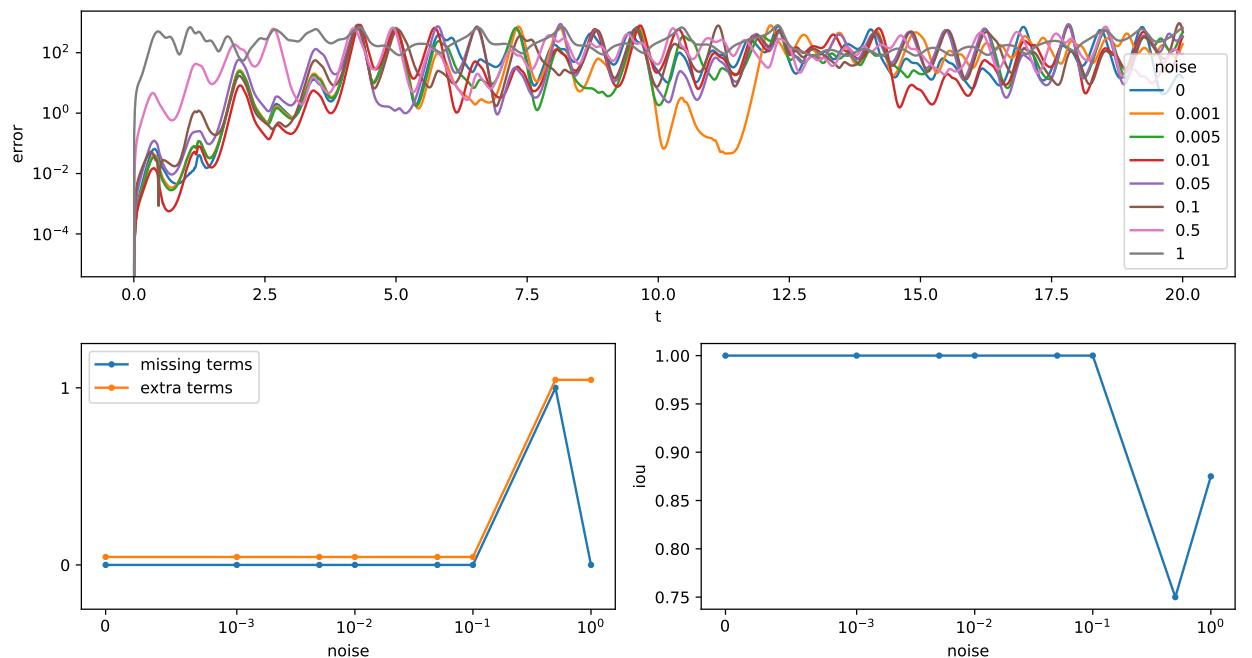


Рисунок 31 – Графики изменения метрик

Видно, что, из-за неточных параметров, сравнивать восстановленную систему с исходной напрямую не имеет смысла. Значения средней ошибки получаются большими и примерно одинаковыми для всех случаев. Остальные графики более осмысленные, видно, что число лишних или потерянных

слагаемых растет очень незначительно.

### 3.1.3.2 Система Мура-Шпигеля

Система Мура-Шпигеля (47) взята с начальными значениями  $x_0 = 0.1$ ,  $y_0 = 0.2$ ,  $z_0 = 0.3$  и в диапазоне  $t \in [600, 900]$ . Решение системы изображено на рисунке 32.

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= z - 0.2x \\ \dot{z} &= -z/50 + y(1 - 0.8x^2)\end{aligned}\tag{47}$$

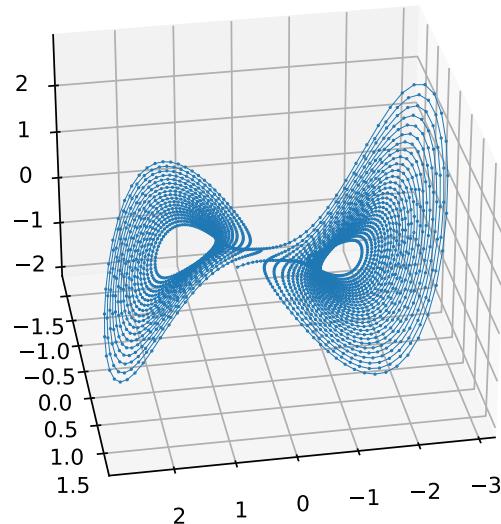


Рисунок 32 – Система Мура-Шпигеля

Эту систему было значительно сложнее идентифицировать, как можно видеть из рисунка 33.

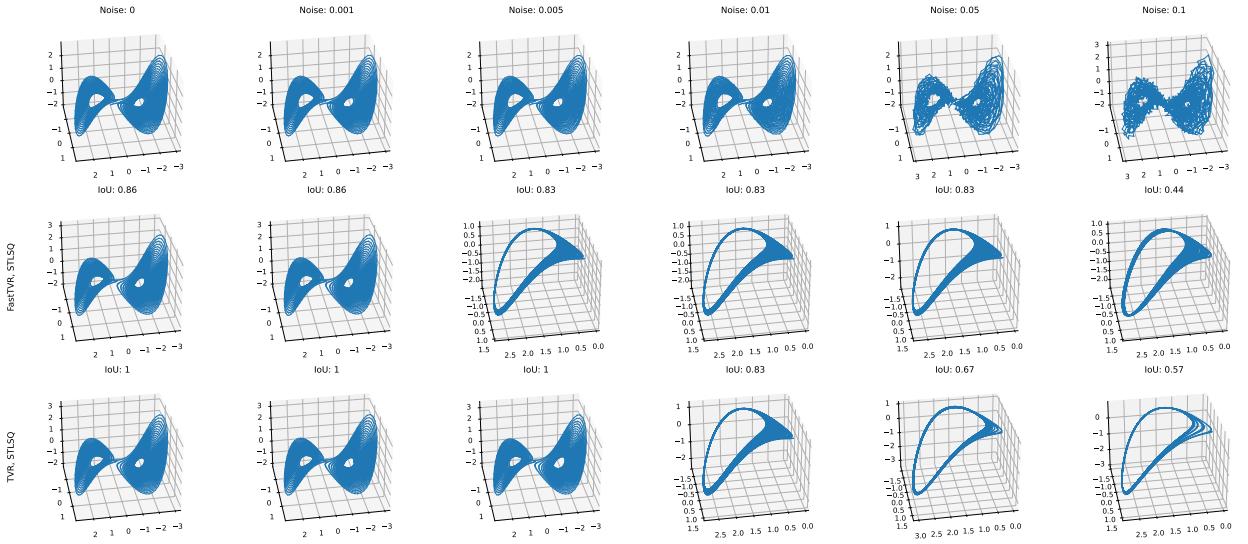


Рисунок 33 – Графики восстановленных систем  
Мура-Шпигеля

Графики изменения метрик для обоих методов дифференцирования показаны на рисунке 34.

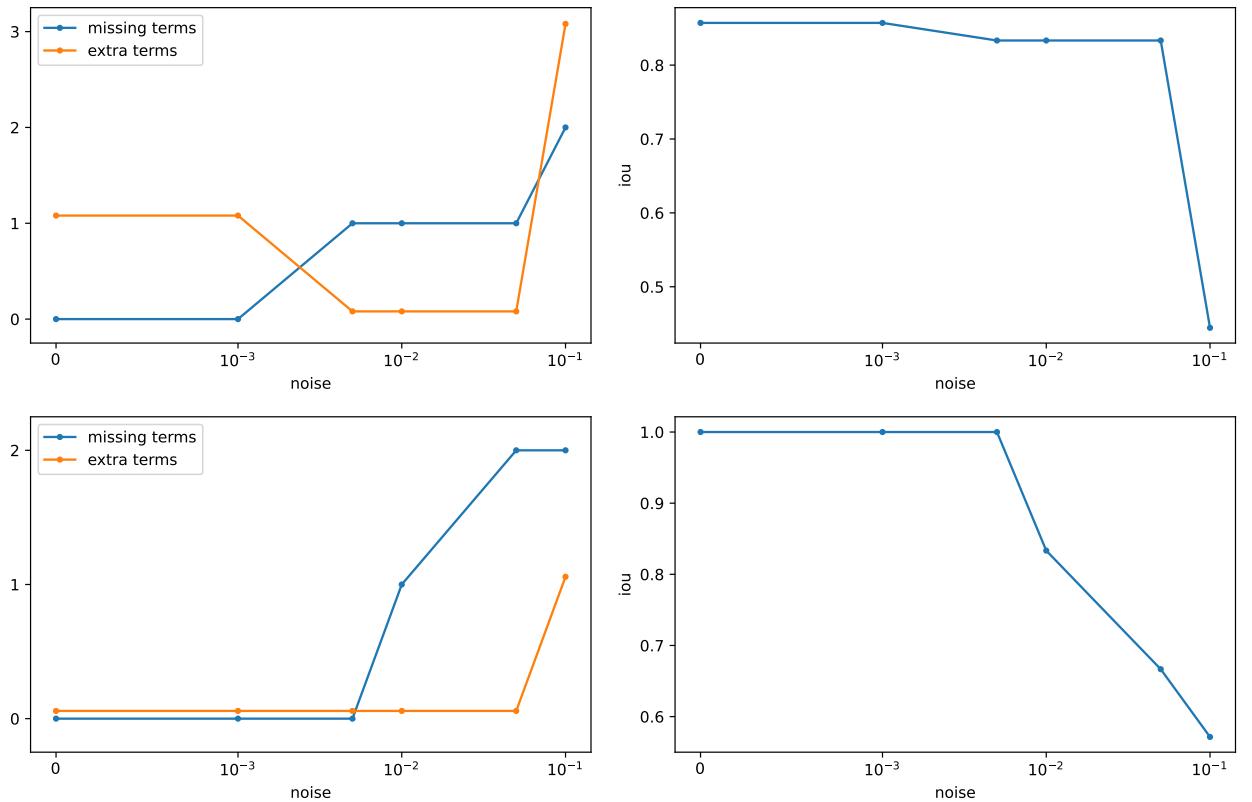


Рисунок 34 – Графики изменения метрик

Ниже приведены восстановленные системы уравнений при использовании TVR, видно, как, начиная с четвертого уровня шума,

уравнения теряют слагаемые:

$$\begin{aligned}
 \dot{x} &= 0.977y & \dot{x} &= 0.977y \\
 \dot{y} &= -0.186x + 0.975z & \dot{y} &= -0.186x + 0.976z \\
 \dot{z} &= 0.992y - 0.021z - 0.798x^2y & \dot{z} &= 0.993y - 0.021z - 0.799x^2y \\
 \\ 
 \dot{x} &= 0.984y & \dot{x} &= 0.982y \\
 \dot{y} &= -0.185x + 0.978z & \dot{y} &= -0.187x + 0.979z \\
 \dot{z} &= 1.006y - 0.021z - 0.802x^2y & \dot{z} &= 1.046y - 0.81x^2y \\
 \\ 
 \dot{x} &= 0.963y & \dot{x} &= 0.905y \\
 \dot{y} &= 0.948z & \dot{y} &= 0.917z \\
 \dot{z} &= 0.9y - 0.729x^2y & \dot{z} &= 0.757y - 0.642x^2y - 0.167yz^2
 \end{aligned} \tag{48}$$

### 3.2 Перспективы дальнейшего развития алгоритма

В данной работе задача идентификации ограничивается только системами ОДУ 1-ого порядка. Однако возможности алгоритма позволяют делать гораздо больше этого.

Рассмотрим описанный базовый алгоритм. Имея методы обычного численного дифференцирования первого порядка, можно довольно легко получить методы дифференцирования высоких порядков, равно как и методы получения частных производных. Это позволяет производить идентификацию дифференциальных уравнений в частных производных [3], что довольно сильно расширяет область задач, к которым применим данный алгоритм. Также можно отказаться от производных вообще и идентифицировать системы рекуррентных соотношений [3] вида:

$$x_{k+1} = f(x_k). \tag{49}$$

Кроме этого, можно расширять сам алгоритм идентификации для решения других классов задач. Например, задач теории управления, в частности управления с прогнозирующими моделями (model predictive control) [19; 20]. Такая модификация позволяет идентифицировать системы, в

случае ОДУ, вида:

$$\dot{x} = f(x, u), \quad (50)$$

где  $u$  — управляющий сигнал.

Или, вместо задачи Коши, идентифицировать системы уравнений краевой задачи [21]. Или все ту же задачу Коши, но для уравнений в неявном виде [22] (для таких задач процесс разреженной регрессии еще и крайне хорошо распараллеливается):

$$f(x, \dot{x}) = 0. \quad (51)$$

Однако также можно улучшать сам алгоритм. Например, использовать идею ансамблей и обучать несколько моделей вместо одной [23]. При этом каждая из моделей обучается на подмножестве исходных данных (бутстрэппинг) и идентифицированные системы агрегируются в одну итоговую.

Что касается отдельных частей алгоритма, помимо использования специализированных методов дифференцирования можно пытаться фильтровать сами данные, например, при помощи нейросетевого подхода [24], который отделяет шумовую компоненту, что позволяет работать с шумом любого распределения и даже это распределение идентифицировать.

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения выпускной квалификационной работы бакалавра был реализован алгоритм идентификации систем ОДУ на основе данных. Было произведено тестирование алгоритма на известных системах, по результатам которого можно заключить о его работоспособности. Отдельно от алгоритма были также протестированы и проанализированы его составные части: алгоритмы дифференцирования и разреженной регрессии.

Большую часть работы заняло численное дифференцирование. Применительно к задаче идентификации методы регуляризации полной вариации ранее не применялись, поэтому реализация, анализ, подбор параметров осуществлялись полностью с нуля. С этой точки зрения, использование этих методов можно считать некоторым развитием данной области.

В ходе анализа методов дифференцирования была обнаружена зависимость между уровнем шума и параметром метода. Поэтому для эффективного противодействия шуму в данных, необходимо оценивать его уровень. Этот вопрос не рассматривался в данной работе, однако, для этого существуют различные методы, например, с использованием спектрального анализа.

Алгоритм разреженной регрессии Lasso, сначала показался вполне работоспособным, однако, при тестировании для задачи идентификации работоспособность не подтвердилась. Таким образом, этот алгоритм отброшен.

Тестирование алгоритма идентификации показало сложность данной задачи, отдельно подсветив тот факт, что алгоритмы регрессии и дифференцирования, хорошо работающие в своей области, не достаточно просто объединить, и такие проблемы, как подбор параметров, приходится решать снова, учитывая специфику задачи. Однако с другой стороны, полученные результаты вселяют уверенность в перспективность данного метода и возможность его успешного использования в практических задачах.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bongard J., Lipson H. Automated reverse engineering of nonlinear dynamical systems // Proceedings of the National Academy of Sciences of the United States of America. — 2007. — Т. 104.
2. Koza J. Genetic Programming: On the Programming of Computers by Means of Natural Selection. T. 1. — Bradford, 1992.
3. Brunton S. L., Proctor J. L., Kutz J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems // Proceedings of the National Academy of Sciences. — 2016. — Т. 113, № 15. — С. 3932—3937.
4. Tibshirani R. Regression Shrinkage and Selection via the Lasso // Journal of the Royal Statistical Society. Series B (Methodological). — 1996. — Т. 58, № 1. — С. 267—288.
5. Schmid P. J. Dynamic mode decomposition of numerical and experimental data // Journal of Fluid Mechanics. — 2010. — Т. 656. — С. 5—28.
6. Mezic I. Analysis of Fluid Flows via Spectral Properties of the Koopman Operator // Annual Review of Fluid Mechanics. — 2013. — Т. 45. — С. 357—378.
7. Тихонов А. Н. О регуляризации некорректно поставленных задач // Докл. АН СССР. — 1963. — Т. 153, вып. 1. — С. 49—52.
8. Chartrand R. Numerical Differentiation of Noisy, Nonsmooth Data // International Scholarly Research Notices. — 2011. — Т. 2011. — С. 1—11.
9. Vogel C. R., Oman M. E. Iterative Methods for Total Variation Denoising // SIAM J. Sci. Comput. — 1996. — Т. 17, № 1. — С. 227—238.
10. Vogel C. Computational Methods for Inverse Problems. — Society for Industrial, Applied Mathematics, 2002.
11. Python — The programming language. — URL: <https://www.python.org/> (дата обращения 28.04.2023).
12. NumPy — The fundamental package for scientific computing with Python. — URL: <https://numpy.org/> (дата обращения 28.04.2023).
13. SciPy — Fundamental algorithms for scientific computing in Python. — URL: <https://scipy.org/> (дата обращения 28.04.2023).

14. Scikit-learn — Machine Learning in Python. — URL: <https://scikit-learn.org/> (дата обращения 28.04.2023).
15. Jupyter — Python notebook interface. — URL: <https://jupyter.org/> (дата обращения 28.04.2023).
16. Matplotlib — Visualization with Python. — URL: <https://matplotlib.org/> (дата обращения 28.04.2023).
17. Seaborn — Statistical data visualization with Python. — URL: <https://seaborn.pydata.org/> (дата обращения 28.04.2023).
18. Павлов Б. М., Новиков М. Д. Автоматизированный практикум по нелинейной динамике (синергетике) : Учебное пособие. — М. : Издательский отдел факультета ВМК МГУ, 2000. — 115 с.
19. Brunton S. L., Proctor J. L., Kutz J. N. Sparse Identification of Nonlinear Dynamics with Control (SINDYc) // IFAC-PapersOnLine. — 2016. — Т. 49, № 18. — С. 710—715.
20. SINDy with Control: A Tutorial / U. Fasel [и др.] // 2021 60th IEEE Conference on Decision and Control (CDC). — 2021. — С. 16—21.
21. Shea D. E., Brunton S. L., Kutz J. N. SINDy-BVP: Sparse identification of nonlinear dynamics for boundary value problems // Physical Review Research. — 2021. — Т. 3, вып. 2.
22. Kaheman K., Kutz J. N., Brunton S. L. SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics // Proceedings of the Royal Society A. Т. 476. — 2020.
23. Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control / U. Fasel [и др.] // Proceedings of the Royal Society A. Т. 478. — 2022.
24. Kaheman K., Brunton S. L., Kutz J. N. Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data // Machine Learning: Science and Technology. — 2022. — Т. 3, № 1. — С. 15—31.

## **ПРИЛОЖЕНИЕ А**

### **Исходный код**

Репозиторий с исходным кодом расположен по адресу <https://github.com/iktovr/bachelor-diploma>. Ссылка в формате QR-кода представлена на рисунке А.1.



Рисунок А.1 – QR-код со ссылкой на репозиторий