Поповкин Артемий Андреевич
Б9122-02.03.01 СЦТ

Постановка задачи

Требуется для модели **конкуренции двух популяций** 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 конкуренции двух видов описывает, как два вида конкурируют за общий ресурс, такой как пища, территория, покупатели и др.

Обозначения

- t — время (размерность: годы)

Входные данные

- x_0 — начальная численность популяции хищников (размерность: тысячи)
- y_0 — начальная численность популяции жертв (размерность: тысячи)

Фазовые переменные

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

Параметры

- r_1, r_2 — темпы роста каждой из популяций (разница между рождаемостью и смертностью в единицу времени)
- α_1, α_2 — сила взаимодействия групп x, y между собой
- β_1, β_2 — сила внутривидовых взаимодействий каждой из групп (конкуренция за ресурсы)

Уравнение

$$\begin{cases} \dot{x} = x\left(r_1 - \beta_1 x - \alpha_2 y\right) \\ \dot{y} = y\left(r_2 - \beta_2 y - \alpha_1 x\right) \end{cases}$$

Решением уравнения будет называть функцию, удовлетворяющую следующей задаче Коши:

$$\begin{cases} \dot{x} = x\left(r_1 - \beta_1 x - \alpha_2 y\right) \\ \dot{y} = y\left(r_2 - \beta_2 y - \alpha_1 x\right) \\ x\left(0\right) = x_0 \quad y\left(0\right) = y_0 \end{cases}$$

Аналитическое исследование системы

Условие устойчивости

Для начала найдём равновесия модели:

$$\begin{cases} 0 = x\left(r_1 - \beta_1 x - \alpha_2 y\right) \\ 0 = y\left(r_2 - \beta_2 y - \alpha_1 x\right) \end{cases}$$

Решив систему, получаем следующие два равновесия:

Тривиальное

$x_t = y_t = 0$

Нетривиальные

$$x_1 = 0, \quad y_1 = \frac{r_2}{\beta_2}$$

$$x_2 = \frac{r_1}{\beta_1}, \quad y_2 = 0$$

$$x_3 = \frac{\alpha_1\beta_2 - \alpha_2 r_2}{\beta_1\beta_2 - \alpha_1\alpha_2}, \quad y_3 = \frac{\beta_1 r_2 - \alpha_1 r_1}{\beta_1\beta_2 - \alpha_1\alpha_2}$$

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

$$J - \lambda E = \begin{pmatrix} r_1 - \alpha_2 y - 2\beta_1 x - \lambda & -\alpha_2 x \\ -\alpha_1 y & r_2 - \alpha_1 x - 2\beta_2 y - \lambda \end{pmatrix}$$

Тривиальное равновесие

$$\det\left(J - \lambda E\right) \underset{x=x_t,\, y=y_t}{=} \left(r_1 - \lambda\right)\left(r_2 - \lambda\right) = 0 \implies \begin{cases} \lambda_1 = r_1 \\ \lambda_2 = r_2 \end{cases}$$

Так как $\mathrm{Im}\lambda_{1,2} = 0,\ \mathrm{Re}\lambda_1 > 0,\ \mathrm{Im}\lambda_2 > 0$ можем заключить, что тривиальное равновесие **неустойчиво** и является узлом.

Нетривиальные равновесия

x_1, y_1:

$$\det\left(J - \lambda E\right) \underset{x=x_1,\, y=y_1}{=} \left(r_1 - \frac{\alpha_2 r_2}{\beta_2} - \lambda\right)\left(r_2 - \frac{2\beta_2 r_2}{\beta_2} - \lambda\right) \implies \begin{cases} \lambda_1 = r_1 - \dfrac{\alpha_2 r_2}{\beta_2} \\ \lambda_2 = -r_2 \end{cases}$$

Имеем следующие случаи:

- $r_1\beta_2 > \alpha_2 r_2$
  - Неустойчивая (седло)

- $r_1\beta_2 < \alpha_2 r_2$
    - Устойчивая (узел)

x_2, y_2:

$$\det(J-\lambda E) \underset{x=x_2,\, y=y_2}{=} (-r_1-\lambda)\left(r_2 - \frac{\alpha_1 r_1}{\beta_1} - \lambda\right) \implies \begin{cases}\lambda_1 = r_2 - \dfrac{\alpha_1 r_1}{\beta_1} \\ \lambda_2 = -r_1\end{cases}$$

Имеем следующие случаи:

- $r_2\beta_1 > \alpha_1 r_1$
    - Неустойчивая (седло)
- $r_2\beta_1 < \alpha_1 r_1$
    - Устойчивая (узел)

x_3, y_3:

$$\det(J-\lambda E) \underset{x=x_3,\, y=y_3}{=} (r_1 - \alpha_2 y_3 - 2\beta_1 x_3 - \lambda)(r_2 - \alpha_1 x_3 - 2\beta_2 y_3 - \lambda) - \alpha_1\alpha_2 x_3 y_3 =$$

$$= (-\beta_1 x_3 - \lambda)(-\beta_2 y_3 - \lambda) - \alpha_1\alpha_2 x_3 y_3 = 0$$

$$\lambda^2 + (\beta_2 y_3 + \beta_1 x_3)\lambda + (\beta_1\beta_2 - \alpha_1\alpha_2)x_3 y_3 = 0$$

$$p = \beta_1 x_3 + \beta_2 y_3 > 0$$

$$q = (\beta_1\beta_2 - \alpha_1\alpha_2)x_3 y_3 = \frac{(\alpha_1\beta_2 - \alpha_2 r_2)(\beta_1 r_2 - \alpha_1 r_1)}{\beta_1\beta_2 - \alpha_1\alpha_2}$$

$$D = p^2 - 4q$$

На удивление, воспользуемся теоремой Виетта:

$\lambda_1 + \lambda_2 = -p < 0$

$\lambda_1 \cdot \lambda_2 = 1$

Имеем следующие случаи:

$q < 0$ (Седло):

$$\begin{cases}\lambda_1 \cdot \lambda_2 < 0 \\ \lambda_1 + \lambda_2 < 0 \\ D = p^2 - 4q > 0\end{cases} \implies \begin{matrix}\lambda_1, \lambda_2 \in R \\ \lambda_1 \cdot \lambda_2 < 0\end{matrix}$$

$q > 0, D \geq 0$ (Устойчивый узел):

$$\begin{cases}\lambda_1 \cdot \lambda_2 > 0 \\ \lambda_1 + \lambda_2 < 0 \\ D = p^2 - 4q > 0\end{cases} \implies \begin{matrix}\lambda_1, \lambda_2 \in R \\ \lambda_1 < 0,\ \lambda_2 < 0\end{matrix}$$

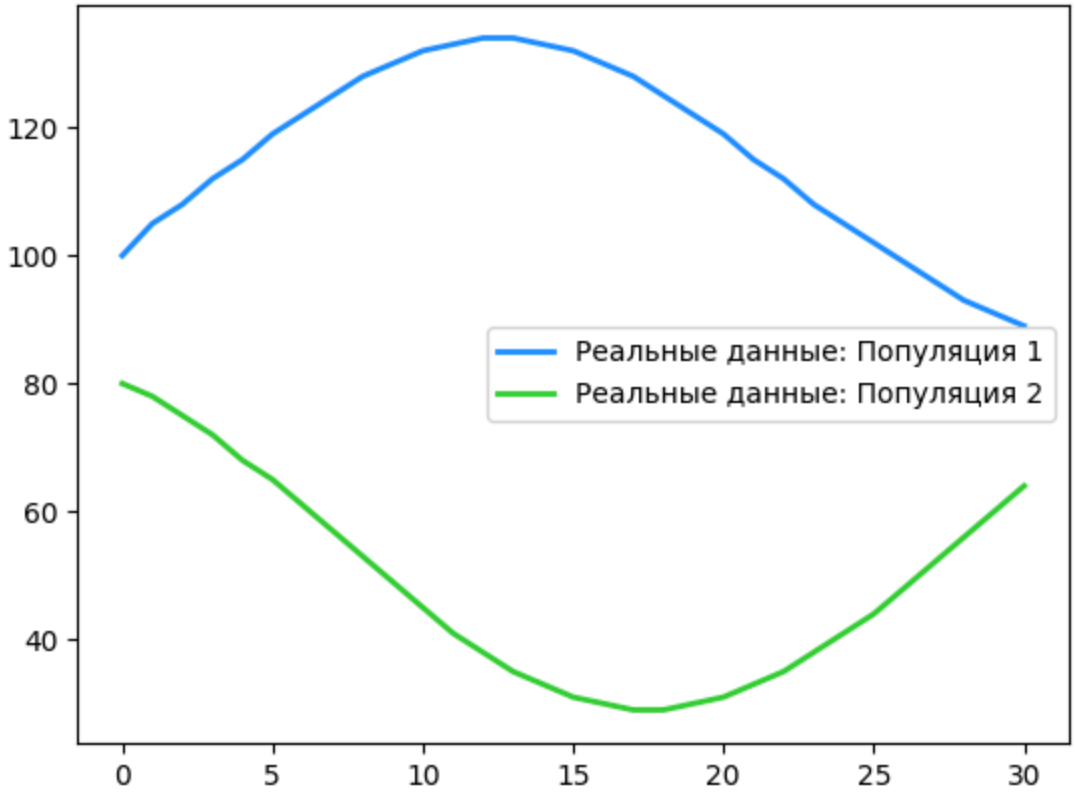
$q > 0, D < 0$ (Устойчивый фокус):

$$\lambda_{1,2} = \frac{-p \pm \sqrt{p^2 - 4q}\cdot i}{2}$$

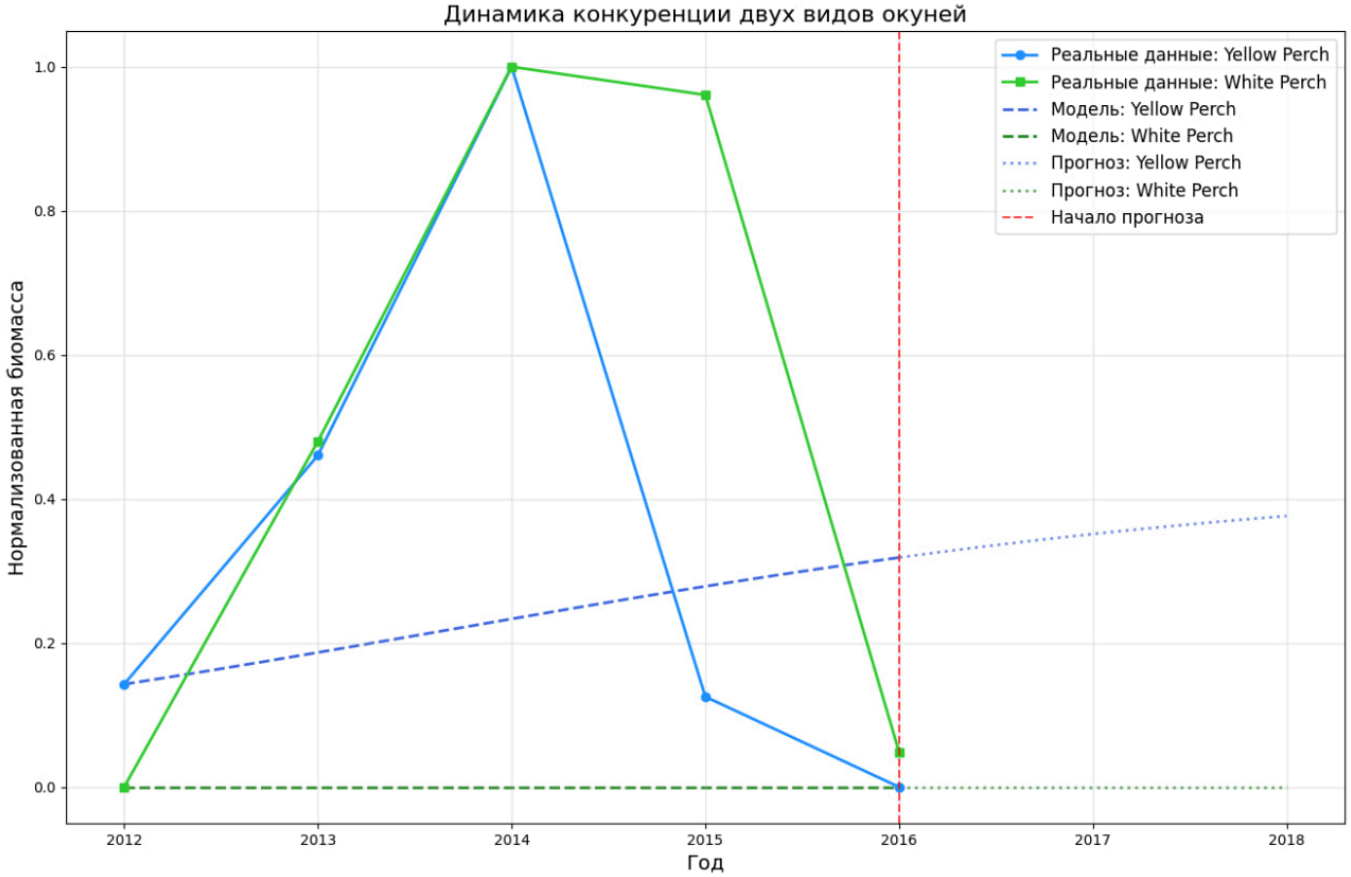
$$\mathrm{Re}\lambda_{1,2} < 0, \quad \mathrm{Im}\lambda_{1,2} \neq 0$$

Численные эксперименты

Были сгенерированы синтетические данные за 30 лет.



Пытался изначально работать с датасетом про рыб, но... Видимо, там мне сильно не везло с начальными условиями ибо выходило так:

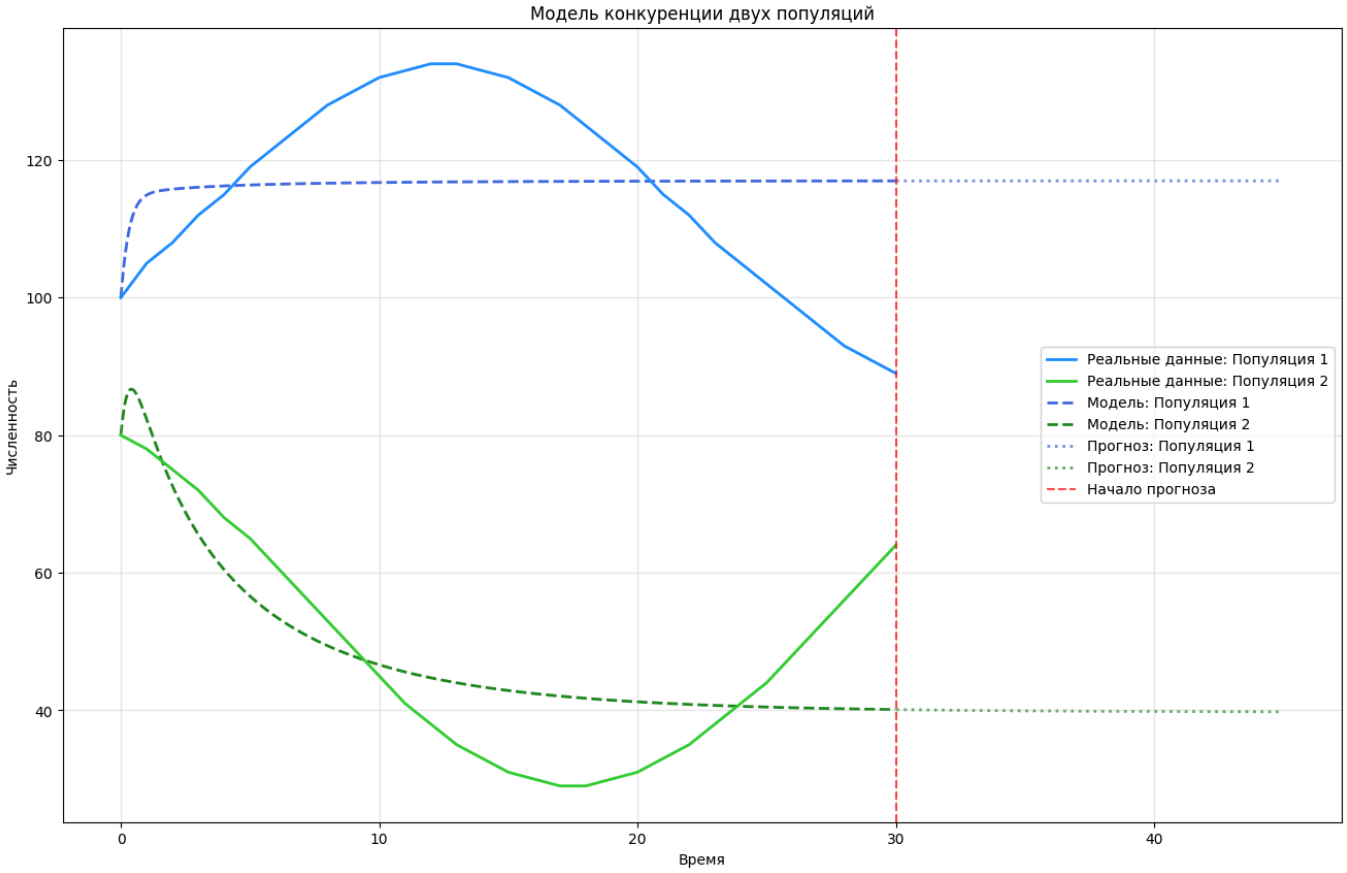


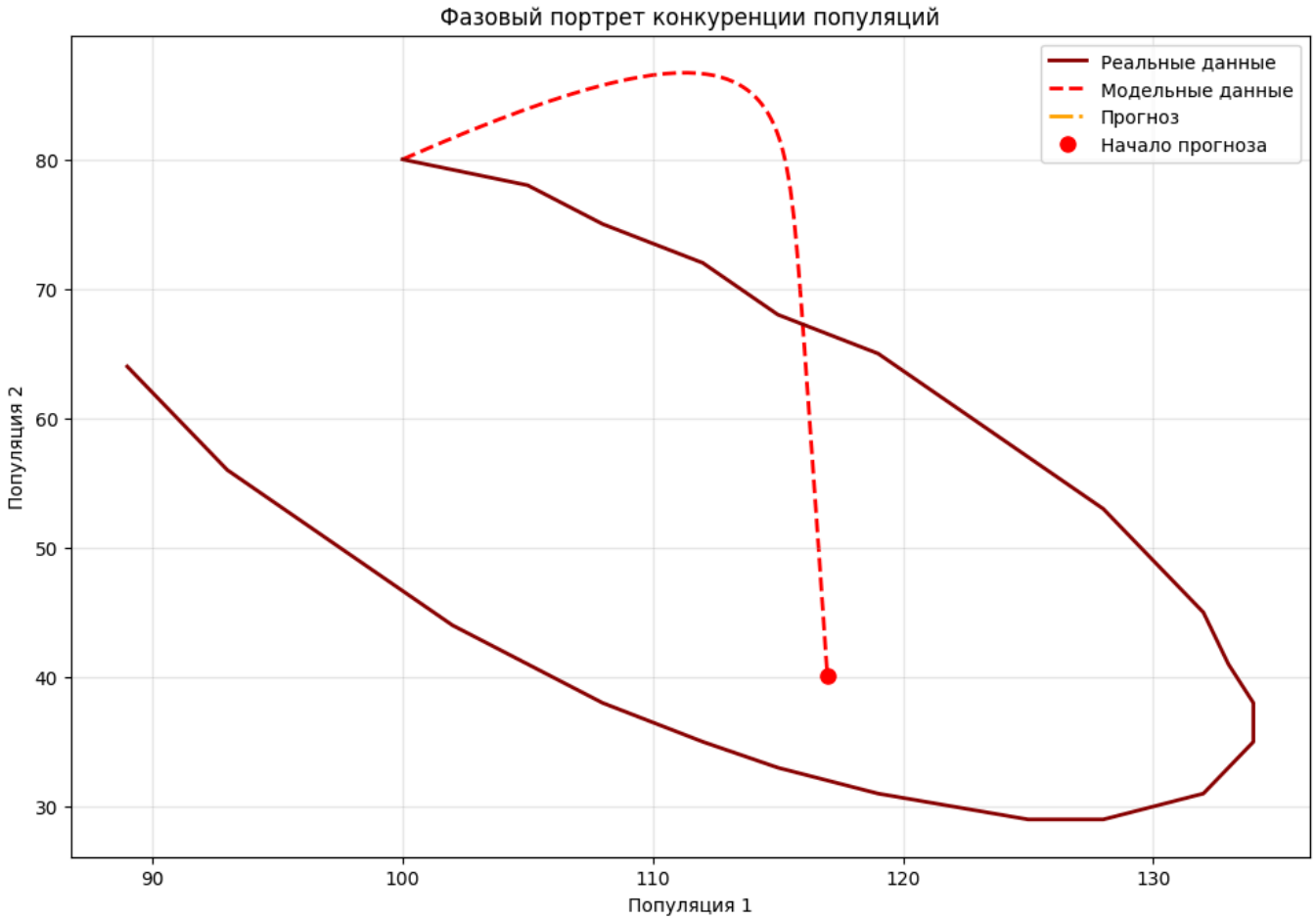
Основные расчёты

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. Динамика популяций представлена на рисунке 1, тогда как фазовый портрет на рисунке 2.





Заключение

В результате проведенного исследования была успешно применена модель конкуренции двух популяций для анализа конкуренции двух синтетических популяций. Аналитическое исследование системы позволило идентифицировать состояния равновесия, найдены условия устойчивости этих равновесий. 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.

Использованные данные

Неудачные данные: https://www.sciencebase.gov/catalog/item/5cdc66f9e4b0f324a7cefeea

Листинг кода

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import minimize
from scipy.integrate import solve_ivp
```

```python
# Загрузка данных
df = pd.read_csv("../data/d.csv")  # предполагая, что данные сохранены в этом файле
x_name, y_name, t_name = "population_1", "population_2", "year"

t_data = df[t_name].values
H_data = df[x_name].values
L_data = df[y_name].values

# Начальные условия из данных
y0 = [H_data[0], L_data[0]]

plt.plot(
    df[t_name],
    df[x_name],
    "dodgerblue",
    label="Реальные данные: Популяция 1",
    zorder=2,
    linewidth=2,
)
plt.plot(
    df[t_name],
    df[y_name],
    "limegreen",
    label="Реальные данные: Популяция 2",
    zorder=2,
    linewidth=2,
)
plt.legend()
plt.show()
def objective(params):
    y_pred = solve_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 solve_model(params, t, y0):
    r1, r2, a1, a2, b1, b2 = params

    def lotka_volterra(t, y):
        H, L = y
        dHdt = r1 * H - a1 * H * L - b1 * H * H
        dLdt = r2 * L - a2 * H * L - b2 * L * L
        return [dHdt, dLdt]
```

```
    solution = solve_ivp(lotka_volterra, [t[0], t[-1]], y0, t_eval=t,
method="RK45")
    return solution.y


# Подбор параметров
initial_guess = np.random.random(size=6)
print("Начальное приближение:", initial_guess)

bounds = [(0.001, 5), (0.001, 5), (0.001, 2), (0.001, 2), (0.001, 2), (0.001,
2)]
result = minimize(objective, initial_guess, method="L-BFGS-B", bounds=bounds)
print("Найденные параметры:", result.x)

r1, r2, a1, a2, b1, b2 = result.x


def lotka_volterra_model(t, y):
    H, L = y
    dHdt = r1 * H - a1 * H * L - b1 * H * H
    dLdt = r2 * L - a2 * H * L - b2 * L * L
    return [dHdt, dLdt]


def runge_kutta_4th_order(f, y0, t_span, n_steps):
    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]))
        k2 = h * np.array(f(t[i] + h / 2, y[:, i] + k1 / 2))
        k3 = h * np.array(f(t[i] + h / 2, y[:, i] + k2 / 2))
        k4 = h * np.array(f(t[i] + h, y[:, i] + k3))
        y[:, i + 1] = y[:, i] + (k1 + 2 * k2 + 2 * k3 + k4) / 6

    return t, y


# Прогнозирование
H0 = df[x_name].iloc[0]
L0 = df[y_name].iloc[0]
n_steps = 1000
y0 = [H0, L0]
```

```
forecast_period = 1.5
t_max_original = df[t_name].iloc[-1]
t_min_original = df[t_name].iloc[0]
t_range_original = t_max_original - t_min_original

# Временной диапазон для прогноза
t_span_forecast = (t_min_original, t_min_original + t_range_original *
forecast_period)
n_steps_forecast = int(n_steps * forecast_period)

# Вычисляем прогноз
t_forecast, y_forecast = runge_kutta_4th_order(
    lotka_volterra_model, y0, t_span_forecast, n_steps_forecast
)

# Разделяем данные на исторические и прогнозные
historical_mask = t_forecast <= t_max_original
forecast_mask = t_forecast > t_max_original

t_historical = t_forecast[historical_mask]
t_forecast_only = t_forecast[forecast_mask]
H_historical = y_forecast[0][historical_mask]
H_forecast = y_forecast[0][forecast_mask]
L_historical = y_forecast[1][historical_mask]
L_forecast = y_forecast[1][forecast_mask]

# Визуализация с прогнозом
plt.figure(figsize=(16, 10))

# Исторические данные
plt.plot(
    df[t_name],
    df[x_name],
    "dodgerblue",
    label="Реальные данные: Популяция 1",
    zorder=2,
    linewidth=2,
)
plt.plot(
    df[t_name],
    df[y_name],
    "limegreen",
    label="Реальные данные: Популяция 2",
    zorder=2,
    linewidth=2,
```

```
)

# Модель на историческом периоде
plt.plot(
    t_historical,
    H_historical,
    "royalblue",
    linestyle="--",
    label="Модель: Популяция 1",
    zorder=1,
    linewidth=2,
)
plt.plot(
    t_historical,
    L_historical,
    "forestgreen",
    linestyle="--",
    label="Модель: Популяция 2",
    zorder=1,
    linewidth=2,
)

# Прогноз
plt.plot(
    t_forecast_only,
    H_forecast,
    "royalblue",
    linestyle=":",
    alpha=0.7,
    label="Прогноз: Популяция 1",
    zorder=1,
    linewidth=2,
)
plt.plot(
    t_forecast_only,
    L_forecast,
    "forestgreen",
    linestyle=":",
    alpha=0.7,
    label="Прогноз: Популяция 2",
    zorder=1,
    linewidth=2,
)

# Вертикальная линия разделяющая историю и прогноз
plt.axvline(
```

```
    x=t_max_original, color="red", linestyle="--", alpha=0.7, label="Начало
прогноза"
)

plt.legend()
plt.xlabel("Время")
plt.ylabel("Численность")
plt.title("Модель конкуренции двух популяций")
plt.grid(True, alpha=0.3)
plt.show()

# Фазовый портрет с прогнозом
plt.figure(figsize=(12, 8))

plt.plot(
    df[x_name], df[y_name], "darkred", label="Реальные данные", zorder=2,
linewidth=2
)
plt.plot(
    H_historical,
    L_historical,
    "red",
    linestyle="--",
    label="Модельные данные",
    zorder=1,
    linewidth=2,
)
plt.plot(
    H_forecast,
    L_forecast,
    "orange",
    linestyle="-.",
    label="Прогноз",
    zorder=1,
    linewidth=2,
)

# Точка начала прогноза
plt.plot(
    H_historical[-1], L_historical[-1], "ro", markersize=8, label="Начало
прогноза"
)

plt.legend()
plt.xlabel("Популяция 1")
plt.ylabel("Популяция 2")
```

```
plt.title("Фазовый портрет конкуренции популяций")
plt.grid(True, alpha=0.3)
plt.show()

# Вывод найденных параметров для интерпретации
print("\nИнтерпретация параметров:")
print(f"r1 = {r1:.4f} - коэффициент роста популяции 1")
print(f"r2 = {r2:.4f} - коэффициент роста популяции 2")
print(f"a1 = {a1:.4f} - влияние популяции 2 на популяцию 1")
print(f"a2 = {a2:.4f} - влияние популяции 1 на популяцию 2")
print(f"b1 = {b1:.4f} - внутривидовая конкуренция популяции 1")
print(f"b2 = {b2:.4f} - внутривидовая конкуренция популяции 2")
```