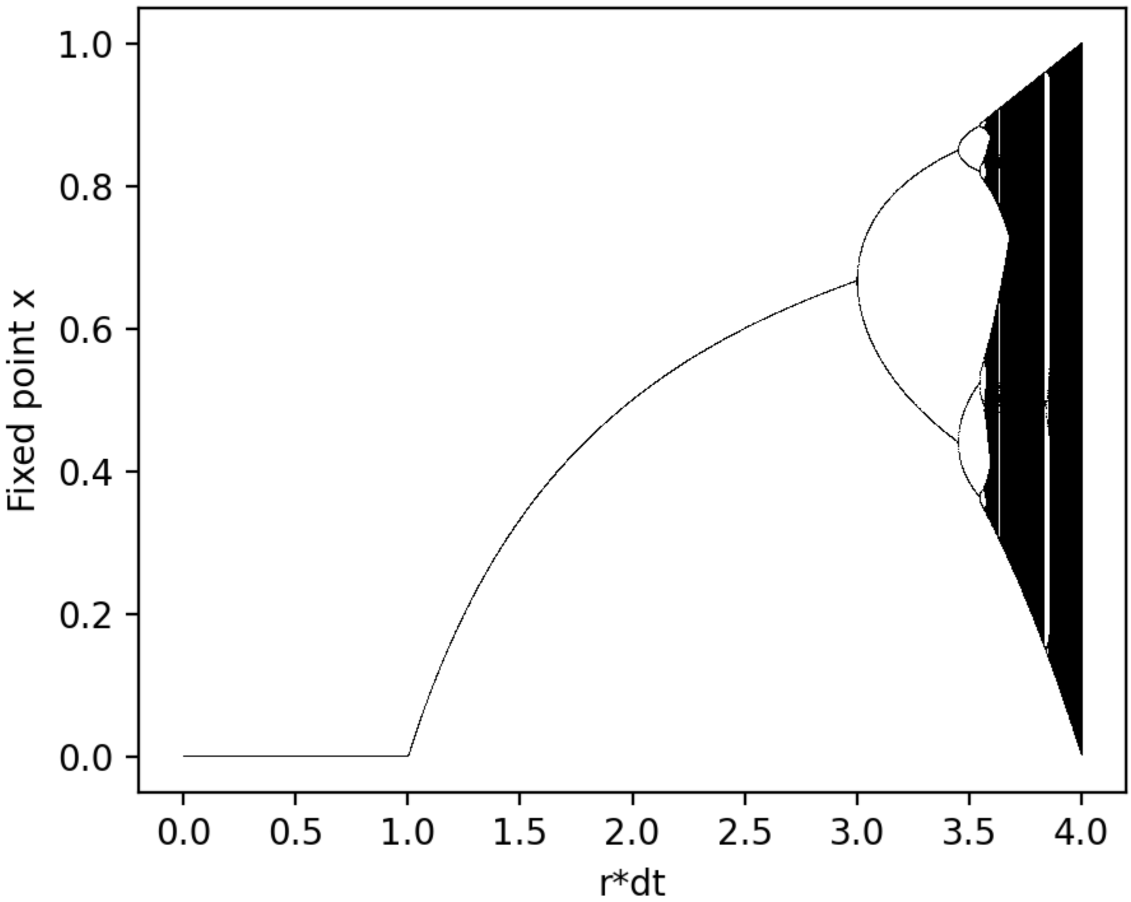
Lab 1 – modele populacyjne - równania różniczkowe i ich układy

Arkadiusz Kurnik, Jan Cichoń

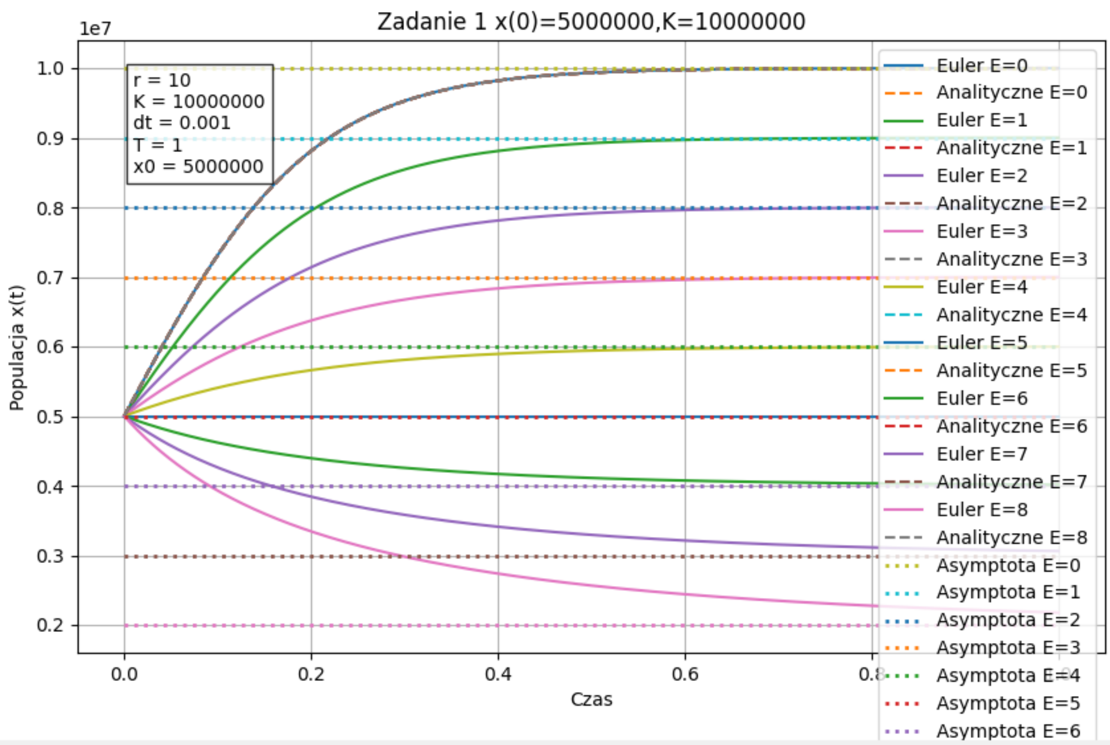
Zadanie 1:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

## Zadanie 1 ##
print("ZADANIE 1\n")

def euler_method(x0, r, K, E, dt, T):
    N = int(T / dt)
    t_vals = np.linspace(0, T, N)
    x_vals = np.zeros(N)
    x_vals[0] = x0

    for i in range(1, N):
        x_vals[i] = x_vals[i-1] + dt * (r * x_vals[i-1] * (1 - x_vals[i-1] / K) - E * x_vals[i-1])

    return t_vals, x_vals

def analytical_solution(x0, r, K, t_vals):
    return (K * x0 * np.exp(r * t_vals)) / (K - x0 + x0 * np.exp(r * t_vals))

# Parametry
r = 10
K = 1e7
dt = 0.001
T = 1
x0 = 5e6

# Obliczenia
E_values = range(0, 9)
plt.figure(figsize=(10, 6))

for E in E_values:
    t_vals, x_vals = euler_method(x0, r, K, E, dt, T)
    x_analytic = analytical_solution(x0, r, K, t_vals)
    plt.plot(t_vals, x_vals, label=f'Euler E={E}', linewidth=1.5)
    plt.plot(t_vals, x_analytic, linestyle='dashed', linewidth=1.5, label=f'Analityczne E={E}')
```

```python
# Wykres 1
for E in range(0, 9):
    x_fixed_t = K * (1 - E / r) * np.ones_like(t_vals)
    plt.plot(t_vals, x_fixed_t, linestyle='dotted', linewidth=2, label=f'Asymptota E={E}')

plt.text(0.01 * T, 0.85 * K, f"r = {r}\nK = {K:.0f}\ndt = {dt}\nT = {T}\nx0 = {x0:.0f}",
         fontsize=10, bbox=dict(facecolor='white', alpha=0.8))

plt.xlabel('Czas')
plt.ylabel('Populacja x(t)')
plt.title('Zadanie 1 x(0)=5000000,K=10000000')
plt.legend()
plt.grid()

def calc_bifurk():
  min_r = 0
  max_r = 4
  step_r = 0.001
  max_iterations = 10000
  skip_iterations = 500
  max_counter = int((max_iterations - skip_iterations) * (max_r - min_r) / step_r)
  result_x = np.zeros(max_counter)
  result_r = np.zeros(max_counter)
  i = 0
  for r in np.arange(min_r, max_r, step_r):
    x = 0.1
    for it in range(max_iterations):
      x = r * x * (1-x)
      if it > skip_iterations:
        result_x[i] = x
        result_r[i] = r
        i += 1
  result_x = result_x[result_r != 0].copy()
  result_r = result_r[result_r != 0].copy()
  return result_x, result_r

result_x, result_r = calc_bifurk()

# Wykres 2
plt.figure(figsize=(5, 4), dpi=200)
plt.plot(result_r, result_x, ",", color='k')
plt.xlabel('r*dt')
plt.ylabel('Fixed point x')
plt.show()
```



Fixed point x
r*dt



Zadanie 1 x(0)=5000000,K=10000000
1e7
r = 10
K = 10000000
dt = 0.001
T = 1
x0 = 5000000
Populacja x(t)
Czas
Euler E=0
Analityczne E=0
Euler E=1
Analityczne E=1
Euler E=2
Analityczne E=2
Euler E=3
Analityczne E=3
Euler E=4
Analityczne E=4
Euler E=5
Analityczne E=5
Euler E=6
Analityczne E=6
Euler E=7
Analityczne E=7
Euler E=8
Analityczne E=8
Asymptota E=0
Asymptota E=1
Asymptota E=2
Asymptota E=3
Asymptota E=4
Asymptota E=5
Asymptota E=6

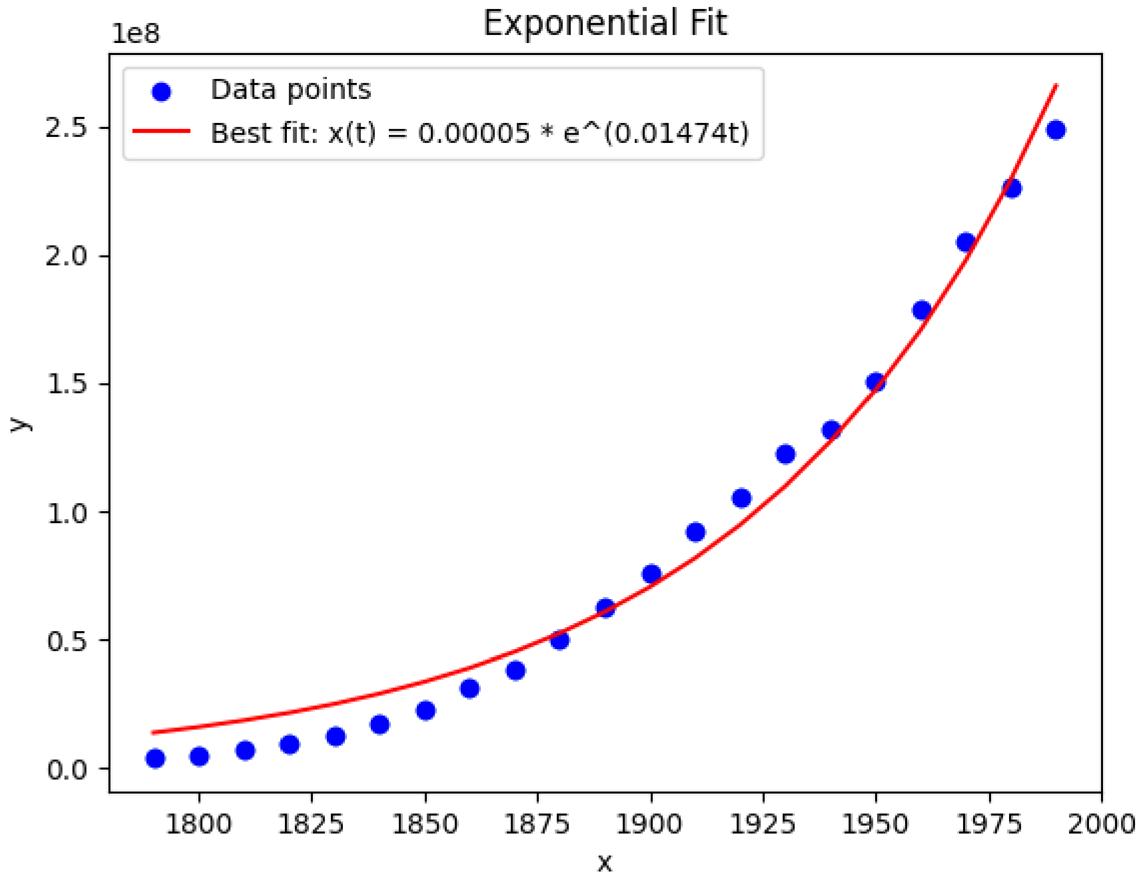
Zadanie 2:

```python
## Zadanie 2 ##
print("ZADANIE 2\n")

P = 0.01    # Roczna wpłata
r = 0.05    # Roczna stopa oprocentowania (5%)
target = 10**6  # Cel: 1 milion

# Obliczenie
T = np.log((target * r / P) + 1) / r
print(f"Czas potrzebny na zgromadzenie 1 miliona: {T:.2f} lat\n")
```

```
ZADANIE 2

Czas potrzebny na zgromadzenie 1 miliona: 308.50 lat
```

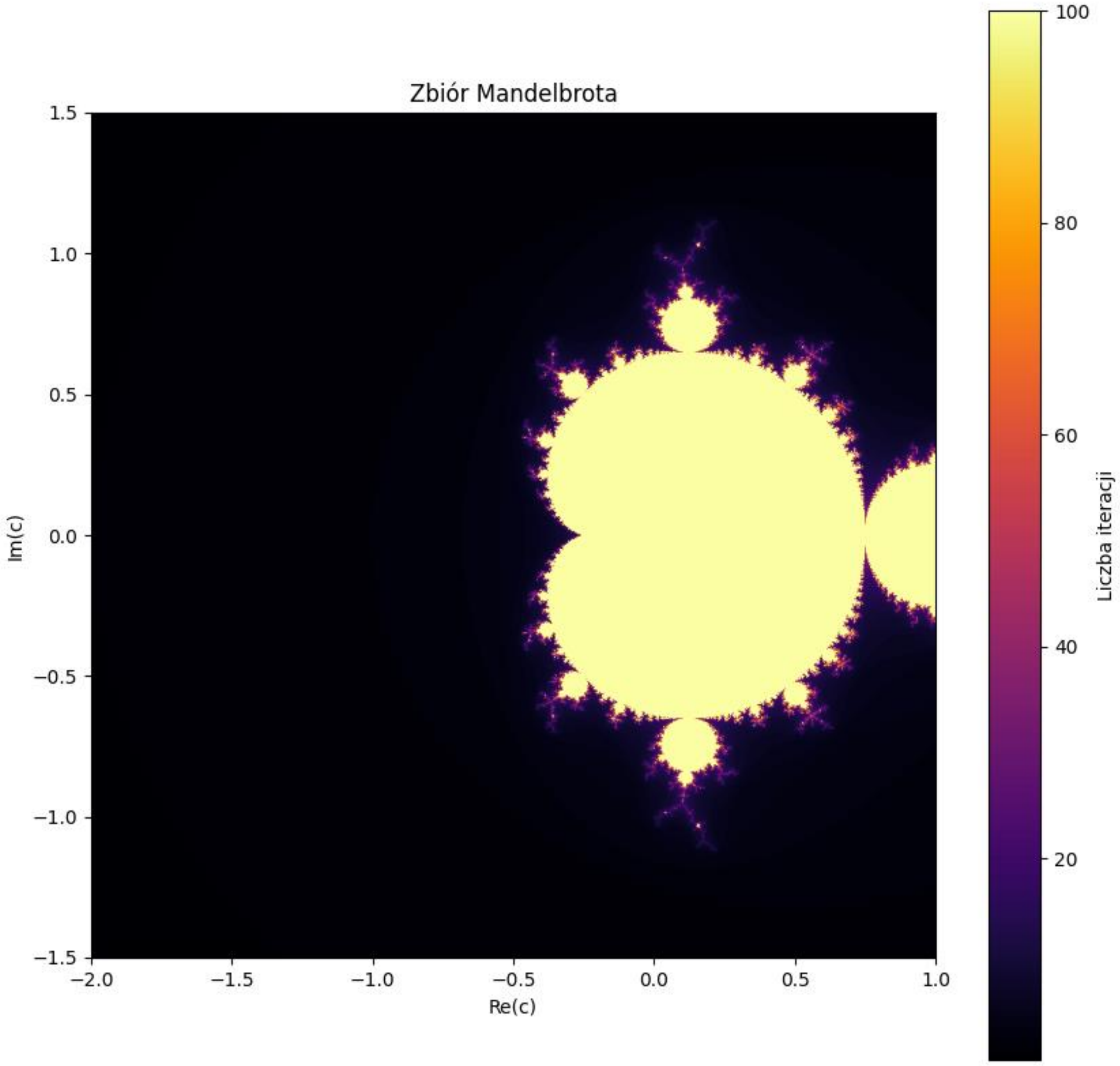
Zadanie 3:

```python
## Zadanie 3 ##
print("ZADANIE 3\n")

x = np.array(range(1790, 2000, 10))
y = np.array([3900000, 5300000, 7200000, 9600000, 12900000, 17100000, 23100000, 31400000, 38600000, 50200000, 62900000,
              76000000, 92000000, 105700000, 122800000, 131700000, 150700000, 179000000, 205000000, 226500000, 248700000])

plt.scatter(x, y)
plt.xlabel("Year")
plt.ylabel("Population")
plt.grid(True)


def exponential_func(t, x_0, r):
    return x_0*np.exp(r*t)

popt, _ = curve_fit(exponential_func, x, y, p0=(1, 1e-8))

y_fit = exponential_func(x, *popt)

plt.scatter(x, y, label="Data points", color="blue")
plt.plot(x, y_fit, label=f"Best fit: x(t) = {popt[0]:.5f} * e^({popt[1]:.5f}t)", color="red")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Exponential Fit")
plt.legend()
plt.grid()
plt.show()

print(f"Dopasowane równanie: x(t) = {popt[0]:.5f} * e^({popt[1]:.5f}*t)")
```


Exponential Fit
1e8
Data points
Best fit: x(t) = 0.00005 * e^(0.01474t)
2.5
2.0
1.5
1.0
0.5
0.0
y
1800
1825
1850
1875
1900
1925
1950
1975
2000
x

Zadanie 4:

```python
## Zadanie 4 ##
print("ZADANIE 4\n")

def mandelbrot(c, max_iter):
    z = 0
    for n in range(max_iter):
        if abs(z) > 2:
            return n
        z = z**2 - c
    return max_iter

def mandelbrot_set(xmin, xmax, ymin, ymax, width, height, max_iter):
    x = np.linspace(xmin, xmax, width)
    y = np.linspace(ymin, ymax, height)
    fractal = np.zeros((height, width))

    for i in range(height):
        for j in range(width):
            fractal[i, j] = mandelbrot(complex(x[j], y[i]), max_iter)

    return fractal

xmin, xmax, ymin, ymax = -2, 1, -1.5, 1.5
width, height = 800, 800
max_iter = 100

fractal = mandelbrot_set(xmin, xmax, ymin, ymax, width, height, max_iter)

# Wykres
plt.figure(figsize=(10, 10))
plt.imshow(fractal, extent=(xmin, xmax, ymin, ymax), cmap='inferno')
plt.colorbar(label='Liczba iteracji')
plt.title('Zbiór Mandelbrota')
plt.xlabel('Re(c)')
plt.ylabel('Im(c)')
plt.show()
```


Zbiór Mandelbrota
Im(c)
Re(c)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
Liczba iteracji
100
80
60
40
20