Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине: «Вычислительная математика»

Студент	Фуров Павел Павлович
Группа	РК6-61Б
Тип задания	Лабораторная работа №2
Тема лабораторной работы	Регрессия и анализ временных рядов

Студент _____ **Фуров П.П.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Першин А.Ю.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2021 г.

Оглавление

Задание на лабораторную работу

Требуется (базовая часть):

1. Написать функцию *poly_regression(x_nodes, y_nodes, degree, l)*, которая возвращает коэффициенты многочлена степени *degree*, наилучшим образом приближающегося к точкам с абсциссами *x_nodes* и ординатами, используя L_2-регуляризацию со значением гиперпараметра. Коэффициенты должны вычисляться с помощью подходящего нормального уравнения.
2. Сформировать выборку данных зависимости усредненной месячной концентрации атмосферного углекислого газа от времени на основе измерений в обсерватории Мауна-Лоа. Данные можно найти на сайте NOOA: *https://www.esrl.noaa.gov/gmd/aftp/data/trace_gases/co2/flask/surface/co2_mlo_surface-flask_1_ccgg_month.txt*. Полученный набор данных мы будем обозначать D. Из данных следует исключить начальный период, где значения концентрации заметно ниже тренда.
3. Разбить набор данных D на два набора одинакового размера: D_{train}, который будет использоваться для решения нормального уравнения, и D_{valid}, который будет использоваться для поиска оптимального значения гиперпараметра. Разбиение следует произвести случайным образом.
4. Для каждого p из множества $\{1,2,3,4,5,10,20\}$ и случая отсутствия L_2-регуляризации провести следующий анализ:
    - С помощью набора данных D_{train} и функции $poly\_regression$ построить многочлен степени p, наилучшим образом приближающийся к данным.
    - Вычислить среднеквадратичные погрешности $\epsilon_{train}^{(p)}$ и $\epsilon_{valid}^{(p)}$ аппроксимации полученным многочленом наборов данных D_{train} и D_{valid} соответственно.
5. Вывести на отдельных графиках полученные многочлены вместе с данными, использованными для решения нормального уравнения, и добавить таблицу полученных среднеквадратичных погрешностей. Сделать вывод о наиболее подходящей степени p, используя зависимости $\epsilon_{train}^{(p)}$ и $\epsilon_{valid}^{(p)}$ от p.

6. Для $p = 20$ найти оптимальные значения коэффициентов полинома, используя L_2-регуляризацию и график зависимости $\epsilon_{valid}^{(20)}$ от значений гиперпараметра. Сделать вывод об использовании L_2-регуляризации.

Требуется (продвинутая часть):

1. Привести временной ряд, определенный данными D, к стацинарному виду, исключив из него тренд. Продемонстрировать стационарный временной ряд на графике.
2. Используя алгоритм Кули–Тьюки, написать функцию *fft_coeff(y_nodes)*, которая вычисляет и возвращает комплексные коэффициенты тригонометрического полинома, интерполирующего узлы y_nodes.
3. Определить периоды сезонности данного временного ряда, идентифицировав его характерные частоты с помощью функции *fft_coeff(y_nodes)* и выведя связанные с ними амплитуды коэффициентов тригонометрического ряда на графике.

Цель выполнения лабораторной работы

Классическая задача регрессии заключается в предсказании значений данного временного ряда. Для этого в первую очередь необходимо построить кривую, оптимально приближающуюся к исходным данным. С точки вычислительной математики необходимо решить задачу оптимизации, полученную с помощью метода наименьших квадратов.

Цели базовой части:

- Рассмотреть и решить задачу полиномиальной регрессии, полученную с помощью метода наименьших квадратов на примере построения многочлена, наилучшим образом приближающегося к данным о зависимости усреднённой месячной концентрации углекислого газа от времени.
- Изучить подход разбиения данных, позволяющий предотвращать переобучение и находить оптимальное значение гиперпараметра.
- Определить погрешность аппроксимации и её поведение в зависимости от степени аппроксимирующего полинома.
- Используя L_2 -регуляризацию и зависимость среднеквадратичной погрешности от значения гиперпараметра, найти оптимальные значения коэффициентов полинома и сделать соответствующие выводы.

Цели продвинутой части:

- Продемонстрировать результат применения алгоритма Кули-Тьюки при интерполировании тригонометрическими полиномами
- Определить период сезонности временного ряда и продемонстрировать корректность быстрого преобразования Фурье.

Ход выполнения лабораторной работы

1. Базовая часть

1.1. Разработка функции poly_regression

Рассмотрим аппроксимирующую функцию $f: \mathbb{R}^{\boldsymbol{n}} \to \mathbb{R}$**:**

$$f(\boldsymbol{a}, \boldsymbol{x}) = \sum_{i=1}^{n} a_i \phi_i(\boldsymbol{x}), \tag{1}$$

где матрица $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{l}}, \phi_0(\boldsymbol{x}) = 1$ и $\phi_i(\boldsymbol{x}), i = 1, \ldots, n$ – базисные функции

Нелинейная регрессия с помощью метода наименьших квадратов сводится к задаче минимизации:

$$\min_{\boldsymbol{a}} E(\boldsymbol{a}) = \min_{\boldsymbol{a}} [(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a})^T (y - \boldsymbol{X}\boldsymbol{a})], \tag{2}$$

где матрица $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{m} \times (\boldsymbol{n}+\boldsymbol{1})}$ определена как:

$$\boldsymbol{X} = \begin{bmatrix} 1 & \phi_1(\boldsymbol{x}^{(1)}) & \ldots & \phi_n(\boldsymbol{x}^{(1)}) \\ 1 & \phi_1(\boldsymbol{x}^{(2)}) & \ldots & \phi_n(\boldsymbol{x}^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \phi_1(\boldsymbol{x}^{(m)}) & \ldots & \phi_n(\boldsymbol{x}^{(m)}) \end{bmatrix}. \tag{3}$$

Требуется же написать функцию, реализующую полиномиальную регрессию, которая является частным случаем нелинейной регрессии. В таком случае выражения (1) и (3) принимают вид:

$$f(\boldsymbol{a}, \boldsymbol{x}) = \sum_{i=1}^{n} a_i x^i,$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{(1)} & \ldots & x_{(1)}^n \\ 1 & x_{(2)} & \ldots & x_{(2)}^n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{(m)} & \ldots & x_{(m)}^n \end{bmatrix}.$$

Исходя из того факта, что $\boldsymbol{X}$ является матрицей Вандермонда, чей определитель отличен от нуля при уникальных дискретных данных, нормальное уравнение всегда будет иметь единственное и нетривиальное решение:

$$\boldsymbol{a} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{4}$$

L_2 −регуляризация является модификацией задачи МНК:

$$\min_{\boldsymbol{a}} E(\boldsymbol{a}) = \min_{\boldsymbol{a}}[(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a})^T(y - \boldsymbol{X}\boldsymbol{a}) + \lambda\|\boldsymbol{a}\|_2^2],$$

где λ – значение гиперпараметра.

Аналитическое решение находится через модификацию нормального уравнения (4):

$$\boldsymbol{a} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{E})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \quad (5)$$

где размерность $\boldsymbol{E}$ – $(n+1) \times (n+1)$, n – степень аппроксимирующего полинома.

В листинге 1 представлена реализация формулы (5), где *x_nodes, y_nodes* – координаты точек, к которым происходит приближение, *degree* – степень полинома, *l* – значение гиперпараметра.

Листинг 1 – функция diff1, возвращающая оптимальные коэффициенты аппроксимирующего полинома

```
def poly_regression(x_nodes, y_nodes, degree, l):
    X = np.vander(x_nodes, N=degree+1, increasing=True)
    return np.linalg.inv(X.T @ X+l * np.identity(degree+1)) @ X.T @ y_nodes
```

1.2. Формирование выборки данных

Текстовый файл с данными, взятыми по ссылке из задания, был обработан встроенной в nympy функцией loadtxt. Месяц и год в этом файле содержатся в разных колонках, которые в совокупности отражают время измерения. Для того, чтобы можно было работать с временем, как с осью абсцисс, время будет исчисляться в «годах», например, 7 месяц 2000 года будет представлен как 2000,5.

Листинг 2 – Считывание данных и их преобразования к виду «ось абсцисс» - «ось ординат»

```
data = np.loadtxt("data.txt", usecols=(1,2,3))
x_full = (data[:,0]+(data[:,1]-1)/12.0)
y_full = data[:,2]
```

Визуализация полученных данных изображена на рисунке 1.

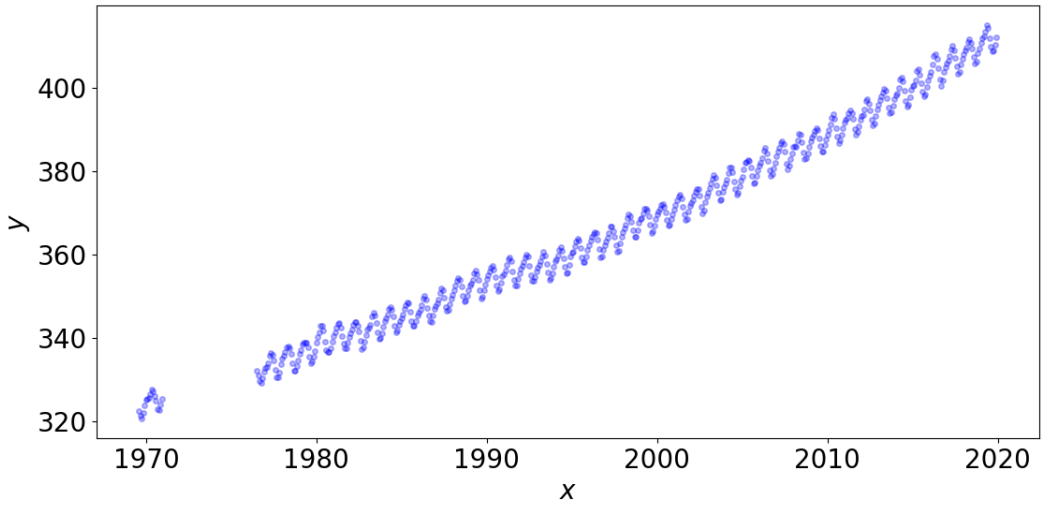

Рисунок 1 – график зависимости концентрации атмосферного углекислого газа от времени.

Однако, как видно на рисунке 1, в период по 1976 год, значение концентрации заметно ниже (что упоминалось в задании), поэтому этот период был исключён:

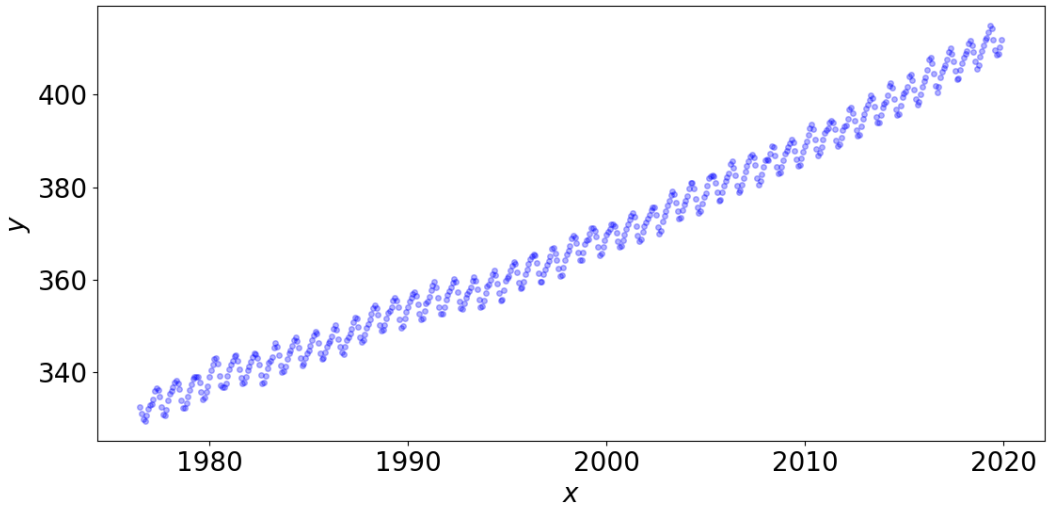

Рисунок 2 – график зависимости концентрации атмосферного углекислого газа от времени с 1976 года.

1.3. Разбиение данных

Полученный набор данных D был случайным образом разбит на два набора одинакового размера: D_{train}, который будет использоваться для решения нормального уравнения, и D_{valid}, который будет использоваться для поиска оптимального значения гиперпараметра:

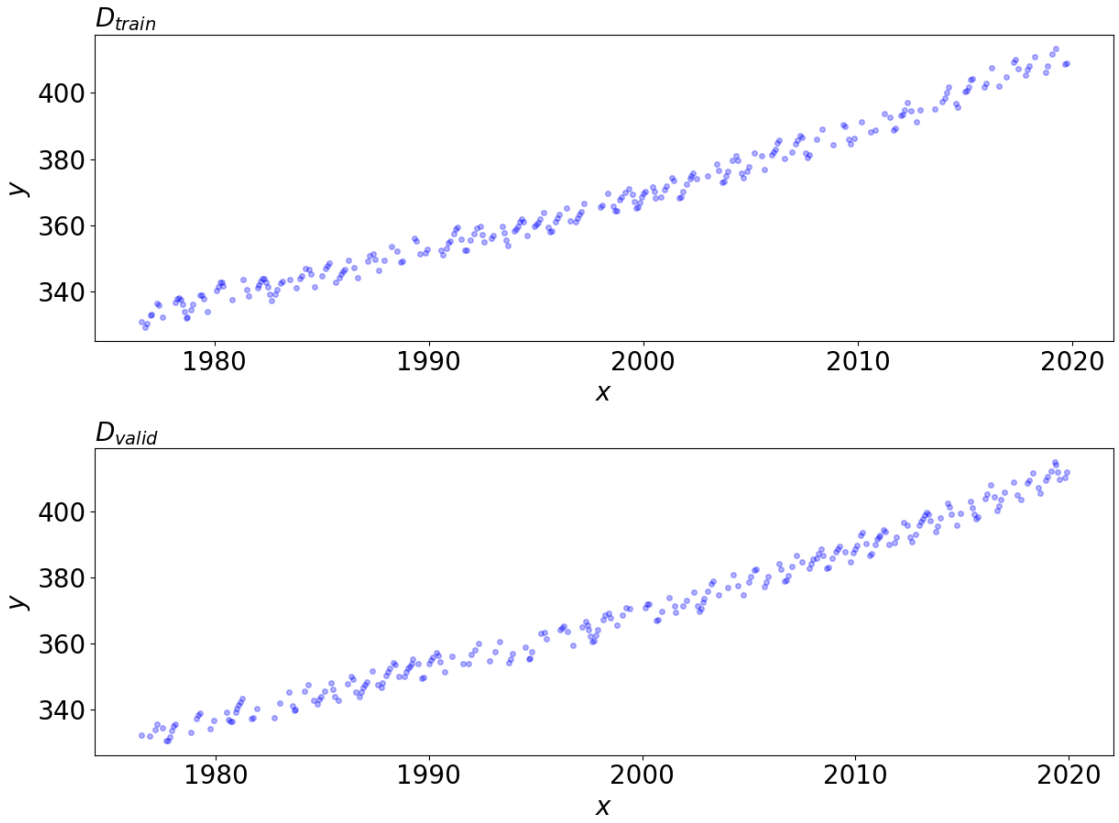

Рисунок 3 – наборы данных для решения нормального уравнения и определения оптимального гиперпараметра соответственно.

1.4. Построение полиномов

С помощью набора данных D_{train} и функции poly_regression, были построены многочлены степеней 1, 2, 3, 4, 5, 10, 20, с точки зрения среднеквадратичной погрешности наилучшим образом приближающиеся к данным:

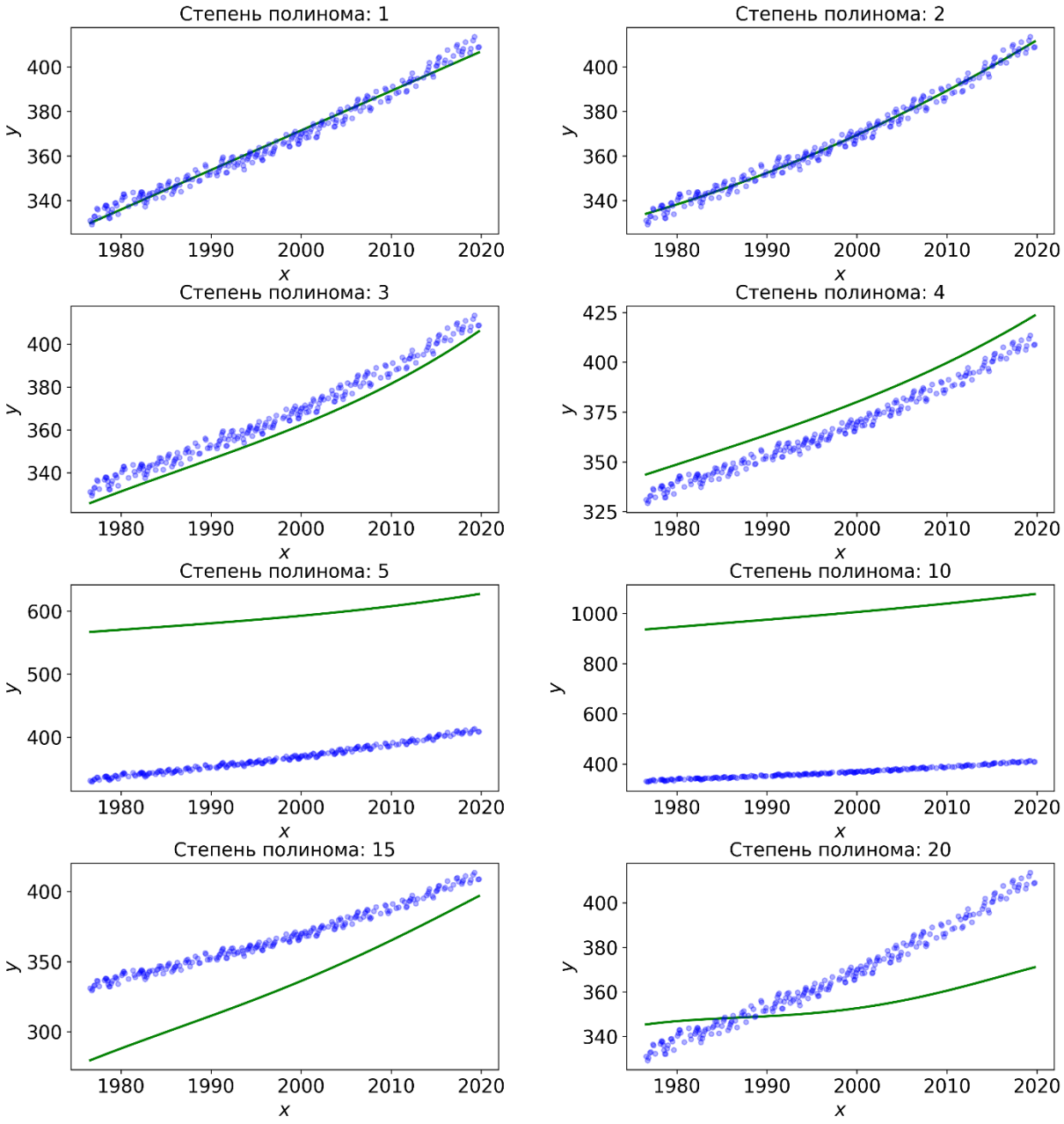

Рисунок 4 – визуализация полиномиальных регрессий (зелёный цвет), на основе данных из набора D_{train} (синий цвет).

Однако полученные данные с точки зрения анализа использования полиномиальной регрессии бесполезны: присутствует вычислительная неустойчивость ввиду наличия значений огромного порядка в матрице Вандермонда. Поэтому следует провести нормализацию значений абсцисс от -1 до 1. Графики, основанные на многочленах, коэффициенты которых были вычислены для нормализованных значений x_train:

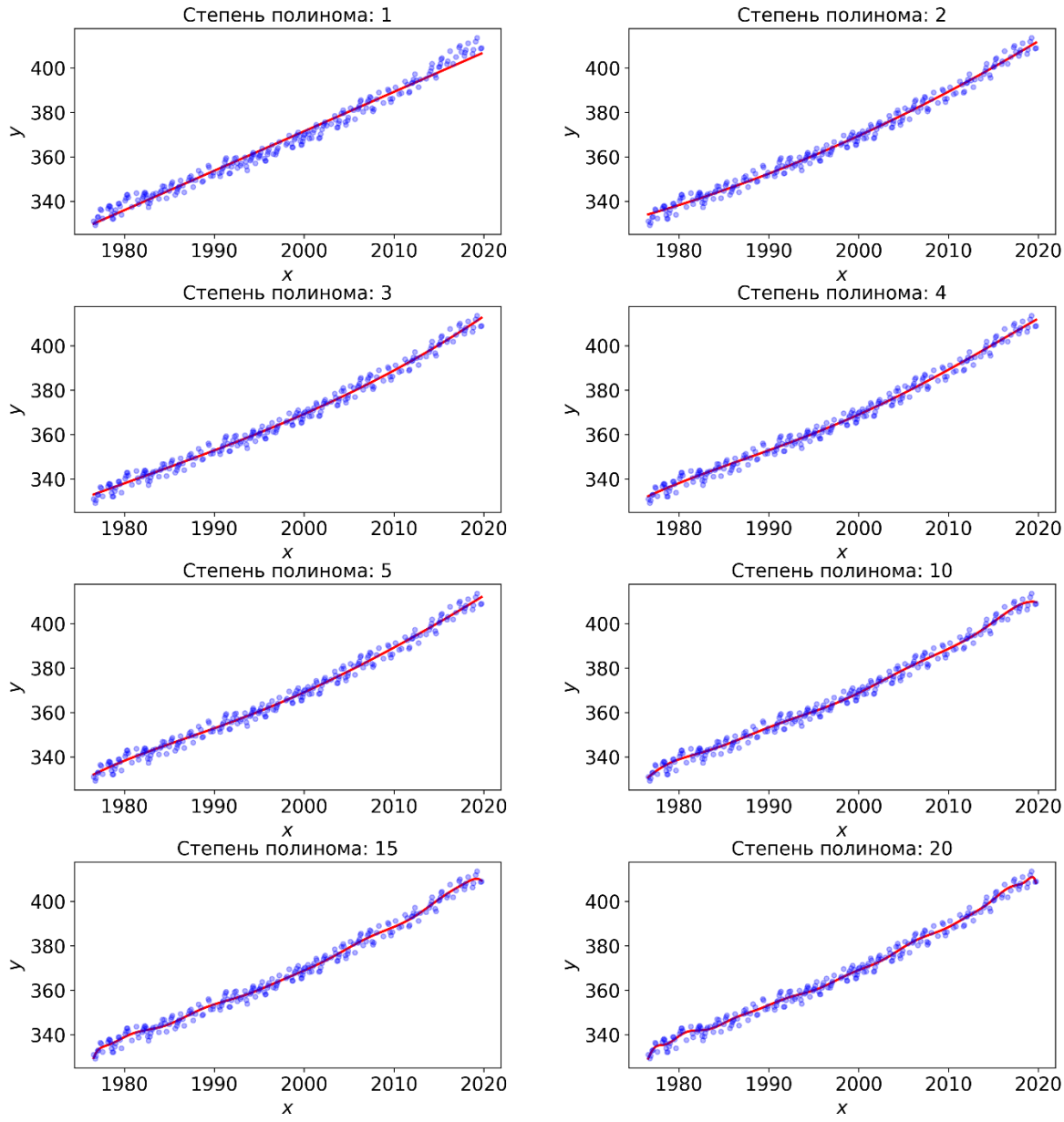

Рисунок 5 – визуализация полиномиальных регрессий (красный цвет), на основе данных из набора D_{train} (синий цвет) с использованием нормализации.

Полученные данные отображают реальные виды многочленов, полученные при помощи метода наименьших квадратов. Видно, что графики большей степени содержат больше осцилляций, однако ввиду линейности тренда это не так ярко выражено визуально, так как, по сути, каждый из многочленов просто пытается вытянуться вдоль набора данных.

С помощью библиотеки python-docx была сформирована таблица среднеквадратичных погрешностей. Данные при отсутствии нормализации были отброшены ввиду большой вычислительной погрешности, а значит и бесполезности этих данных в рамках изучения полиномиальной регрессии.

Таблица 1 – среднеквадратичные погрешности

Степень полинома	ϵ_{train}	ϵ_{valid}
1	9.344	9.598
2	5.323	5.824
3	5.156	5.645
4	5.075	5.662
5	5.069	5.669
10	4.808	5.904
15	4.717	5.859
20	4.624	6.078

Из таблицы 1 видно, что погрешность аппроксимации D_{train} уменьшается с увеличением степени полинома, однако начиная с «оптимальной» степени, начинается увеличивать погрешность аппроксимации D_{valid}, что обусловлено переобучением модели – чем выше степень полинома, тем выше возможность у полинома уменьшить погрешность за счёт не следования тренду, а следования за конкретными точками. Поэтому не всегда минимальная квадратичная погрешность означает высокое качество приближения. Именно поэтому существует разбиение данных, которое и позволяет найти оптимальную степень многочлена, а далее и оптимальное значение гиперпараметра. Исходя из данных таблицы, многочлен 3-ей степени является оптимальным.

1.5. Применение L_2-регуляризации

Для минимизации погрешности ϵ_{valid} при регрессии, основанной на данных D_{train} используется L_2-регуляризация, призванная найти баланс между минимальной погрешностью и величиной коэффициентов многочлена, так как большие коэффициенты и порождают осцилляции. Был построен график зависимости $\epsilon_{valid}^{(20)}$ от значения гиперпараметра:

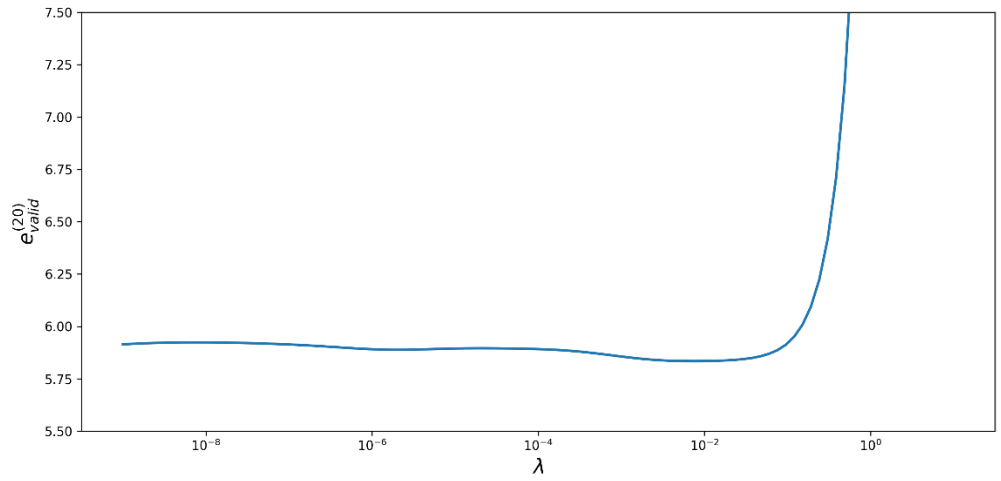

Рисунок 6 – график зависимости $\epsilon^{(20)}_{valid}$ от значения гиперпараметра.

Значение λ, соответствующее минимальной погрешности: 0.007390722

Минимальная погрешность: 5.834, в то время как при отсутствии регуляризации она равна 6.078.

Сравнение графиков аппроксимации с использованием L_2-регуляризации и без:

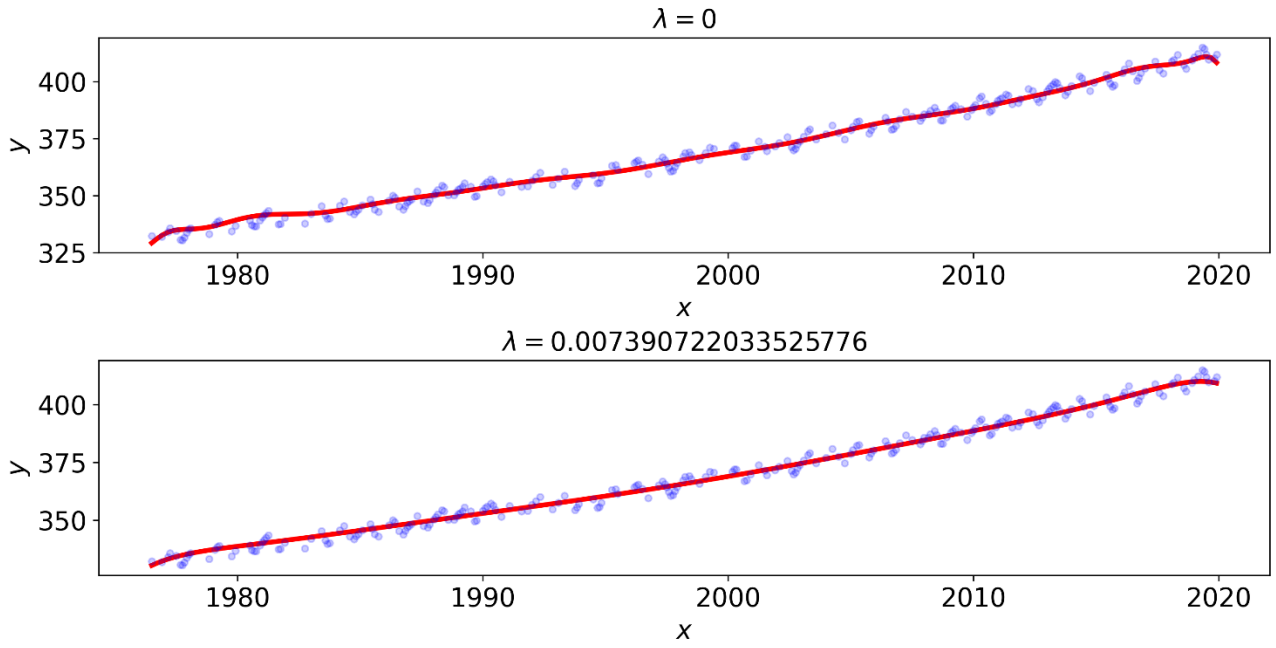

Рисунок 7 – графики полученных многочленов для различных значений гиперпараметра (красный), на фоне набора данных D_{valid} (синий)

Из графиков, изображённых на рисунке 7, можно сделать вывод, что применение L_2-регуляризации сглаживает аппроксимирующий полином за счёт уменьшения модулей его коэффициентов. Особенно это заметно на границах отрезка. Можно сделать вывод, что логично использовать L_2-регуляризацию всегда. Несмотря на то, что на этом примере её использование повлияло не так сильно, мы об этом и знали изначально ввиду того, что гиперпараметр получился близким к нулю. А это значит, что хуже она точно не делает, а во многих ситуациях, когда паразитные осцилляции более ярко выражены, позволяет существенно уменьшить погрешность аппроксимации тестовой выборки (что более важно).

2. Продвинутая часть

2.1. Приведение к стационарному виду

Привести временной ряд к стационарному виду можно одним из двух способов: вычитанием тренда, либо взятием разности с помощью функции numpy.diff. Полученные графики:

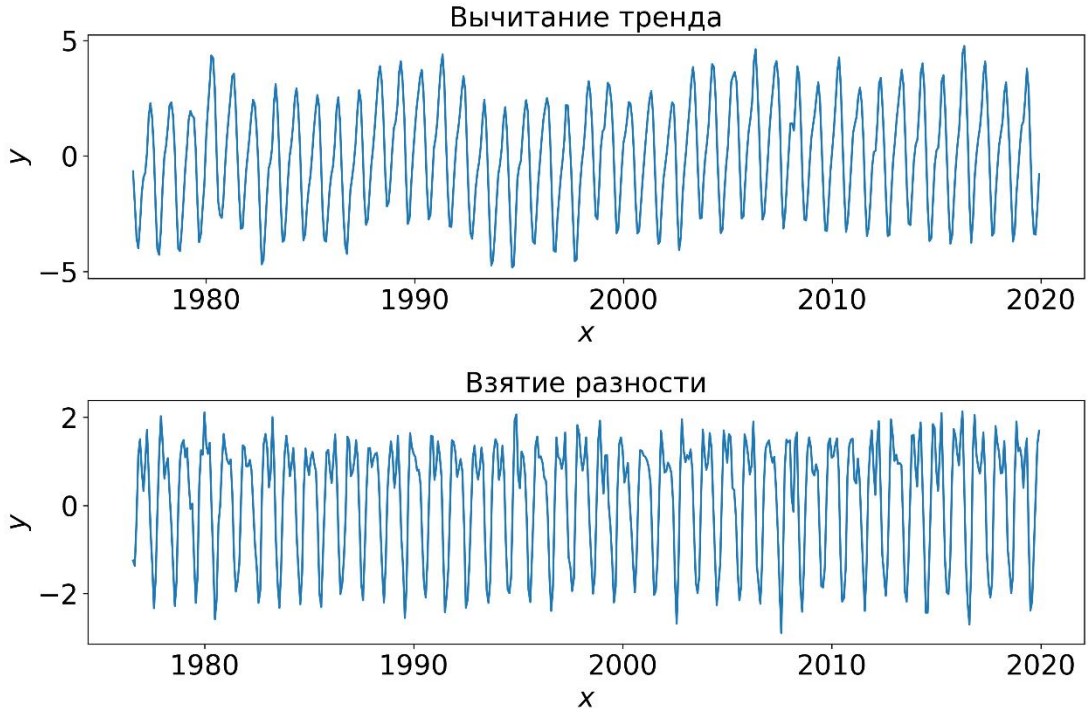

Рисунок 7 – графики временного ряда, приведённого к стационарному виду двумя различными способами.

Результат второго способа выглядит менее похожим исходные данные ввиду наличия непонятных скачков, которых на оригинальном графике нет. Поэтому для дальнейшего анализа будет использоваться стационарный вид временного ряда, полученный вычитанием тренда.

2.2. Разработка функции fft_coeff

Экспоненциальная форма тригонометрического полинома имеет следующий вид:

$$f(x) = \sum_{k=-m}^{m-1} \hat{a}_k e^{ikx} \tag{5}$$

где $2m$ – количество дискретных узлов.

Выражение для коэффициентов $\hat{a}_k$, составляющее дискретное преобразование Фурье:

$$\hat{a}_k = \frac{1}{2m} \sum_{j=0}^{2m-1} y_j e^{-ikx_j} \tag{6}$$

Дискретное преобразование Фурье в представленной форме требует $2m$ комплексных умножений и $2m-1$ комплексных сложений для вычисления одного коэффициента $\hat{a}_k$. Так как всего имеется $2m$ коэффициентов, алгоритмическая сложность дискретного преобразования Фурье имеет форму $O(m^2)$. Однако существует алгоритм быстрого преобразования Фурье, имеющий сложность $O(m\log_2 m)$. Одной из форм его реализации является алгоритм Кули-Тьюки.

$$\hat{a}_k = \frac{1}{2m} \sum_{j=0}^{2m-1} y_j e^{ik\pi} e^{\frac{-ikj\pi}{m}} = \frac{(-1)^k}{2m} \sum_{j=0}^{2m-1} y_j e^{\frac{-ikj\pi}{m}} \tag{7}$$

Сам алгоритм вычисляет значение суммы, обозначенной как:

$$A_k = \sum_{j=0}^{2m-1} y_j e^{\frac{-ikj\pi}{m}}, \qquad k = 0,\ldots,2m-1, \tag{8}$$

A_k раскладывается на 2 суммы с чётными и нечётными индексами:

$$A_k = \sum_{j=0}^{m-1} y_{2j} e^{\frac{-2jik\pi}{m}} + e^{\frac{-ik\pi}{m}} \sum_{j=0}^{m-1} y_{2j+1} e^{\frac{-2jik\pi}{m}} = E_k + e^{\frac{-ik\pi}{m}} O_k \quad (9)$$

E_k и O_k имеют ту же форму, что и A_k, а значит и к ним можно рекурсивно применить алгоритм Кули-Тьюки, а свойство периодичности этих сумм позволяет считать E_k и O_k только для половины значений. Само свойство:

$$E_{k\pm m} = \sum_{j=0}^{m-1} y_{2j} e^{\frac{-2ji(k\pm m)\pi}{m}} = \sum_{j=0}^{m-1} y_{2j} e^{\frac{-2jik\pi}{m}} = E_k \quad (10)$$

$$O_{k\pm m} = \sum_{j=0}^{m-1} y_{2j+1} e^{\frac{-2ji(k\pm m)\pi}{m}} = \sum_{j=0}^{m-1} y_{2j+1} e^{\frac{-2jik\pi}{m}} = O_k \quad (11)$$

Искомая функция, возвращающая комплексные коэффициенты полинома и функция, реализующая алгоритм Кули-Тьюки, представлены в листинге 3.

```
1  def fft_Ak(y_nodes):
2      N = len(y_nodes)
3      if N <= 1: return y_nodes
4      E = fft_Ak(y_nodes[0::2])
5      E = np.concatenate((E,E))
6      O = fft_Ak(y_nodes[1::2])
7      O = np.exp(-2j*np.pi*np.arange(N)/N)*np.concatenate((O,O))
8      return E+O
9
10 def fft_coef(y_nodes):
11     N=len(y_nodes)
12     return (-1)**np.arange(N)/ (N+1)*fft_Ak(y_nodes)
```

Функция fft_Ak должна возвращать то же, что и функция np.fft.fft. С помощью функции numpy.allclose правильность работы первой можно проверить. В нашем случае был возвращён результат True, значит функция fft_Ak возвращает тот же массив, что и функция np.fft.fft, а значит она реализована правильно.

2.3. Определение периодов сезонности

Для начала требуется извлечь тригонометрические коэффициенты из комплексного коэффициента $\hat{a}_k$ по формуле:

$$a_k = 2 * \mathrm{Re}(\hat{a}_k), \qquad b_k = -2 * \mathrm{Im}(\hat{a}_k)$$

Зависимость коэффициентов от k показана на рисунке 8:

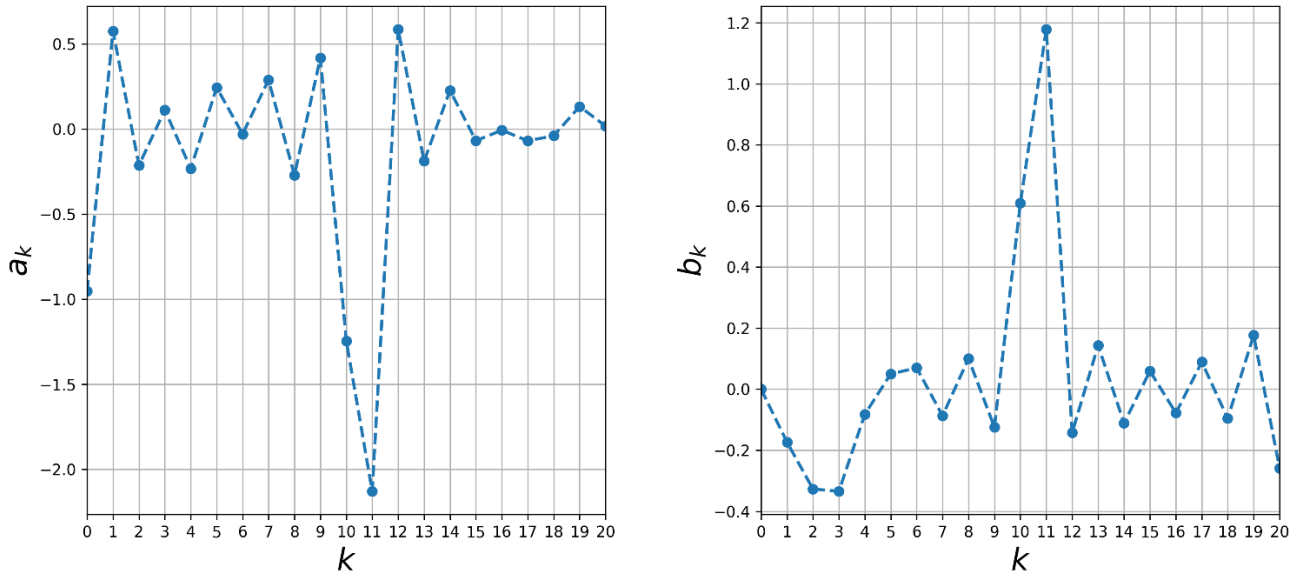

Рисунок 8 – графики зависимости коэффициента при косинусе и синусе от k.

Как видно из графиков, наибольший по модулю коэффициент - a_{11}, что соответствует частоте k=11. С помощью данной частоты можно найти период. Зная, что одной точке соответствует один месяц, а было использовано 128 точек:

$$T = \frac{128}{11} \approx 11{,}6 \text{ мес.}$$

Также на обоих графиках видно, что следующий по модулю коэффициент равен 10, а значит, при разбиении на количество узлов, большее в два раза, период бы немного увеличился. При увеличении разбиения период будет стремиться к одному году. Сейчас это будет косвенно доказано:

- Известно, что искомая частота находится между 10 и 11.
- $T = \frac{128}{12} = 10,(6)$ – частота, соответствующая периоду в ровно один год.

- Если сравнить величины амплитуд, соответствующие частотам 10 и 11, то «доля» частоты 11 примерно равна 0.63, в случае косинусов и 0.66 в случае синусов, что и наводит на мысль, что «спрятанная» между 10 и 11 частота и будет примерно равна 10,6 в случае разбиения на 128 месяцев, что близко к частоте, соответствующей периоду в один год.

Поэтому итоговым значением периода сезонности данного временного ряда будет один год. График, изображённый на рисунке 9 – ещё одно тому подтверждение.

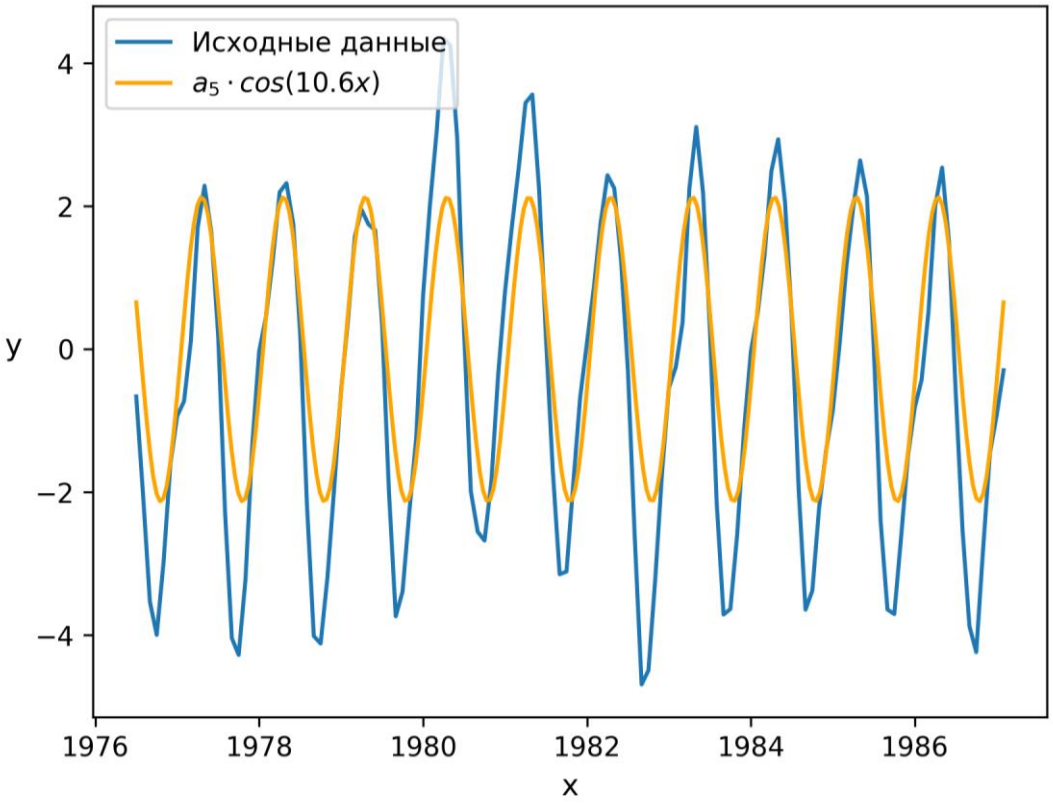

Рисунок 9 – график исходного временного ряда и функции, отображающей период в один год.

Заключение

В ходе выполнения данной лабораторной работы была изучена полиномиальная регрессия, суть которой сводится не только к задаче минимизации погрешности, но и поиску оптимальных значений коэффициентов, которые, ввиду наличия проблемы переобучения, практически никогда не совпадают с теми, что соответствуют минимальной погрешности для данной выборки. Поэтому был изучен подход разбиения данных, на обучающую и тестирующую выборки. Основываясь на данном разбиении, был подобран оптимальный параметр для L_2-регуляризации, которая призвана уменьшить такое явление как паразитные осцилляции. Лишний раз была продемонстрирована важность нормализации данных, участвующих в вычислениях на примере вычислительной неустойчивости матрицы Вандермонда для ненормализованных данных.

Для анализа временного ряда он был приведён к стационарному виду двумя способами, однако для автора данного отчёта остаётся открытым вопрос о странной форме стандартного ряда, полученного через взятие разностей.

Для последующего определения периодов сезонности ряда, было реализовано быстрое преобразование Фурье, основанное на алгоритме Кули-Тьюки, позволяющее аппроксимировать набор данных с помощью тригонометрического полинома. При анализе полученной аппроксимации, была выведена закономерность, позволяющая более точно определить период сезонности временного ряда, чем это позволяет сделать подход со взятием частоты, соответствующей максимальной амплитуде, как базовой

Список использованных источников

1. **Першин А.Ю.** *Лекции по вычислительной математике (черновик)*. [archrk6.bmstu.ru] // Кафедра РК6 МГТУ им. Н.Э. Баумана, Москва, 2020, 145.
2. **Першин А.Ю.** *Вычислительная математика, лекция №7*. Видеохостинг «YouTube» [https://www.youtube.com/watch?v=PvvSvFg5Uzk] // (дата обращения (18.04.2021).
3. **Першин А.Ю.** *Вычислительная математика, лекция №8*. Видеохостинг «YouTube» [https://www.youtube.com/watch?v=2l6Q-vOZjCE] // (дата обращения (18.04.2021).
4. **Першин А.Ю.** *Вычислительная математика, лекция №9*. Видеохостинг «YouTube» [https://www.youtube.com/watch?v=s2TfeF-22zo] // (дата обращения (18.04.2021).
5. Quickstart — python-docx 0.8.10 documentation [https://python-docx.readthedocs.io/en/latest/user/quickstart.html] // (дата обращения (18.04.2021).