Поповкин Артемий Андреевич
Б9122-02.03.01 СЦТ

Постановка задачи

Требуется для модели **Лотки-Вольтерры** определить численность популяции в будущем на основе начальной численности жертв и хищников, темпов роста обоих популяций и коэффициентов, отражающих взаимодействия между ними.

Описание модели

Описание

Модель Лотки-Вольтерры описывает динамику взаимодействия двух биологических видов с отношениями "Хищиник-Жертва" или "Паразит-Хозяин".

Обозначения

- t — время (размерность: годы)

Входные данные

- H_0 — начальная численность популяции хищников (размерность: тысячи)
- P_0 — начальная численность популяции жертв (размерность: тысячи)

Фазовые переменные

- $H\left(t\right)$ — численность популяции хищников в момент времени .
- $P\left(t\right)$ — численность популяции жертв в момент времени t.

Параметры

- α — коэффициент рождаемости жертв.
- β — коэффициент убийства жертв хищниками.
- γ — коэффициент смертности хищников.
- δ — коэффициент рождаемости хищников.

Уравнение

$$\begin{cases} \dot{P} = (\alpha - \beta H)P \\ \dot{H} = (-\gamma + \delta P)H \end{cases}$$

Решением уравнения будет называть функцию, удовлетворяющую следующей задаче Коши:

$$\begin{cases} \dot{P} = (\alpha - \beta H)P \\ \dot{H} = (-\gamma + \delta P)H \\ P(0) = P_0 \quad H(0) = H_0 \end{cases}$$

Аналитическое исследование системы

Условие устойчивости

Для начала найдём равновесия модели:

$$\begin{cases} (\alpha - \beta H)P = 0 \\ (-\gamma + \delta P)H = 0 \end{cases}$$

Решив систему, получаем следующие два равновесия:

Тривиальное

$$P_t = H_t = 0$$

Нетривиальное

$$\begin{cases} \alpha - \beta H = 0 \\ -\gamma + \delta P = 0 \end{cases} \implies \begin{cases} H_{nt} = \dfrac{\alpha}{\beta} \\ P_{nt} = \dfrac{\gamma}{\delta} \end{cases}$$

Теперь исследуем полученные равновесные точки на устойчивость.
Для этого найдем собственные значения матрицы Якоби системы в данных точках.

$$J - \lambda E = \begin{pmatrix} \alpha - \beta H - \lambda & -\beta P \\ \delta H & -\gamma + \delta P - \lambda \end{pmatrix}$$

Тривиальное равновесие

$$\det(J - \lambda E) \underset{P=P_t,\, H=H_t}{=} -(\alpha - \lambda)(\gamma + \lambda) = 0 \implies \begin{cases} \lambda_1 = \alpha \\ \lambda_2 = -\gamma \end{cases}$$

Так как $\operatorname{Im}\lambda_{1,2} = 0,\ \operatorname{Re}\lambda_1 > 0,\ \operatorname{Im}\lambda_2 < 0$, можем заключить, что тривиальное равновесие неустойчиво и является седловой точкой.

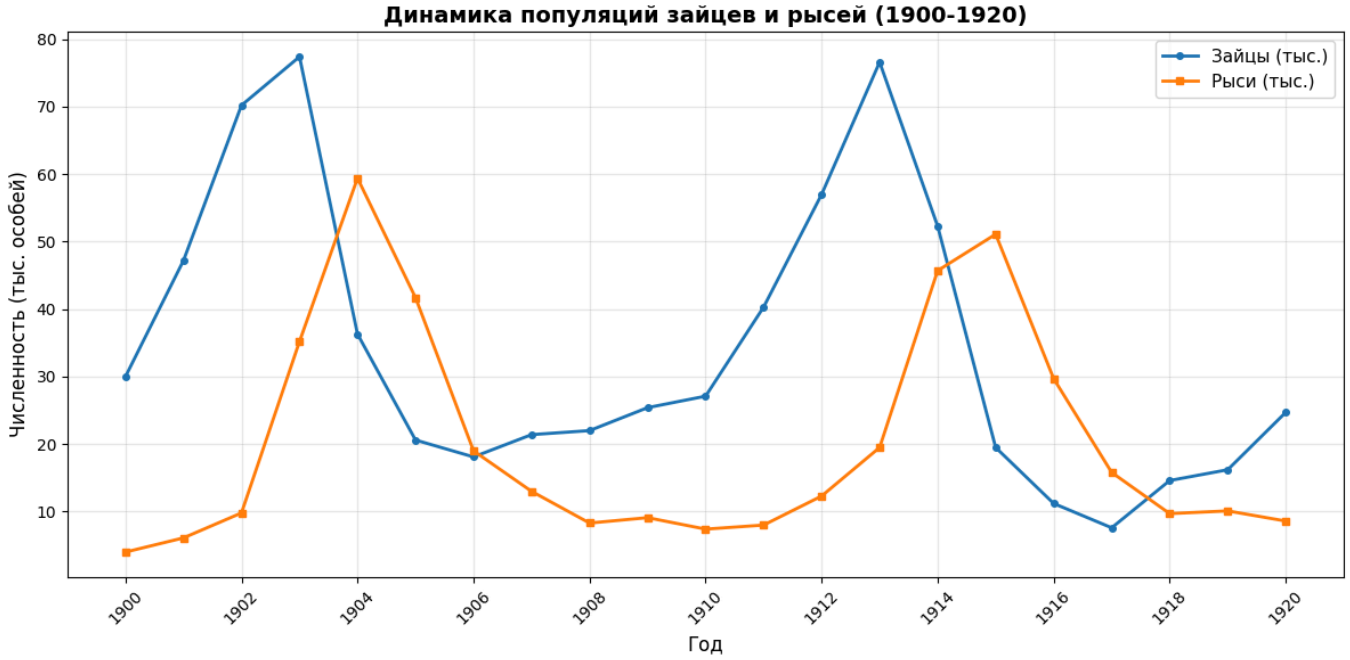
Нетривиальное равновесие

$$\det(J - \lambda E)|_{x=x_{nt}, y=y_{nt}} = \lambda^2 + \alpha\gamma = 0 \implies \lambda_{1,2} = \pm\sqrt{\alpha\gamma}\, i$$

Исходя из того, что $\operatorname{Re}\lambda_{1,2} = 0,\ \operatorname{Im}\lambda_1 > 0,\ \operatorname{Im}\lambda_2 < 0$, получаем, что нетривиальное равновесие является центром (устойчиво).

Численные эксперименты

Был выбран временной диапазон с 1900 по 1920 гг.
Популяция зайцев и рысей указана в тысячах.

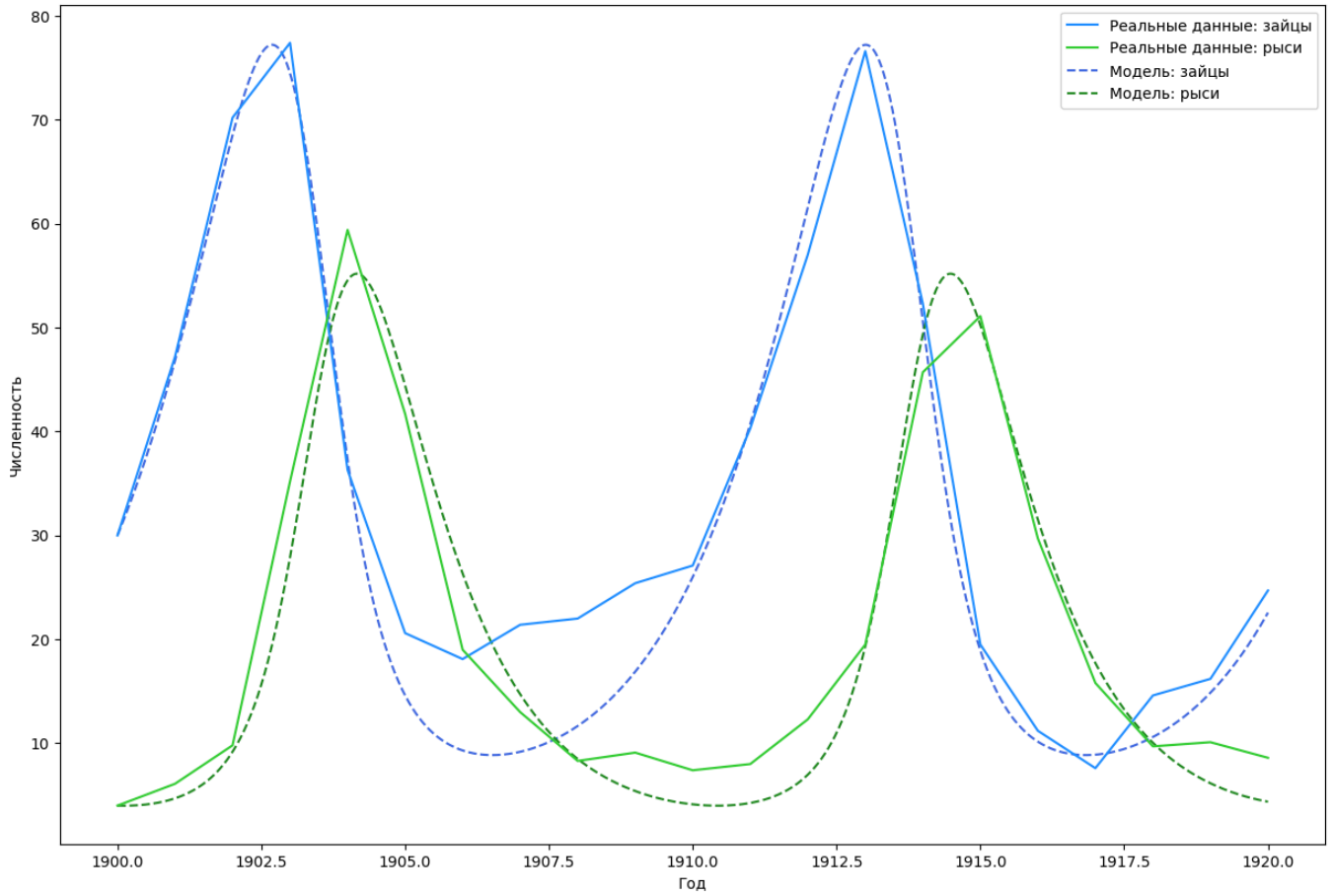


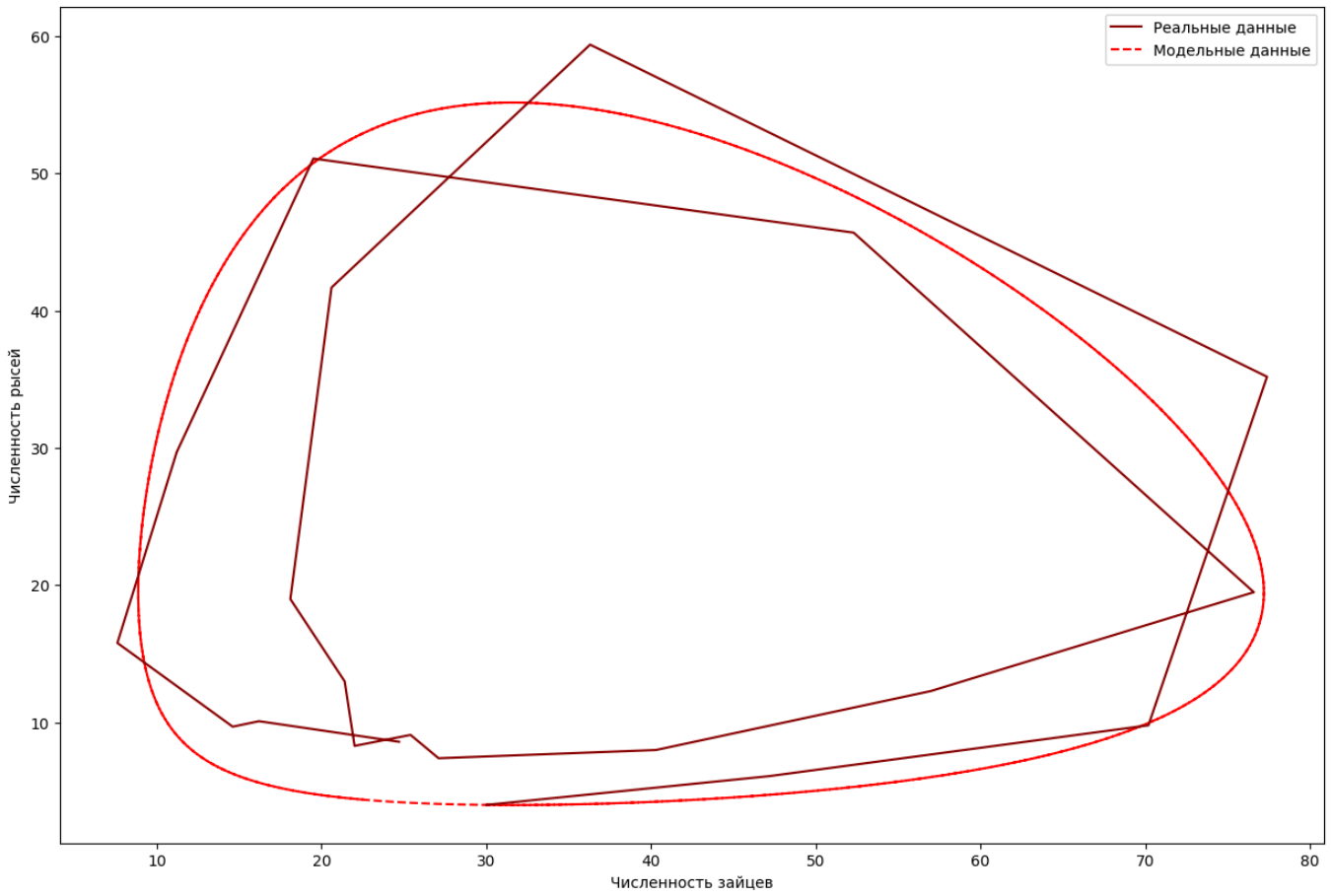
Основные расчёты

Для численного решения задачи Коши, описывающей нашу модель, был выбран метод семейства Кунге-Кутта 4 порядка, в силу его достаточной точности и простоты реализации.

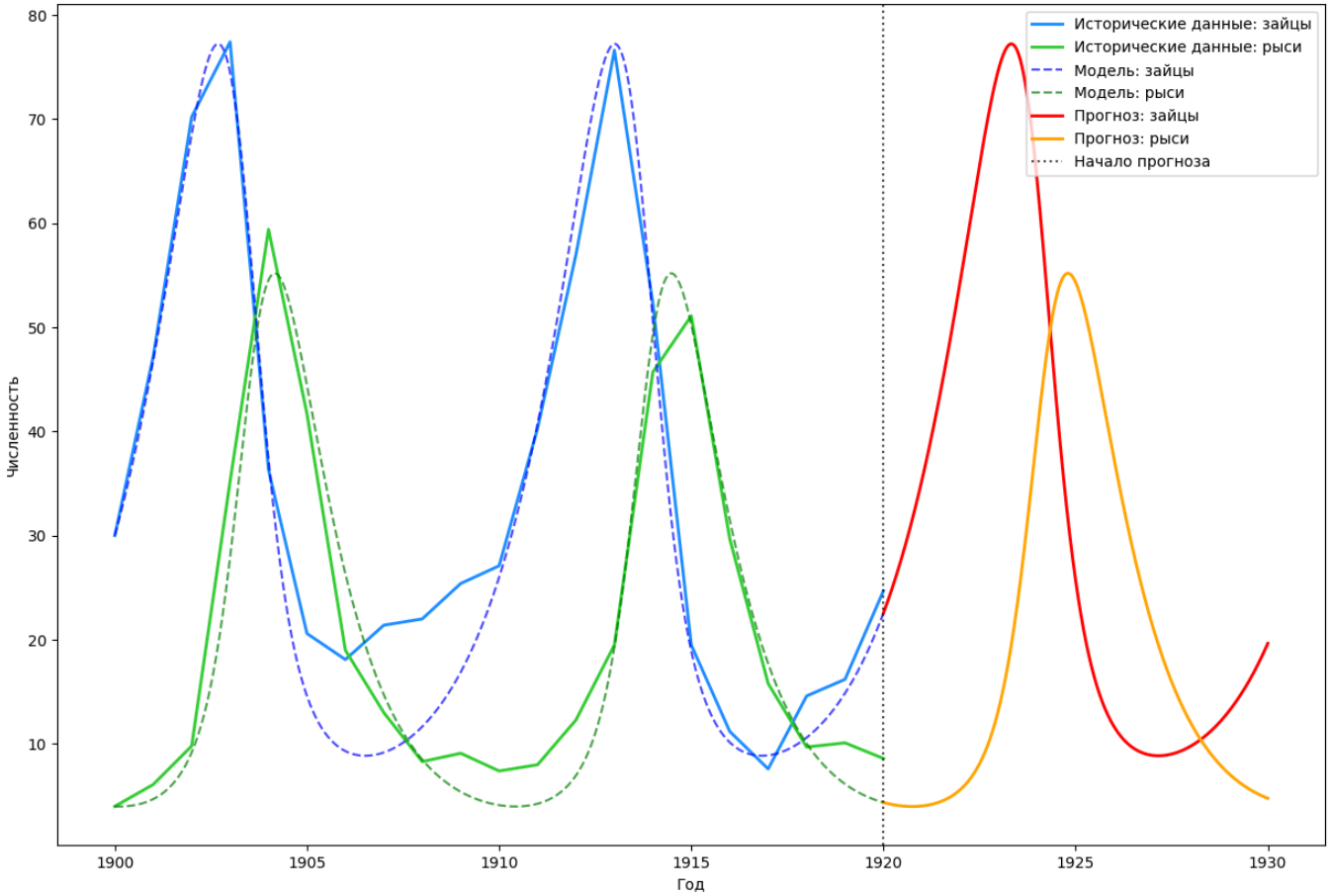
Для подбора параметров использовался нелинейный метод наименьших квадратов, предоставляемый пакетом scipy языка программирования Python.

В итоге были получены оптимальные параметры и численно найдены решения системы ДУ. Сравнение модельных и реальных данных представлено на рисунках 1-2.


Реальные данные: зайцы
Реальные данные: рыси
Модель: зайцы
Модель: рыси
Численность
Год
80
70
60
50
40
30
20
10
1900.0
1902.5
1905.0
1907.5
1910.0
1912.5
1915.0
1917.5
1920.0


Реальные данные
Модельные данные
Численность рысей
Численность зайцев
60
50
40
30
20
10
10
20
30
40
50
60
70
80

Используя полученную модель был построен прогноз динамики экосистемы на следующие 10 лет. Результат на 3-м рисунке



Заключение

В результате проведенного исследования была успешно применена модель Лотки-Вольтерры для анализа реальных данных о популяциях рыси и зайца.

Аналитическое исследование системы позволило идентифицировать два состояния равновесия: тривиальное, являющееся неустойчивым, и нетривиальное, которое представляет собой устойчивый центр.
Это теоретически подтверждает циклический, колебательный характер взаимодействия хищника и жертвы, наблюдаемый в природе.

Практическим итогом работы стала численная реализация модели. С помощью нелинейного метода наименьших квадратов были подобраны оптимальные параметры системы, что позволило добиться качественного соответствия модельных траекторий реальным данным.

Как видно из графиков, модель корректно воспроизводит общий тренд, хотя и не всегда точно совпадает с конкретными точками, что обуславливается простотой предположенной модели.

Таким образом, классическая модель «хищник-жертва» подтвердила свою адекватность для качественного анализа динамики данной экосистемы.

Использованные данные

https://mc-stan.org/learn-stan/case-studies/lotka-volterra-predator-prey.html#data-lynx-and-hare-pelts-in-canada
https://github.com/stan-dev/example-models/blob/master/knitr/lotka-volterra/hudson-bay-lynx-hare.csv

Листинг кода

```
# Imports
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import minimize
from scipy.integrate import solve_ivp
# Funcs
## Math
def objective(
    params: tuple[float, float, float, float], data: tuple[float, float,
float], y0
):
    H_data, L_data, t_data = data
    y_pred = model(params, t_data, y0)
    error_H = np.mean((y_pred[0] - H_data) ** 2)
    error_L = np.mean((y_pred[1] - L_data) ** 2)
    return error_H + error_L


def model(params: tuple[float, float, float, float], t, y0):
    solution = solve_ivp(
        lotka_volterra, [t[0], t[-1]], y0, args=(params), t_eval=t,
method="RK45"
    )
    return solution.y


def lotka_volterra(t, y, alpha, beta, delta, gamma):
    # alpha, beta, delta, gamma = params
    H, L = y
    dHdt = alpha * H - beta * H * L
    dLdt = delta * H * L - gamma * L
    return [dHdt, dLdt]
```

```
def runge_kutta_4th_order(
    f, y0, t_span, n_steps: int, params: tuple[float, float, float, float]
):
    alpha, beta, delta, gamma = params
    t_start, t_end = t_span
    t = np.linspace(t_start, t_end, n_steps)
    h = (t_end - t_start) / (n_steps - 1)
    y = np.zeros((len(y0), n_steps))
    y[:, 0] = y0
    for i in range(n_steps - 1):
        k1 = h * np.array(f(t[i], y[:, i], alpha, beta, delta, gamma))
        k2 = h * np.array(f(t[i] + h / 2, y[:, i] + k1 / 2, alpha, beta,
delta, gamma))
        k3 = h * np.array(f(t[i] + h / 2, y[:, i] + k2 / 2, alpha, beta,
delta, gamma))
        k4 = h * np.array(f(t[i] + h, y[:, i] + k3, alpha, beta, delta,
gamma))
        y[:, i + 1] = y[:, i] + (k1 + 2 * k2 + 2 * k3 + k4) / 6
    return t, y
# Data
df = pd.read_csv("../data/hudson-bay-lynx-hare.csv")
df

P_col_name = " Hare"
H_col_name = " Lynx"
t_col_name = "Year"

t_data = df[t_col_name].values
H_data = df[P_col_name].values
L_data = df[H_col_name].values
data = (H_data, L_data, t_data)

y0 = [H_data[0], L_data[0]]
# Calculation
## Params calculation
initial_guess = [1.0, 0.1, 0.01, 0.375]
bounds = [(0.001, 10), (0.001, 1), (0.001, 1), (0.001, 1)]

result = minimize(objective, initial_guess, args=(data, y0), bounds=bounds,
method="L-BFGS-B")
params = result.x
alpha, beta, delta, gamma = params
alpha, beta, delta, gamma
## Calculation on hystorical data
```

```
H0 = df[P_col_name].iloc[0]
L0 = df[H_col_name].iloc[0]
t_span = (df[t_col_name].iloc[0], df[t_col_name].iloc[-1])
n_steps = 1000
y0 = [H0, L0]
t_rk, y_rk = runge_kutta_4th_order(lotka_volterra, y0, t_span, n_steps,
params)
# Visualisation
# Визуализация результатов
plt.figure(figsize=(15, 10))
plt.plot(
    df[t_col_name],
    df[P_col_name],
    "dodgerblue",
    label="Реальные данные: зайцы",
    zorder=2,
)
plt.plot(
    df[t_col_name], df[H_col_name], "limegreen", label="Реальные данные:
рыси", zorder=2
)
plt.plot(t_rk, y_rk[0], "royalblue", linestyle="--", label="Модель: зайцы",
zorder=1)
plt.plot(t_rk, y_rk[1], "forestgreen", linestyle="--", label="Модель: рыси",
zorder=1)
plt.legend()
plt.xlabel("Год")
plt.ylabel("Численность")
plt.plot()

# Фазовый портрет
plt.figure(figsize=(15, 10))
plt.plot(df[P_col_name], df[H_col_name], "darkred", label="Реальные данные",
zorder=2)
plt.plot(y_rk[0], y_rk[1], "red", linestyle="--", label="Модельные данные",
zorder=1)
plt.legend()
plt.xlabel("Численность зайцев")
plt.ylabel("Численность рысей")
plt.plot()
H0_forecast = y_rk[0][-1]
L0_forecast = y_rk[1][-1]
last_year = df[t_col_name].iloc[-1]
# Период прогноза
forecast_years = 10
t_span_forecast = (last_year, last_year + forecast_years)
```

```
n_steps_forecast = 1000
# Прогноз с использованием последних известных значений
t_forecast, y_forecast = runge_kutta_4th_order(
    lotka_volterra,
    [H0_forecast, L0_forecast],
    t_span_forecast,
    n_steps_forecast,
    params,
)
# Визуализация прогноза
plt.figure(figsize=(15, 10))
# Исторические данные
plt.plot(
    df[t_col_name],
    df[P_col_name],
    "dodgerblue",
    label="Исторические данные: зайцы",
    linewidth=2,
)
plt.plot(
    df[t_col_name],
    df[H_col_name],
    "limegreen",
    label="Исторические данные: рыси",
    linewidth=2,
)
# Модель на историческом периоде
plt.plot(t_rk, y_rk[0], "blue", linestyle="--", label="Модель: зайцы",
alpha=0.7)
plt.plot(t_rk, y_rk[1], "green", linestyle="--", label="Модель: рыси",
alpha=0.7)
# Прогноз
plt.plot(
    t_forecast, y_forecast[0], "red", linestyle="-", label="Прогноз: зайцы",
linewidth=2
)
plt.plot(
    t_forecast,
    y_forecast[1],
    "orange",
    linestyle="-",
    label="Прогноз: рыси",
    linewidth=2,
)
# Вертикальная линия разделяющая историю и прогноз
plt.axvline(
```

```
    x=last_year, color="black", linestyle=":", alpha=0.7, label="Начало
прогноза"
)
plt.legend()
plt.xlabel("Год")
plt.ylabel("Численность")
plt.show()
```