**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 «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# Лабораторная работа № 1 по дисциплине «Методы машинного обучения»

**Тема** Модель полиномиальной регрессии и феномен Рунге

**Студент** Сапожков А.М,

**Группа** ИУ7-23М

**Преподаватель** Солодовников В.И.

Москва, 2025

# Содержание

# ВВЕДЕНИЕ

Полиномиальная регрессия представляет собой одну из наиболее широко используемых моделей машинного обучения, нацеленную на аппроксимацию нелинейной зависимости между параметрами. Эта модель служит основным инструментом для прогнозирования значений целевой функции на основе набора входных переменных.

Однако, несмотря на простоту и эффективность этой модели при определённых условиях, она не всегда обеспечивает точные результаты. Практика показывает, что если степень полинома слишком велика по отношению к размеру обучающей выборки или если данные содержат явные или неточные измерения, то модель может стать крайне чувствительной ко всем шумам в данных, что приводит к переобучению. Этот феномен известен как феномен Рунге и является классическим примером одной из наиболее распространённых проблем машинного обучения — проблемы переоценки (переобучения).

Переоценка происходит тогда, когда модель начинает слишком точно подгоняться к данным, а не нацеливается на выявление закономерностей в данных. Это может привести к тому, что модель будет прогнозировать значения целевой переменной с помощью случайных шумов, которые содержатся в исходном наборе данных.

Целью данной лабораторной работы изучение полиномиальной регрессии и феномена Рунге.

Задачи данной лабораторной работы:

1) создать обучающую выборку с использованием функции $y(x) = \theta_1 x + \theta_2 sin(x) + \theta_3$ и с добавлением шума с нормальным распределением;

2) построить модель полиномиальной регрессии, аппроксимирующей данные обучающей выборки;

3) найти оптимальную степень полинома для аппроксимации;

4) рассмотреть феномен (явление) Рунге;

5) рассчитать функционал эмпирического риска (функционал качества) для обучающей и контрольной выборок (вывести графики);

6) оценить обобщающую способность;

7) найти оптимальную степень полинома для аппроксимации.

# 1 Аналитическая часть

## 1.1 Модель полиномиальной регрессии

Полиномиальная регрессия является одним из наиболее широко используемых алгоритмов машинного обучения, нацеленных на аппроксимацию нелинейной зависимости между переменными. Основная идея состоит в том, чтобы найти полиномиальное выражение для целевой переменной на основе набора входных переменных.

Пусть $\mathbf{x}$ является набором входных переменных и $y$ — целевой функцией, которую мы хотим предсказать. Цель полиномиальной регрессии — найти коэффициенты $w = (w_0, w_1, ..., w_n)$ в таких, что

$$y \approx w^T \phi(\mathbf{x})$$

где $\phi$ — функция, которая преобразует исходные данные в набор признаков.

## 1.2 Феномен Рунге

Феномен Рунге является классическим примером проблемы переоценки в контексте полиномиальной регрессии. Этот феномен был впервые описан Карлом Давидом Рунге в 1880 году и относится к проблеме решения разностных уравнений.

В контексте полиномиальной регрессии феномен Рунге проявляется в следующем: когда степень полинома слишком велика по отношению к размеру обучающей выборки, модель начинает воспроизводить не закономерности в данных, а случайные шумы. Это приводит к тому, что точность её прогнозов снижается.

# 2 Конструкторская часть

## 2.1 Обучение моделей

Для обучения моделей полиномиальной регрессии мы будем использовать метод наименьших квадратов ( Least Squares Method ). Этот метод основан на минимизации функции потерь, представляющей сумму квадратичных отклонений данных от полученных предсказаний. Мы воспользуемся следующей функцией потерь:

$$L = \sum_i (y_i - f(x_i))^2, \tag{2.1}$$

где $y_i$ — фактические значения, а $f(x_i)$ — предсказанные значения для $i$-го наблюдения.

## 2.2 Подбор степени полинома

Следующий этап — подбор оптимальной степени полинома для модели. Для этого мы будем использовать информационный критерий Акаике (AIC). Критерий AIC определяется следующим образом:

$$AIC = 2k - 2log(L), \tag{2.2}$$

где $k$ — количество параметров в модели, а $L$ — функция потерь. Таким образом, критерий не только вознаграждает за качество приближения, но и штрафует за использование излишнего количества параметров модели. Считается, что наилучшей будет модель с наименьшим значением критерия AIC.

На основе значения критерия AIC мы сможем выбрать модель с оптимальной степенью полинома, которая обеспечивает наилучшее соотношение точности и простоты модели.

# 3 Технологическая часть

## 3.1 Средства реализации

В качестве языка программирования для реализации выбранных алгоритмов был выбран язык программирования Python ввиду наличия библиотек для обучения регрессионных моделей, таких как sklearn и numpy.

## 3.2 Реализация алгоритмов

На листинге 3.1 представлена реализация обучения модели полиномиальной регрессии с использованием метода наименьших квадратов и выбор оптимальной степени полинома по критерию AIC.

Листинг 3.1 — Обучение модели полиномиальной регрессии и выбор оптимальной степени полинома

```
import numpy as np
import matplotlib.pyplot as plt

x = np.random.rand(150)*10
y = 1/2*x + 2*np.sin(x) + 5
y = y + np.random.randn(150)*0.5
plt.scatter(x,y)
plt.show()

degrees = np.arange(1, 11)
n = len(y)

aic_scores = []
for degree in degrees:
  p = np.polyfit(x, y, degree)
  y_hat = np.polyval(p, x)
  rss = np.sum((y - y_hat) ** 2) # residual sum of squares
  aic = n * np.log10(rss / n) + 2 * (degree + 1)
  aic_scores.append(aic)

print("AIC: ", aic_scores)

best_degree = degrees[np.argmin(aic_scores)]
print("Степень полинома, обеспечивающая наибольшую точность: ", best_degree)
model = np.poly1d(np.polyfit(x, y, best_degree))
polyline = np.linspace(np.min(x), np.max(x), 250)
```

```
plt.scatter(x, y)
plt.plot(polyline, model(polyline), color='red')
plt.show()

print(model)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size
   =0.20, random_state=777)
model = np.poly1d(np.polyfit(X_train, y_train, best_degree))
X_train.sort()
X_test.sort()
plt.scatter(x, y)
plt.plot(X_train, model(X_train), color='red', label='train data')
plt.plot(X_test, model(X_test), color='green', label='test data')
plt.legend()
plt.show()
```

На листинге 3.2 представлена реализация алгоритма визуализации феномена Рунге.

Листинг 3.2 — Визуализация феномена Рунге

```
import numpy as np
import matplotlib.pyplot as plt

def source_func(x):
  return 1 / (1 + 25*x**2)

def mse(y_true, y_pred):
  return np.mean((y_true - y_pred)**2)

l = 50
train_set = np.array([4*(i-1)/(l-1) - 2 for i in range(1, l+1)])
train_labels = source_func(train_set)

k = 50
test_set = np.array([4*(i-0.5)/(k-1) - 2 for i in range(1, k)])
test_labels = source_func(test_set)

x = np.arange(-2, 2, 0.01)
plt.plot(x, source_func(x))
plt.xlabel('x')
plt.ylabel('y')
```

```
plt.title('Функция y(x) = 1 / (1 + 25*x^2)')
plt.show()

degrees = range(1, 30)
train_loss = []
test_loss = []
for deg in degrees:
  p = np.polyfit(train_set, train_labels, deg)
  model = np.poly1d(p)
  train_pred = np.polyval(model, train_set)
  test_pred = np.polyval(model, test_set)
  train_loss.append(mse(train_labels, train_pred))
  test_loss.append(mse(test_labels, test_pred))

plt.plot(degrees, train_loss, label='Training loss')
plt.plot(degrees, test_loss, label='Test loss')
plt.legend()
plt.xlabel('Степень полинома')
plt.ylabel('Среднеквадратическая ошибка')
plt.show()

min_idx = np.argmin(test_loss)
best_deg = degrees[min_idx]
print("Оптимальная степень полинома:", best_deg)
plt.plot(x, source_func(x), label='Реальное')

p = np.polyfit(train_set, train_labels, best_deg)
model = np.poly1d(p)
plt.plot(x, np.polyval(model, x), label='Степень {}'.format(best_deg))

plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.show()

plt.plot(x, source_func(x), label='Реальное')
for deg in [best_deg-2, best_deg, best_deg+2]:
  p = np.polyfit(train_set, train_labels, deg)
  model = np.poly1d(p)
  plt.plot(x, np.polyval(model, x), label='Степень {}'.format(deg))
```

```
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.show()

plt.plot(x, source_func(x), label='Реальное')
for deg in range(5,24):
  p = np.polyfit(train_set, train_labels, deg)
  model = np.poly1d(p)
  plt.plot(x, np.polyval(model, x), label='Степень {}'.format(deg))

plt.xlabel('x')
plt.ylabel('y')
plt.show()

for i, deg in enumerate(degrees):
  print(f"Степень {deg}:")
  print(f"Train MSE = {train_loss[i]:.3f}")
  print(f"Test MSE = {test_loss[i]:.3f}\n")

l = 50
train_set = np.array([4*(i-1)/(l-1) - 2 for i in range(1, l+1)])
train_labels = source_func(train_set)

k = 50
test_set = np.array([4*(i-0.5)/(k-1) - 2 for i in range(1, k)])
test_labels = source_func(test_set)

deg = 55
p = np.polyfit(train_set, train_labels, deg)
model = np.poly1d(p)

plt.plot(train_set, np.polyval(model, train_set), color='red', label
   ='Степень {} (train)'.format(deg), marker = 'o')
plt.scatter(test_set, test_labels, color='blue', marker = 'o',
   facecolors='none')
plt.plot(test_set, np.polyval(model, test_set), color='blue', label='
   Степень {} (test)'.format(deg), marker = 'o', markerfacecolor='none
   ')
plt.plot(x, source_func(x), color='green', label='Исходная кривая')
plt.xlabel('x')
```

```
plt.ylabel('y')
plt.xlim(-0.25, 2)
plt.ylim(-0.1, 1.5)
plt.legend()
plt.show()
```

# 4 Исследовательская часть

## 4.1 Среда для тестирования

Для тестирования разработанного алгоритма применялась облачная платформа Google Colab, не требующая установки ПО на локальный компьютер.

## 4.2 Обучение модели полиномиальной регрессии

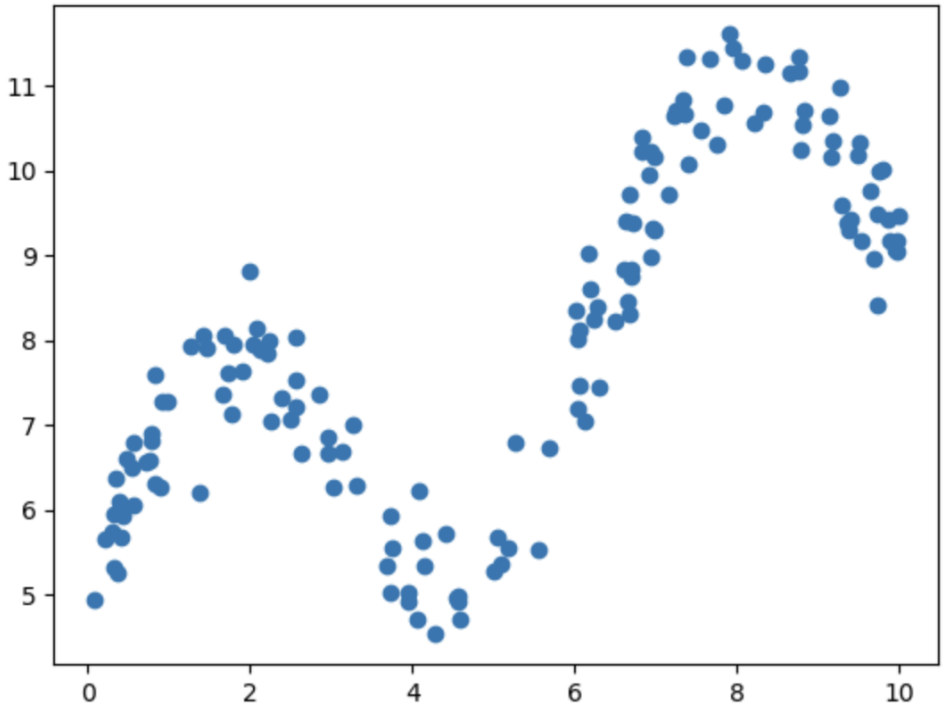


Рисунок 4.1 — Исходные данные

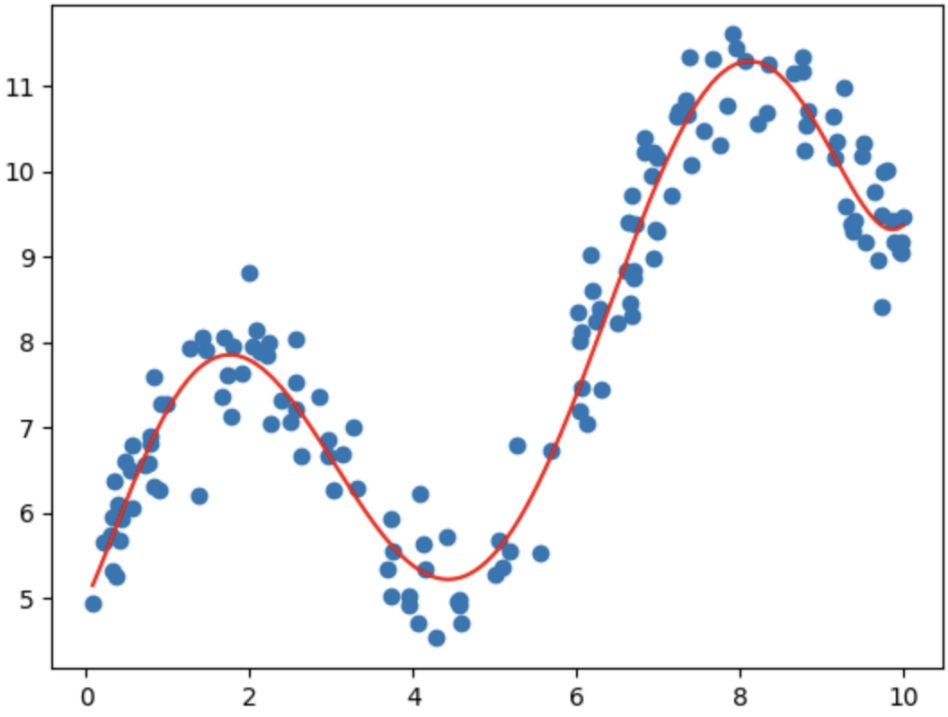


Рисунок 4.2 — Аппроксимация полиномом оптимальной степени (6)

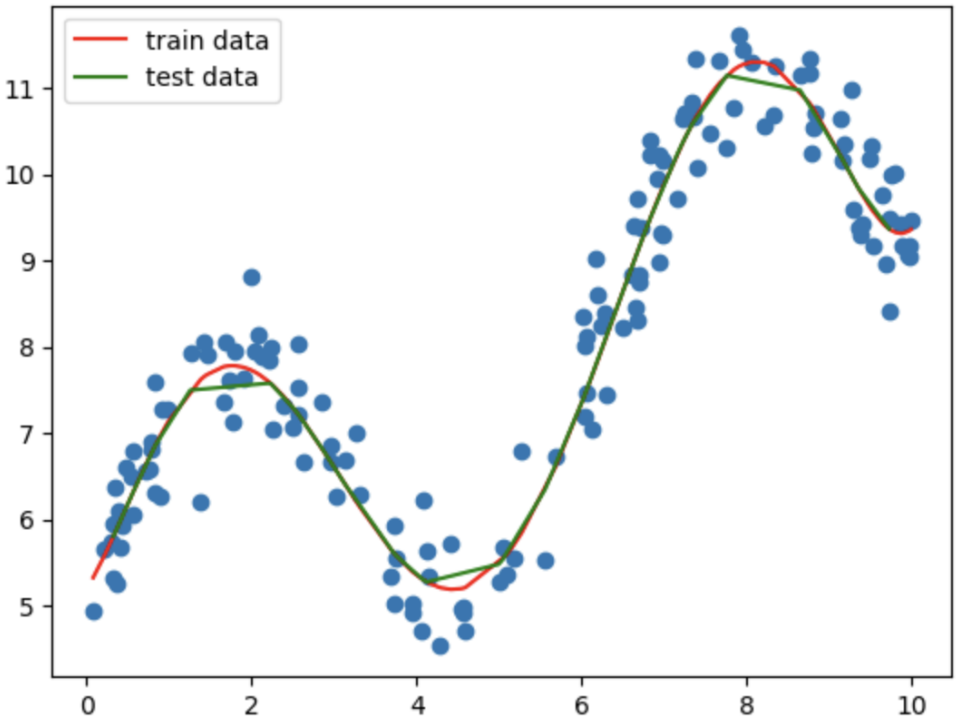


Рисунок 4.3 — Разделение выборки на обучающую и тестовую в отношении 80 на 20

## 4.3 Визуализация феномена Рунге

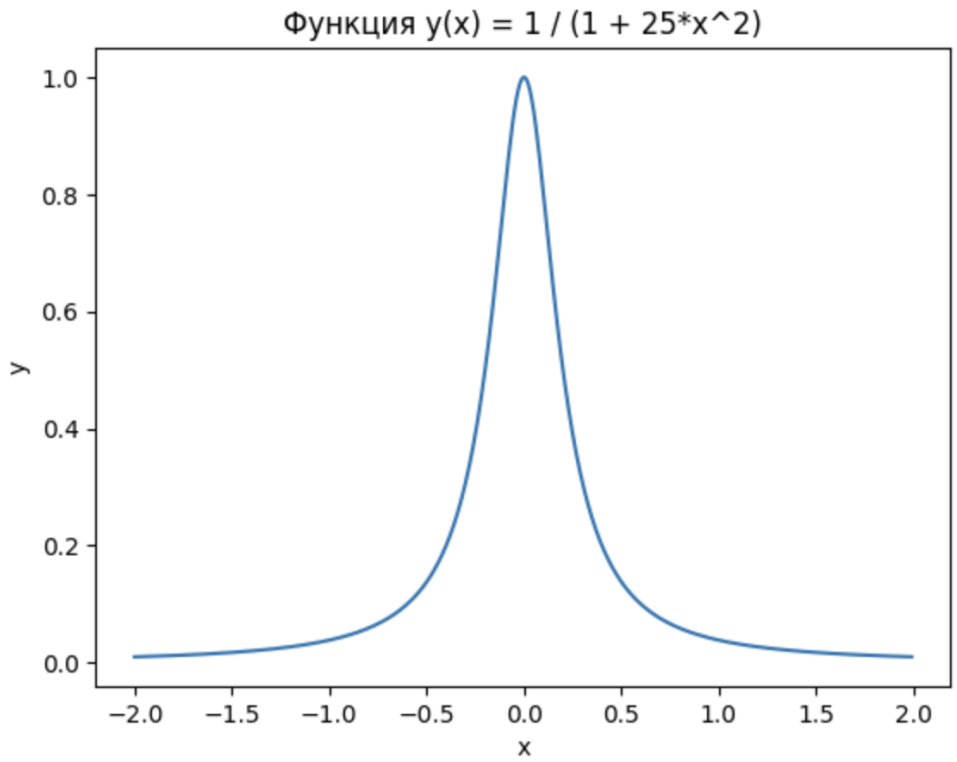


Рисунок 4.4 — Исходные данные

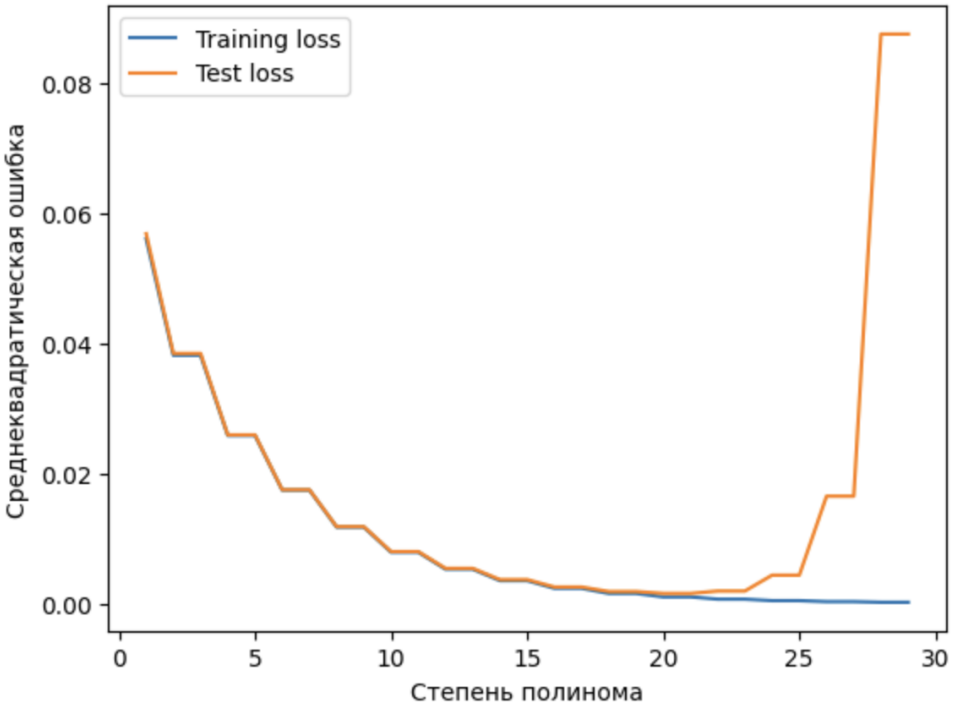


Рисунок 4.5 — Зависимость точности обучения от степени полинома

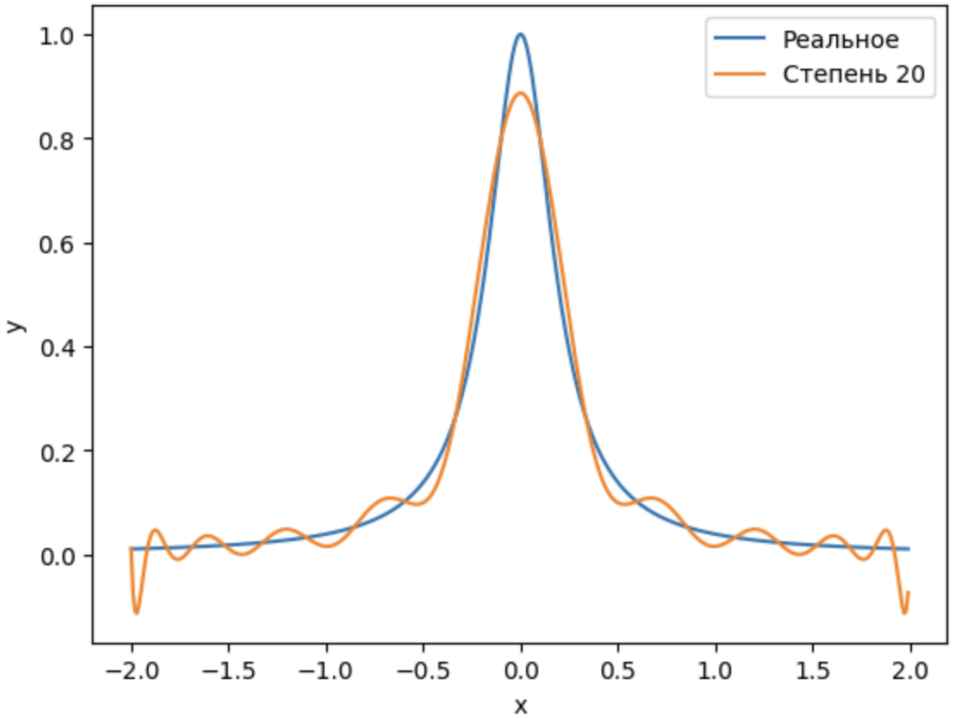


Рисунок 4.6 — Сравнение исходной кривой с полиномом оптимальной степени (20)

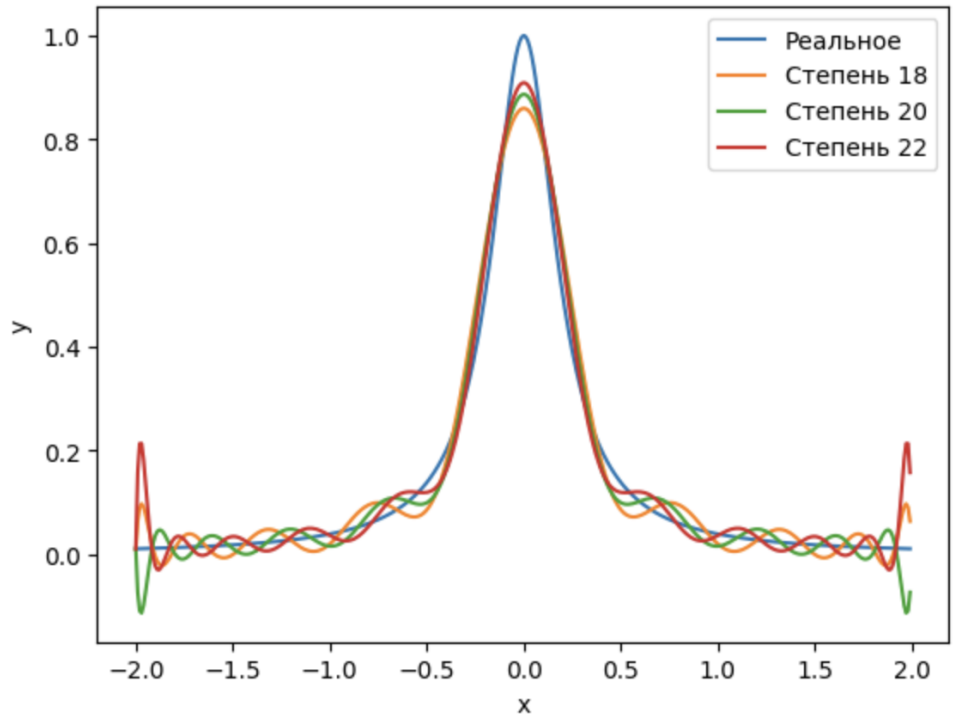


Рисунок 4.7 — Сравнение исходной кривой с полиномами степеней 18, 20 и 22

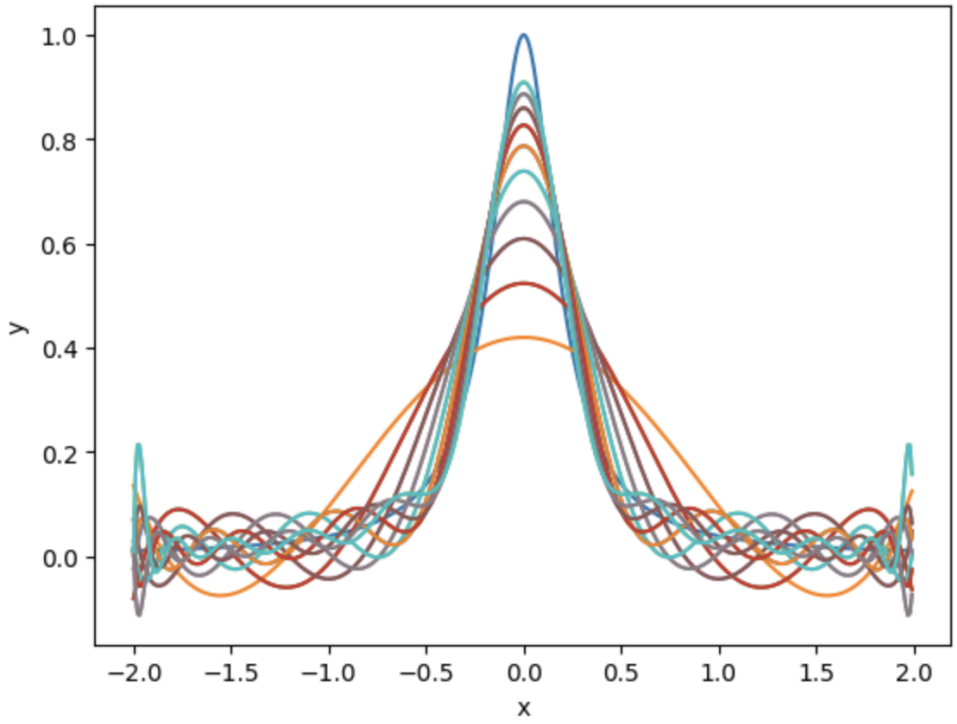


Рисунок 4.8 — Сравнение исходной кривой с полиномами степеней от 5 до 24

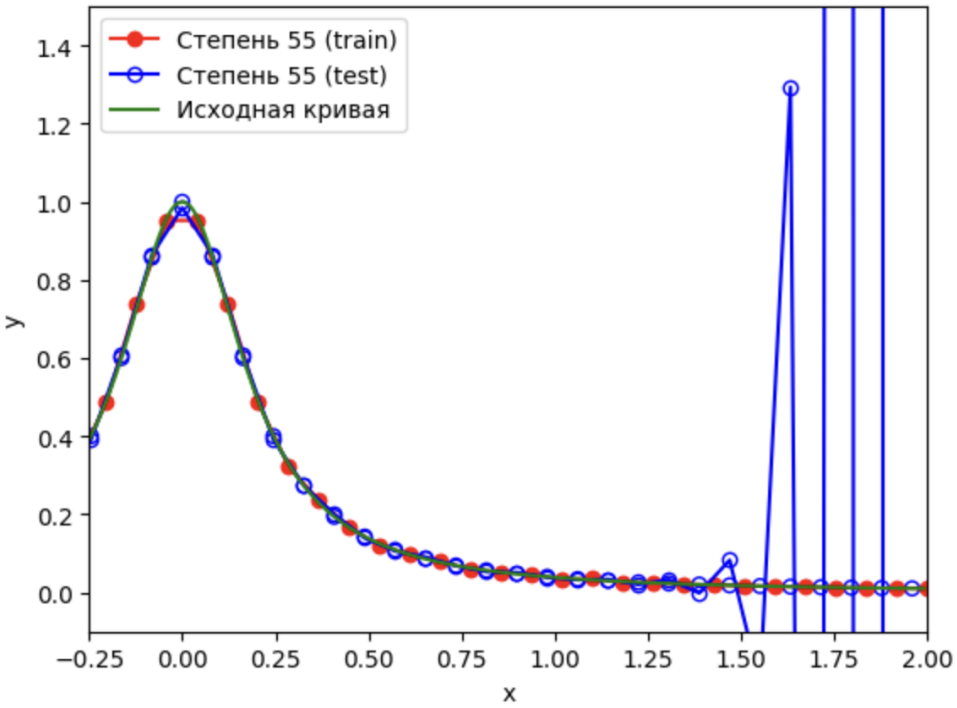


Рисунок 4.9 — Иллюстрация феномена Рунге для исходной кривой, аппроксимированной полиномом степени 55

# ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы была изучена модель полиномиальной регрессии и феномен Рунге. Все поставленные задачи были выполнены.

1) Создана обучающая выборка с использованием функции $y(x) = \theta_1 x + \theta_2 sin(x) + \theta_3$ и с добавлением шума с нормальным распределением;

2) Построена модель полиномиальной регрессии, аппроксимирующей данные обучающей выборки;

3) Найдена оптимальная степень полинома для аппроксимации.

4) Рассмотрен феномен (явление) Рунге;

5) Рассчитан функционал эмпирического риска (функционал качества) для обучающей и контрольной выборок (выведены графики);

6) Оценена обобщающая способность;

7) Найдена оптимальная степень полинома для аппроксимации;

Оптимальная степень полинома для аппроксимации зависимости $y(x) = \theta_1 x + \theta_2 sin(x) + \theta_3$ — 6.

Оптимальная степень полинома для аппроксимации зависимости $y(x) = \frac{1}{1+25x^2}$ — 20. Феномен Рунге можно наблюдать при степени полинома 55.