**Sprawozdanie MSD - Lista 4**

Tymon Pabian

7 czerwca 2022

**1 Wstęp**

Równanie logistyczne to pierwszy model populacyjny nieliniowy. Został on zaproponowany przez matematyka z Belgii - Pierre’a F. Verhulsta. Został on stworzony jako następnik liniowego modelu Malthusa, skupiając się głównie na tym, że istnieją naturalne procesy, które hamują przyrost liczebności populacji. W ten sposób matematyk stworzył swój model opisujący rozrost populacji nazwany później od jego nazwiska.

Gdzie K opisuje pojemność populacji i r przyrost naturalny.

Moim zadaniem było przy użyciu biblioteki sympy przedstawić wykresy dla tego modelu oraz wyliczyć błędy dla tego modelu przy użyciu biblioteki scipy.

**2 Model**

import numpy as np

import matplotlib.pyplot as plt

from sympy import Function, dsolve, Eq, symbols, plot

from scipy.integrate import odeint

from os import system

plt.style.use('ggplot')

system('cls')

# Dane:

t = symbols('t')            # czas

N = Function('N')           # funkcja N

r = 0.75                    # współczynnik reprodukcji

K = 2                       # pojemność środowiska

N\_0 = [K/40, 3/4\*K, K+2]    # wielkości początkowe populacji

"""# Model Verhulsta: dN\_dt = rN(1-N/K)"""

verhulst = Eq(N(t).diff(t), r\*N(t)\*(1- N(t)/K))

# Tworzenie wykresu dla 3 róznych wielkości początkowych (sympy)

sol1 = dsolve(verhulst, ics={N(0): N\_0[0]})

sol2 = dsolve(verhulst, ics={N(0): N\_0[1]})

sol3 = dsolve(verhulst, ics={N(0): N\_0[2]})

p1 = plot(sol1.rhs, sol2.rhs, sol3.rhs, (t, 0, 12), yaxis=(0,4), legend=True, show=False, title='sympy: Model Verhulsta', xlabel='czas $t$', ylabel ='liczebnosc populacji')

p1[0].line\_color = 'red'

p1[1].line\_color = 'blue'

p1[2].line\_color = 'purple'

p1.show()

# Tworzenie tablic z wynikami (get\_points() daje dla 65 kroków)

time1, so1 = p1[0].get\_points()

time2, so2 = p1[1].get\_points()

time3, so3 = p1[2].get\_points()

def s\_ver(x, t):

    """Model Verhulsta dla scipy"""

    return x\*r\*(1-x/K)

T1 = len(time1)   # Liczba kroków

T2 = len(time2)

T3 = len(time3)

# Tworzenie wykresu dla 3 róznych wielkości początkowych (scipy)

time1 = np.linspace(0, 12, T1)

time2 = np.linspace(0, 12, T2)

time3 = np.linspace(0, 12, T3)

X = odeint(s\_ver, np.asarray(N\_0[0]), time1, full\_output=True)[0]

Y = odeint(s\_ver, np.asarray(N\_0[1]), time2, full\_output=True)[0]

Z = odeint(s\_ver, np.asarray(N\_0[2]), time3, full\_output=True)[0]

plt.plot(time1, X, label='N(0) = K/40')

plt.plot(time2, Y, label='N(0) = 3/4\*K')

plt.plot(time3, Z, label='N(0) = K+2')

plt.legend()

plt.title('scipy: Model Verhulsta')

plt.xlabel('czas $t$')

plt.ylabel('liczebnosc populacji')

plt.show()

# Obliczanie błędów

MAE\_1, MAE\_2, MAE\_3 = 0, 0, 0

MSE\_1, MSE\_2, MSE\_3 = 0, 0 , 0

error1, error2, error3 = [], [], []

for i in range(T1):

    MAE\_1 += abs(X[i] - so1[i])/T1

    MSE\_1 += ((X[i] - so1[i])\*\*2)/T1

    error1.append((X[i] - so1[i])/T1)

for i in range(T2):

    MAE\_2 += abs(Y[i] - so2[i])/T2

    MSE\_2 += ((Y[i] - so2[i])\*\*2)/T2

    error2.append((Y[i] - so2[i])/T2)

for i in range(T3):

    MAE\_3 += abs(Z[i] - so3[i])/T3

    MSE\_3 += ((Z[i] - so3[i])\*\*2)/T3

    error3.append((Z[i] - so3[i])/T3)

print("Bledy bezwzgledne:")

print(f"K/40  (krok: {12/T1}): {MAE\_1[0]}\n3/4\*K (krok: {12/T2}): {MAE\_2[0]}\nK+2   (krok: {12/T3}): {MAE\_3[0]}")

print("\nBledy kwadratowe:")

print(f"K/40  (krok: {12/T1}): {MSE\_1[0]}\n3/4\*K (krok: {12/T2}): {MSE\_2[0]}\nK+2   (krok: {12/T3}): {MSE\_3[0]}\n")

# Tworzenie wykresu dla błędów

plt.plot(time1, error1, label='N(0) = K/40')

plt.plot(time2, error2, label='N(0) = 3/4\*K')

plt.plot(time3, error3, label='N(0) = K+2')

plt.legend()

plt.title(f'bledy')

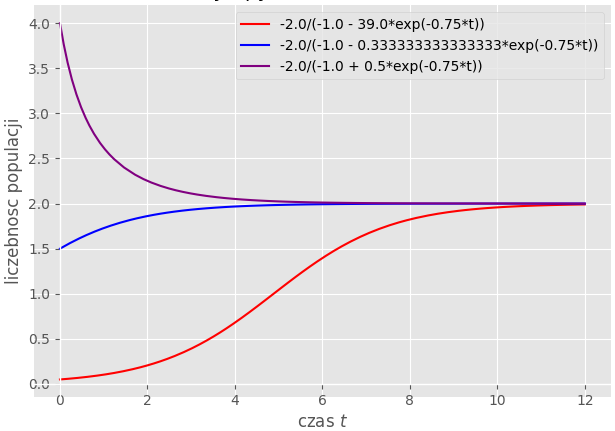
plt.xlabel('czas $t$')

plt.ylabel('scipy - sympy')

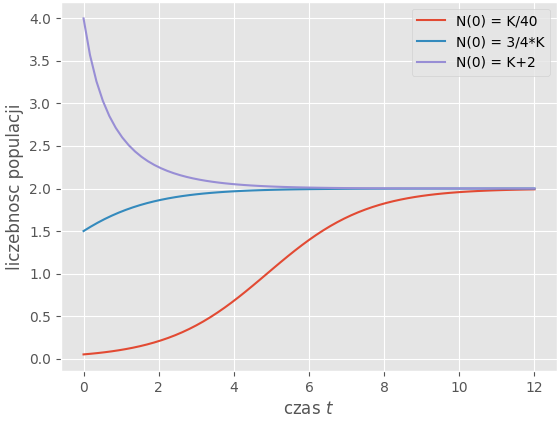
plt.show()

**3 Wyniki (**dla K =2 i r = 0.75)

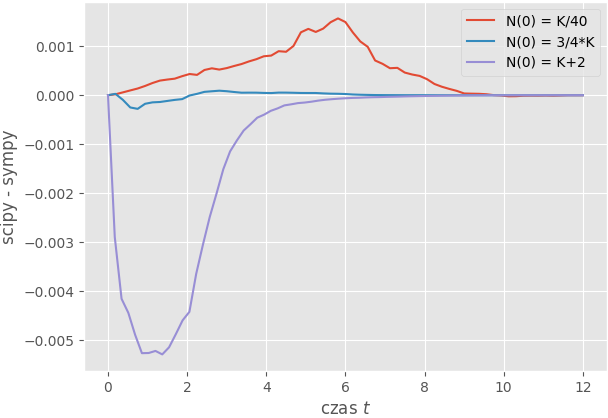
**Wykres sympy dla Modelu Verhulsta**



**Wykres scipy dla Modelu Verhulsta**



**Wykres błędów scipy - sympy  
  
Błędy bezwzględne:**N(0) = K/40 (krok: 0.18461538461538463): 0.030247643306392784  
N(0) = 3/4\*K (krok: 0.18461538461538463): 0.002531659976007831  
N(0) = K+2 (krok: 0.16901408450704225): 0.07569501464989956



**Błędy kwadratowe:**N(0) = K/40 (krok: 0.18461538461538463): 0.0018064020442492457  
N(0) = 3/4\*K (krok: 0.18461538461538463): 2.026484396338138e-05  
N(0) = K+2 (krok: 0.16901408450704225): 0.021988499192004254

**4 Wnioski**

Wyniki oraz wykresy nie różnią się od przewidywań. Wykresy stworzone przez scipy oraz sympy są prawie identyczne – jest to zapewne spowodowane małym skomplikowaniem modelu. Największą różnicą jest pojedynczy skok o 5\*. Ciekawą obserwacją jest, że moduł sympy podczas tworzenia wykresu sam decyduje o wielkości skoku (dt), co trochę utrudniało automatyzację wyliczania błędów.

Obserwacja samego modelu prowadzi do wniosku, że populacje zawsze dążą do pojemności populacji, która pojawia się przez np. konkurencji wewnątrzgatunkowej. Równanie logiczne przez to lepiej opisuje przyrost populacyjny, gdyż w rzeczywistości możemy zauważyć, że istnieje coś takiego jak przeludnienie, które wpływa negatywnie na populacje.