# Estymacja parametrów modelu

## Modelowanie Matematyczne 2023/2024 semestr zimowy

Zadanie Projektowe nr 2

Autor: Mikołaj Karbowski
Prowadzący: dr inż. Paweł Mazurek

Politechnika Warszawska,
Wydział Matematyki i Nauk Informacyjnych,
Warszawa, 22 grudnia 2023

Oświadczam, że niniejsza praca, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu Modelowanie Matematyczne została wykonana przeze mnie samodzielnie

# Spis treści


**1 Wprowadzenie** **3**
- 1.1 Przedstawienie zadania . . . 3
- 1.2 Przedstawienie użytych metod i funkcji w projekcie . . . 3
  - 1.2.1 fminsearch . . . 3
  - 1.2.2 interp1 . . . 4
  - 1.2.3 ode45 . . . 4
  - 1.2.4 Jawna metoda Eulera . . . 4
  - 1.2.5 Niejawna metoda Eulera . . . 5
  - 1.2.6 Metoda Adamsa-Bashfortha drugiego rzędu . . . 5

**2 Metodyka i wyniki doświadczeń** **6**
- 2.1 Zadanie 1 . . . 6
- 2.2 Zadanie 2 . . . 6
  - 2.2.1 Niejawna metoda Eulera . . . 6
  - 2.2.2 Metoda Adamsa-Bashfortha drugiego rzedu . . . 7
- 2.3 Zadanie 3 . . . 7
- 2.4 Zadanie 4 . . . 7
  - 2.4.1 Dla jakiej pary wartości $x, y > 0$ układ osiąg stan równowagi (tzn. brak zmiany $x(t)$ i $y(t)$ w czasie)? . . . 7
  - 2.4.2 Jak można interpretować wartości parametrów $r_x, r_{xy}, r_{xx}, r_y, r_{yx}, r_{yy}$? . . . 8
- 2.5 Zadanie 5 . . . 8

**3 Dyskusja wyników eksperymentów numerycznych** **9**
- 3.1 Zadanie 1 . . . 9
- 3.2 Zadanie 2 . . . 11
  - 3.2.1 Niejawna metoda Eulera . . . 11
  - 3.2.2 Metoda Adamsa-Bashfortha . . . 12
- 3.3 Zadanie 3 . . . 13
- 3.4 Zadanie 4 . . . 15
- 3.5 Zadanie 5 . . . 16

**4 Wnioski** **19**

**5 Listing** **19**

## Lista symboli matematycznych

**J** - współczynnik dopasowania modelu

$t$ - wektor czasu

$\Delta t$ - krok całkowy

$\hat{x}$ - wektor estymacji populacji ofiar

$\hat{y}$ - wektor estymacji populacji drapieżników

$\tilde{x}$ - wektor wyników pomiarów populacji ofiar

$\tilde{y}$ - wektor wyników pomiarów populacji drapieżników

$f_x(x, y)$ - wartość pochodnej funkcji $x$

$f_y(x, y)$ - warotść pochodnej funkcji $y$

$r_x, rxy, ..., ryy$ - parametry modelu

# 1 Wprowadzenie

W tym projekcie będziemy zajmować się równaniami Lotki-Volterry, które służą do modelowania liczebności populacji dwóch gatunków, między którymi występuje zależność drapieżnik-ofiara. Postaramy się dopasować model do dostarczonych danych, tak aby zminimalizować wskaźnik dopasowania (2).
Model Lotki-Volterry opisany jest przez następujący układ równań różniczkowych:

$$\begin{cases} \frac{dx}{dt} = r_x x(t) + r_{xy} x(t) y(t) + r_{xx} x^2(t) \\ \frac{dy}{dt} = r_y y(t) + r_{yx} x(t) y(t) + r_{yy} y^2(t) \end{cases} \tag{1}$$

Gdzie:
$x$ to liczebność populacji gatunku ofiary,
$y$ - liczebność populacji gatunku drapieżnika,
$t$ - czas,
$r_x, r_y, r_{xx}, r_{xy}, r_{yx}, r_{yy} \in \mathbb{R}$ to parametry modelu.

## 1.1 Przedstawienie zadania

Zadanie polega na znalezieniu takich współczynników w układzie równań (1), aby wyznaczone rozwiązania $x$ i $y$ były jak najlepiej dopasowane do zadanych danych.
Jako wskaźnik dopasowana modelu do danych posłużymy się funkcją:

$$\mathbf{J} = \mathbf{J}_x + \mathbf{J}_y \tag{2}$$

Gdzie:
$\mathbf{J}_x = \sum_{i=1}^{N} (\hat{x}_n - \tilde{x}_n)^2$, $\mathbf{J}_y = \sum_{i=1}^{N} (\hat{y}_n - \tilde{y}_n)^2$,
$\hat{y}, \hat{x}$, to estymaty $y(t)$ i $x(t)$ w chwilatch $t_1 ... t_N$
$\tilde{y}, \tilde{x}$, to dokładne wartości $y(t)$ i $x(t)$ w chwilatch $t_1 ... t_N$
W zadaniach 1-3 będziemy pracować na syntetycznych danych dostarczonymi w pliku dane22.csv, w zadaniu 5 postaramy się dopasować model do danych zebranych w rzeczywistych pomiarach, a konkretnie populacji zajęcy amerykańskich i rysi kanadyjskich w pewnym obszarze Kanady, dane te znajdują się w pliku HudsonBay.csv

## 1.2 Przedstawienie użytych metod i funkcji w projekcie

### 1.2.1 fminsearch

Jest to funkcja, której bedziemy używać najczęściej w naszym projekcie. Znajduje się ona w matlabowym dodatku Optimization Toolbox.
Posłuży nam ona do znalezienia współczynników w równaniu (1), które jak najlepiej dopasowują model do naszych danych, minimalizując funkcję błędu dopasowania (2).
Funkcja ta bazuje na algorytmie bezgradientowym służącym do minimalizacji funkcji wielu zmiennych. Algorytm ten opiera się na heurystyce, nie wymaga informacji o pochodnych funkcji celu i stara się znaleźć minimum lokalne.
Składnia funkcji wygląda następująco:
[x, fval, exitflag, output] = fminsearch(fun, x0, options)
Parametry:

- fun: Funkcja celu do zminimalizowania.
- x0: Punkt początkowy, od którego algorytm zaczyna poszukiwania minimum.
- options (opcjonalne): Struktura zawierająca ustawienia algorytmu.

Wyjścia:

- x: Znaleziony punkt minimalny.
- fval: Wartość funkcji celu w punkcie minimalnym.
- exitflag: Kod wyjścia informujący o tym, jak zakończył się algorytm.
- output: Struktura zawierająca dodatkowe informacje dotyczące przebiegu algorytmu.

### 1.2.2 interp1

Funkcja interp1 w MATLAB służy do interpolacji jednowymiarowej. Interpolacja to proces estymowania wartości między dostępnymi punktami danych. Funkcja ta jest użyteczna, jeśli chcemy uzyskać wartości dla punktów znajdujących się między danymi, a nie są one bezpośrednio dostępne.
Składnia funkcji:
Vq = interp1(X, V, Xq, method)
Parametry:

- X: Wektor zawierający współrzędne punktów danych.
- V: Wektor zawierający wartości odpowiadające współrzędnym w wektorze X.
- Xq: Wektor zawierający współrzędne punktów, dla których chcemy uzyskać interpolowane wartości.
- method (opcjonalne): Metoda interpolacji. Może przyjmować wartości, takie jak 'linear', 'nearest', 'spline', itp.

Wyjście:

- Vq: Wektor zawierający interpolowane wartości dla współrzędnych zawartych w Xq.

### 1.2.3 ode45

Aby rozwiązać układ równań różniczkowych (1) dla konkretnych współczynników w równaniach, będziemy musieli skorzystać z metod numerycznych. Jedną z nich będzie gotowa funkcja ode45. Jest to bardzo skuteczne narzędzie do rozwiązywania równań różniczkowych za pomocą adaptacyjnej metody Rungego-Kutty czwartego i piątego rzędu. Algorytm ten dostosowuje kroki czasowe w trakcie obliczeń, co pozwala na efektywne radzenie sobie z różnymi obszarami czasowymi i rodzajami równań.
Składnia:
[t, y] = ode45(odefun, tspan, y0, options)
Parametry:

- odefun: Funkcja definiująca system równań różniczkowych. Musi przyjmować dwa argumenty: t (czas) i y (wartości zmiennych zależnych).
- tspan: Wektor z dwoma elementami [tstart, tend] określający zakres czasowy.
- y0: Wektor początkowy zawierający wartości początkowe zmiennych zależnych.
- options (opcjonalne): Struktura zawierająca ustawienia algorytmu.

Wyjścia:

- t: Wektor czasów, w których obliczenia zostały wykonane.
- y: Macierz, w której kolumny zawierają wartości zmiennych zależnych w odpowiednich czasach.

### 1.2.4 Jawna metoda Eulera

Metoda Eulera jest prostą metodą numeryczną do rozwiązywania równań różniczkowych, oparta na przybliżeniu pochodnej funkcji za pomocą ilorazu różnicowego. Zakłada ona, że wartość pochodnej w danym punkcie jest podobna do średniego tempa zmiany tej funkcji w danym przedziale czasowym.

Wzór jawnej metody Eulera:

$$y_{i+1} = y_i + f(t_i, y_i)\Delta t \tag{3}$$

Gdzie:
$\Delta t = (t_{i+1} - t_i)$
$y_i$ to wartości funkcji w kroku czasowym $t_i$
$f(t_i, y_i)$ to wartość pochodnej funkcji w punkcie $(t_i, y_i)$

Kroki algorytmu:
1. Definiujemy równanie różniczkowe: $\frac{dy}{dt} = f(t, y)$
2. Określamy warunki początkowe: $y(t_0) = y_0$
3. Iteracyjnie obliczamy kolejne wartości: $y_{i+1} = y_i + f(t_i, y_i)\Delta t$,

### 1.2.5 Niejawna metoda Eulera

Niejawna metoda Eulera to inna technika numerycznego rozwiązywania równań różniczkowych, która różni się od jawnej metody Eulera tym, że wykorzystuje wartości funkcji w punktach czasowych przyszłych, zamiast obecnych. Czyli, używa wartości, które będą obliczone w przyszłości, co sprawia, że jest metodą niejawną.
Wzór niejawnej metody Eulera ma postać:

$$y_{i+1} = y_i + f(t_{i+1}, y_{i+1})\Delta t \tag{4}$$

Gdzie:
$\Delta t = (t_{i+1} - t_i)$
$y_i$ to wartości funkcji w kroku czasowym $t_i$
$f(t_{i+1}, y_{i+1})$ to wartość pochodnej funkcji w punkcie $(t_{i+1}, y_{i+1})$

Aby być w stanie wyznaczyć wartość przybliżenia funkncji w punkcie $y_{i+1}$ musimy znać wzór $f(t, y)$, by rozpisać równanie (4) i wyznaczyć wzór na $y_{i+1}$.

### 1.2.6 Metoda Adamsa-Bashfortha drugiego rzędu

Metoda Adamsa-Bashfortha to jedna z metod numerycznego rozwiązywania równań różniczkowych. Jest to metoda jawna oparta na koncepcji ekstrapolacji, która wykorzystuje wartości funkcji w dwóch poprzednich krokach czasowych do obliczenia wartości w kroku czasowym aktualnym. Zakłada się, że wartość pochodnej w punkcie czasowym aktualnym jest podobna do średniego tempa zmiany tej funkcji w dwóch poprzednich punktach czasowych.
Wzór metody:

$$y_{i+1} = y_i + \frac{\Delta t}{2}(3f(t_i, y_i) - f(t_{i-1}, y_{i-1})) \tag{5}$$

Gdzie:
$\Delta t = (t_{i+1} - t_i)$
$y_i$ to wartości funkcji w kroku czasowym $t_i$ $f(t_{i+1}, y_{i+1})$ to wartość pochodnej funkcji w punkcie $(t_{i+1}, y_{i+1})$

Kroki Algorytmu:
1. Definiujemy równanie różniczkowe: $\frac{dy}{dt} = f(t, y)$
2. Określamy warunki początkowe: $y(t_0) = y_0$
3. Obliczamy początkowe wartości za pomocą innej metody (np. jawnej metody Eulera):
$y_1 = y_0 + f(t_0, y_0)\Delta t$
4. Iteracyjnie obliczamy kolejne wartości: $y_{i+1} = y_i + \frac{\Delta t}{2}(3f(t_i, y_i) - f(t_{i-1}, y_{i-1}))$

# 2 Metodyka i wyniki doświadczeń

W tym rozdziale postaramy się pokazać jak w praktyce dopasować model matematyczny do dostarczonych danych. Przypomnijmy że model wyraża się następującym układem równań różniczkowych:

$$\begin{cases} \frac{dx}{dt} = r_x(t) + r_{xy}x(t)y(t) + r_{xx}x^2(t) \\ \frac{dy}{dt} = r_y y(t) + r_{yx}x(t)y(t) + r_{yy}y^2(t) \end{cases} \tag{6}$$

## 2.1 Zadanie 1

Najpierw postaramy się znaleźć współczynniki w pierwszym równaniu układu (1). Do znalezienia rozwiązania równania różniczkowego użyjemy jawnej metody Eulera.
Jej wzór będzie miał postać:

$$\begin{aligned} &\hat{x}_n = \hat{x}_{n-1} + f_x(\hat{x}_{n-1}, \tilde{y}_{n-1})\Delta t_n, n = 2, ..., N \\ &\Delta t_n = t_n - t_{n-1}, \\ &f_x(x, y) = r_x x + r_{xy}xy + r_{xx}x^2 \end{aligned} \tag{7}$$

Za wartości $y(t_1), ..., y(t_N)$ podstawimy dane $\tilde{y}_1, ..., \tilde{y}_N$, zauważmy że takie założenie spowoduj wyzerowanie wskaźnika $J_y$ we wzorze (2). Oprócz znalezienia parametrów $r_x, r_{xy}, r_{xx}$, będziemy musieli znaleźć również wartość początkową $\hat{x}_1$.

Aby to zrobić, posłużymy się wskazówką z zadania i poszukamy ich na konkretnych przedziałach. Dla każdego z tych przedziałów, wybieramy pewną niewielką liczbę punktów, następnie dla każdej kombinacji wyznaczamy rozwiązanie równania metodą Eulera, sprawdzamy współczynnik dopasowania (2) i wybieramy najlepszą kombinację.

Oczywiście znaleziona kombinacja nie jest jeszcze dla nas wystarczająco satysfakcjonująca, poprawimy to używając funkcji fminsearch z Optimization Toolbox. Jako argumenty przekazujemy jej funkcje dopasowania i znalezione argumenty początkowe.

W drugim podpunkcie tego zadania przeprowadzimy dokładnie te same operacje z tym, że dla drugiego równania w naszym układzie (1). Wówczas będziemy mieć:

$$\begin{aligned} &\hat{y}_n = \hat{y}_{n-1} + f_y(\hat{x}_{n-1}, \tilde{y}_{n-1})\Delta t_n, n = 2, ..., N \\ &\Delta t_n = t_n - t_{n-1}, \\ &f_y(x, y) = r_y y + r_{yx}xy + r_{yy}y^2 \end{aligned} \tag{8}$$

Dalej postępujemy analogicznie.
Główny kod programu, oraz pomocnicze funkcja znajdują się w listingu : (1),(2) oraz (3)

## 2.2 Zadanie 2

W tym zadaniu bedziemy postępowa dokładnie z tą samą metodyką, co w pierwszym zadaniu, użyjemy jednak innych metod numerycznych rozwiązywania równań różniczkowych.

### 2.2.1 Niejawna metoda Eulera

Najpierw skorzystamy z niejawnej metody Eulera:

$$\begin{aligned} &\hat{x}_n = \hat{x}_{n-1} + f_x(\hat{x}_n, \tilde{y}_n)\Delta t_n, n = 2, ..., N \\ &\hat{y}_n = \hat{y}_{n-1} + f_y(\hat{x}_n, \tilde{y}_n)\Delta t_n, n = 2, ..., N \end{aligned} \tag{9}$$

Jest to metoda niejawna, więc aby móc ją zaimplementować musimy przekształcić jej wzór, i tak rozpisując $f_x$ otrzymujemy:

$$\begin{aligned} &\hat{x}_n = \hat{x}_{n-1} + f_x(\hat{x}_n, \tilde{y}_n)\Delta t_n \\ &\hat{x}_n = \hat{x}_{n-1} + (r_x\hat{x}_n + r_{xy}\hat{x}_n\tilde{y}_n + r_{xx}\hat{x}^2)\Delta t_n \\ &\hat{x}_n^2(r_{xx}\Delta t_n) + \hat{x}_n(\Delta t_n(r_x + r_{xy}\tilde{y}_n) - 1) + \hat{x}_{n-1} = 0 \end{aligned} \tag{10}$$

Otrzymujemy równanie kwadratowe, widzimy że odpowiednie współczynniki w równaniu $ax^2 + bx + c$ są równe:

$a = r_{xx}\Delta t_n$

$b = \Delta t_n(r_x + r_{xy}\tilde{y}_n) - 1$

$c = \hat{x}_{n-1}$

Za rozwiązanie równania kwadratowego przyjmiemy $\frac{-b-\sqrt{b^2-4ac}}{2a}$. Próbując znaleźć startowe parametry będziemy pomijać te, dla których rozwiązania równania różniczkowego wychodzą zespolone. Kod programu i niejawnej metody eulera został przedstawiony w listingu: (5) (4)

### 2.2.2 Metoda Adamsa-Bashfortha drugiego rzedu

W tym podpunkcie wykorzystamy metodę Adamsa-Bashfortha drugiego rzędu do rozwiązania równania różniczkowego.
Wzór metody:

$$\begin{aligned}\hat{x}_n &= \hat{x}_{n-1} + (\frac{3}{2}f_x(\hat{x}_{n-1}, \tilde{y}_{n-1}) - \frac{1}{2}f_x(\hat{x}_{n-2}, \tilde{y}_{n-2}))\Delta t_n, n = 3, ..., N \\ \hat{y}_n &= \hat{y}_{n-1} + (\frac{3}{2}f_y(\hat{x}_{n-1}, \tilde{y}_{n-1}) - \frac{1}{2}f_y(\hat{x}_{n-2}, \tilde{y}_{n-2}))\Delta t_n, n = 3, ..., N\end{aligned} \tag{11}$$

Jest to metoda dwókrokowa, zatem musimy wyznaczyć jeszcze $\hat{x}_2$ i $\hat{y}_2$, wykożystamy do tego jawną metodę eulera:

$$\begin{aligned}\hat{x}_2 &= \hat{x}_1 + f_x(\hat{x}_1, \tilde{y}_1)\Delta t_2 \\ \hat{y}_2 &= \hat{y}_1 + f_y(\hat{y}_1, \tilde{y}_1)\Delta t_2\end{aligned} \tag{12}$$

Implementację metody Adama-Bashfortha przedstawia listing: (6)

## 2.3 Zadanie 3

W tym zadaniu zamiast, tak jak w zadaniach 1 i 2 najpierw szukać współczynników i rozwiązywać jedno, potem drugie równanie, postaramy się rozwiązać cały układ (1), najpierw za pomocą funkcji ode45, później jawnej metody Eulera. Przy szukaniu współczynników będziemy starali się zminimalizować wskaźnik dopasowania modelu: (2).
Przy numerycznym rozwiązywaniu układu równań różniczkowych musimy dobrać dostatecznie mały krok całkowania, a następnie w celu obliczenia wartości $J$ - interpolować uzyskane rozwiązanie w punktach $t1, ..., t_N$ za pomocą funkcji interp1.
Dla metody ode45 jako punkty startowy wybieżemy, te które znaleźliśmy w zadaniu 1, natomiast dla jawnej metody eulera, uzyskane metodą Adamsa-Bashfortha. Kod programu realizujacego zadanie 3 przedstawiony jest w listingu: (7) i (8).

## 2.4 Zadanie 4

W tym zadaniu postaramy się odpowiedzieć na dwa pytania:

### 2.4.1 Dla jakiej pary wartości $x, y > 0$ układ osiąg stan równowagi (tzn. brak zmiany $x(t)$ i $y(t)$ w czasie)?

Jeśli liczba przedstawicieli gatunku ofiar i drapieżników się nie zmienia to:

$$\begin{aligned}\frac{dx}{dt} &= 0 \\ \frac{dy}{dt} &= 0\end{aligned} \tag{13}$$

Po podstawieniu do układu (1) otrzymujemy:

$$\begin{cases}0 = r_x x + r_{xy}xy + r_{xx}x^2 \\ 0 = r_y y + r_{yx}xy + r_{yy}y^2\end{cases} \tag{14}$$

Za $r_x, r_{xy}, r_{xx}, r_y, r_{yx}, r_{yy}$ w układzie równań przyjmiemy współczynniki wyznaczone w zadaniu 3 funkcją ode45. Aby rozwiązać ten układ równań w matlabie, tworzymy najpierw równania symboliczne i stosujemy do nich funkcję solve z Symbloic Math Toolbox. Wybieramy te rozwiązanie którego $x$ i $y$ są dodatnie.
Jak można się spodziewać rozwiązaniem takiego układu nie są liczby całkowite, co nie ma sensu w kontekście naszego zadania w którym pracujemy na całkowitej liczbie przedstawicieli gatunku. Musimy więc zaokrąglić rozwiązanie do liczb całkowitych. Kod znajduje się w listingu (9).

### 2.4.2 Jak można interpretować wartości parametrów $r_x, r_{xy}, r_{xx}, r_y, r_{yx}, r_{yy}$?

Poszczególne parametry w układzie: (1) można interpretować następująco:

- $r_x$ - Jest to tempo wzrostu populacji ofiar, częstość narodzin ofiar,
- $r_{xy}$ - współczynnik wpływu populacji drapieżników na wzrost populacji ofiar, określa częstość umierania ofiar na skutek drapieżnictwa
- $r_{xx}$ - Wartość ta określa w jaki sposób populacja x reguluje swój własny wzrost (konkurencja wewnątrzgatunkowa i umieralność z powodu przepełnienia obszaru)
- $r_y$ - Tempo wzrostu populacji drapieżników, częstość narodzin drapieżników
- $r_{yx}$ - współczynnik wpływu populacji ofiar na populację drapieżników
- $r_{yy}$ - współczynnik samoregulacji populacji drapieżników

## 2.5 Zadanie 5

W tym zadaniu będziemy postępować analogicznie jak w poprzednich. Spróbujemy znaleźć współczynniki $r_x, r_{xy}, r_{xx}, r_y, r_{yx}, r_{yy}$ w układzie równań (1) oraz poszukać wartości początkowych $\hat{x}_1$ i $\hat{y}_1$.

Postaramy się dopasować model używając jawnej metody eulera, niejawnej metody eulera i metody Adama-Bashfortha do znalezienia przybliżeń początkowych, a następnie użyjemy funkcji ode45 do poprawienia parametrów modelu. Jako funkcję dopasowania, tak jak poprzednio użyjemy: (**??**eq:eq2).

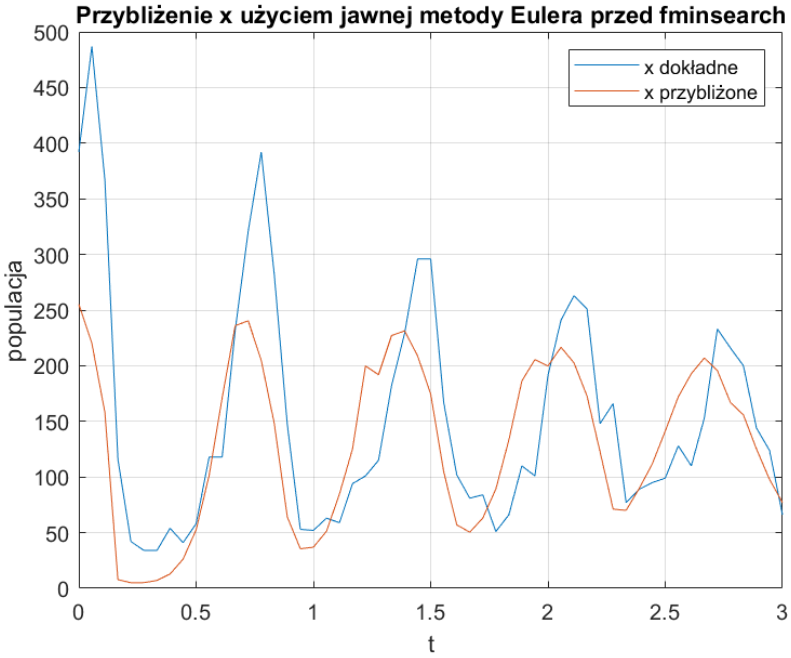
Tym razem nie będziemy, jak w poprzednich zadanich pracować na danych syntetycznych, tylko na rzeczywistych danych, a dokładnie będziemy działać ze zbiorem danych, w którym znajdują się pomiary liczebności populacji zajęcy amerykańskich i rysi kanadyjskich w pewnym obszarze Kanady.

Dane były zbierane w latach 1845 - 1935. Daty kolejnych pomiarów przeskalujemy tak, aby zakres był taki sam jak w poprzednich zadaniach. Dalej postępujemy analogicznie.

# 3 Dyskusja wyników eksperymentów numerycznych

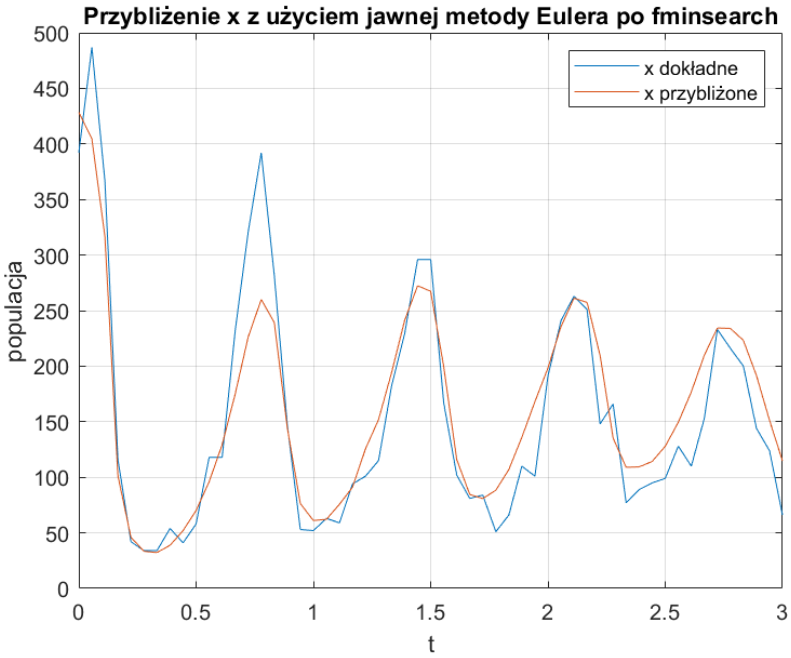
## 3.1 Zadanie 1

Tak prezentują się wyniki próby znalezienia przybliżenia $x(t)$:




Rysunek 1: Przybliżenie metodą jawną Eulera dla najlepszych parametrów startowych

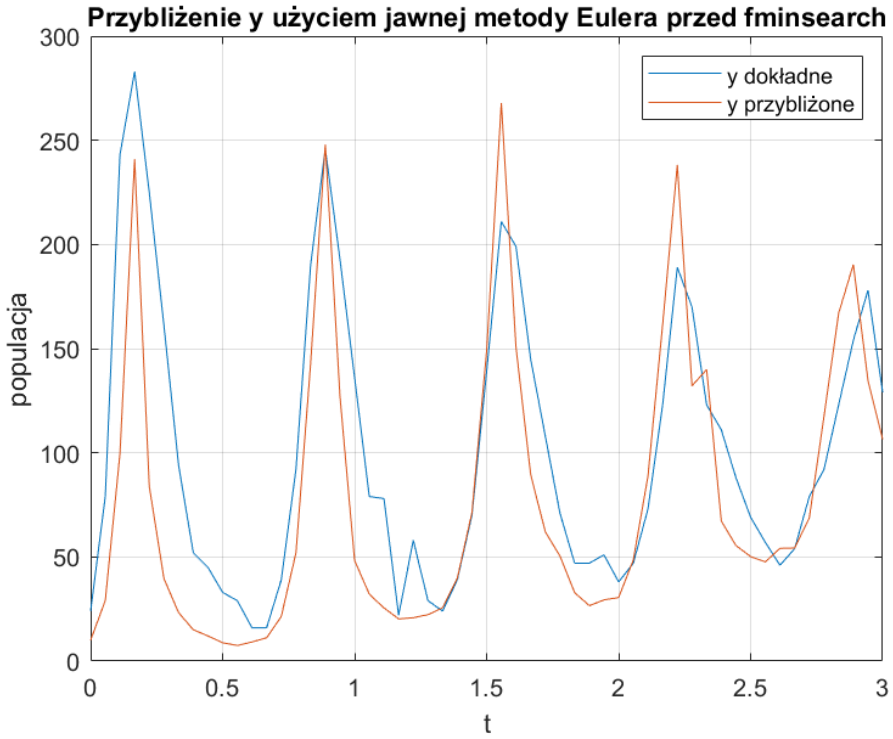
Jak widać początkowe przybliżenie nie jest zbyt dokładne, ale widać pewną tendencję. Użyjemy funkcji fminsearch aby dopasowanie było jeszcze lepsze.



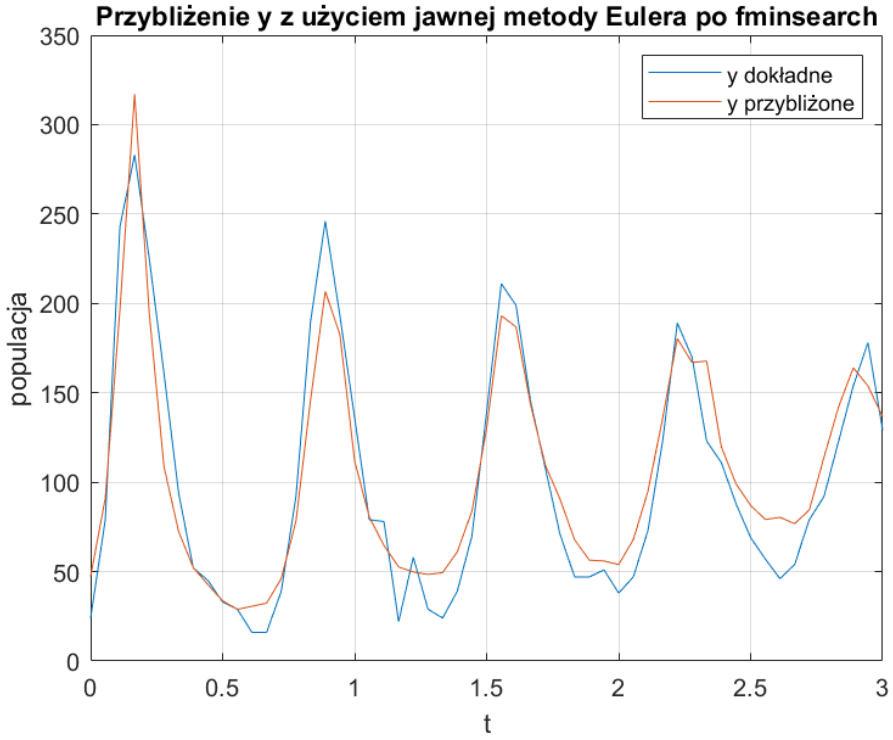

Rysunek 2: Przybliżenie metodą jawną Eulera po użyciu fminsearch

Jak widać użycie funkcji fminsearch znacząco poprawiło dokładność, choć przybliżenie nie jest dokładnie takie samo jak orginalne dane, to jednak w większości wykresy pokrywają się i funkcje zachowują w podobny sposób.

Tak prezentują się wyniki próby znalezienia przybżenia $y(t)$:




Rysunek 3: Przybliżenie metodą jawną Eulera przed użyciem fminsearch



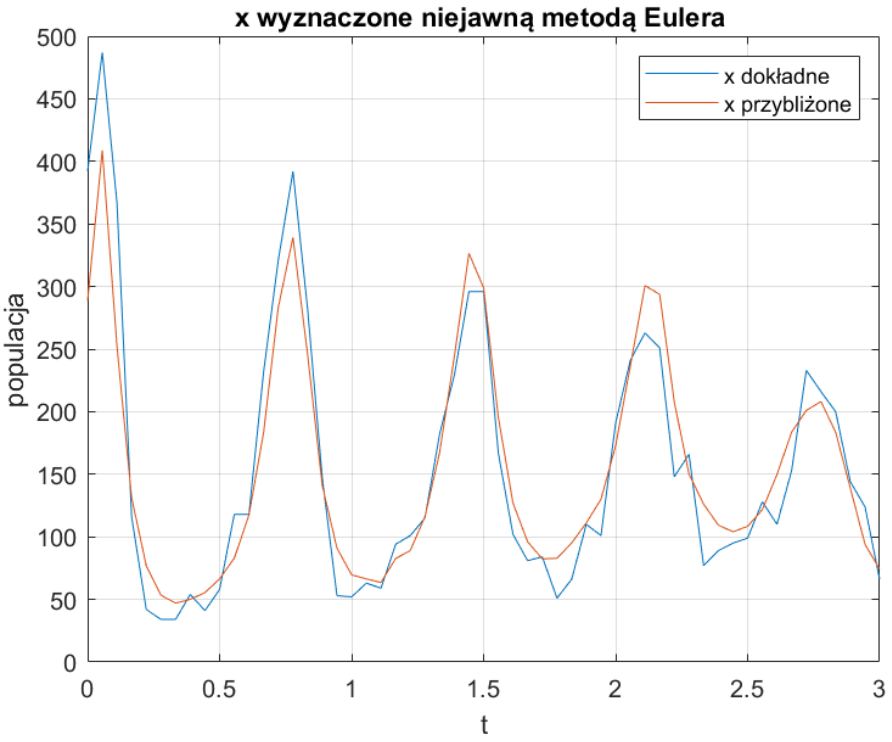

Rysunek 4: Przybliżenie metodą jawną Eulera po użyciu fminsearch

Tutaj ponownie obserwujemy znaczną poprawę dopasowania modelu po urzyciu funkcji fminsearch.

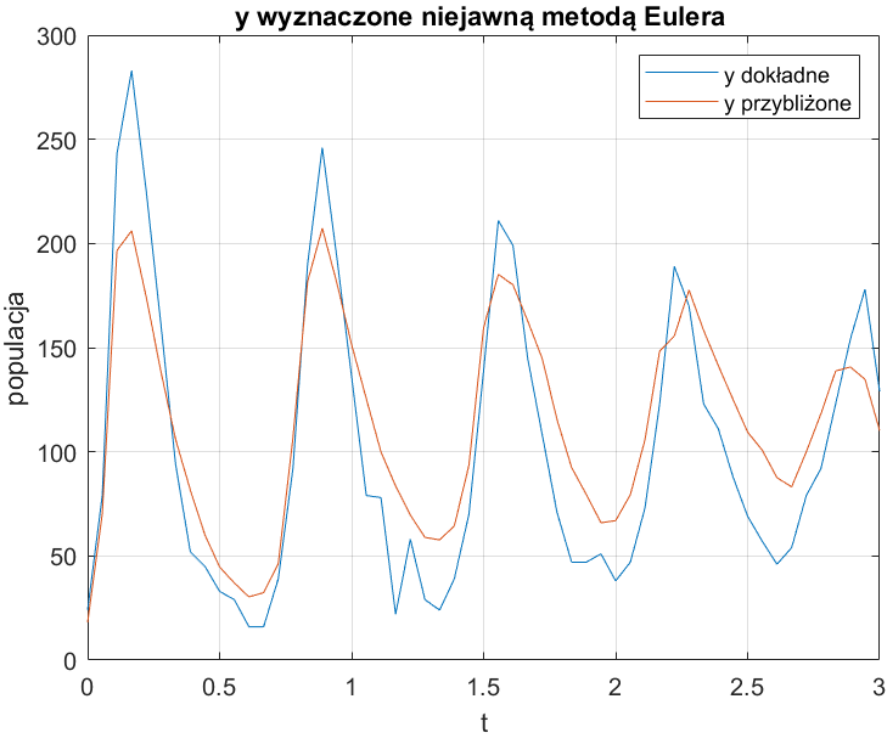
## 3.2 Zadanie 2

### 3.2.1 Niejawna metoda Eulera

Tak prezentuje się wykres przybliżonej funkcji, uzyskany niejawną metodą eulera dla wartości parametrów poprawionych funkcją fminsearch:



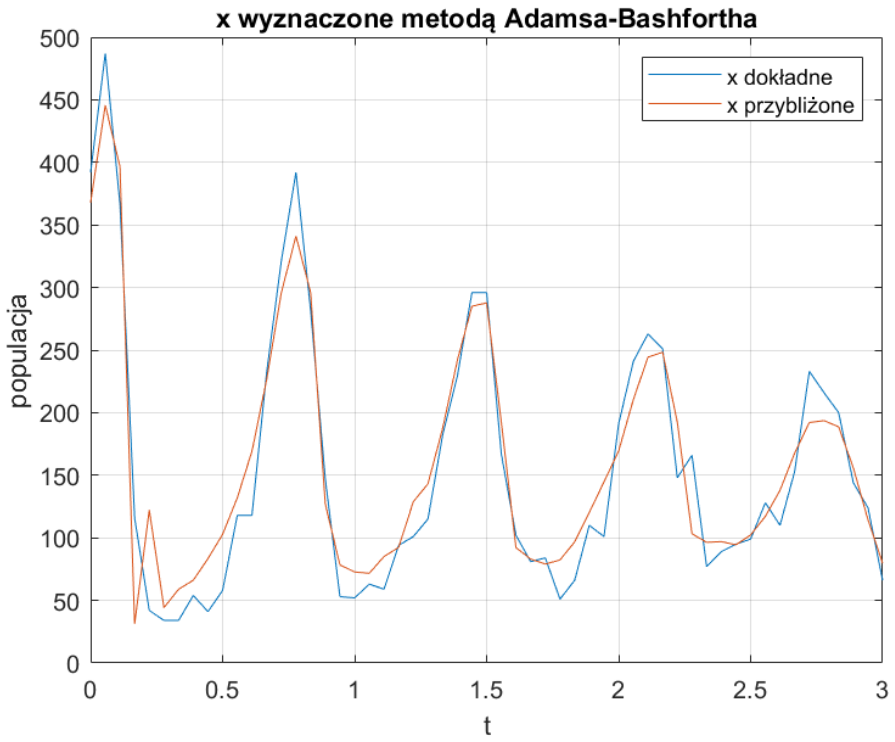

Rysunek 5: Przybliżenie x niejawną metodą Eulera



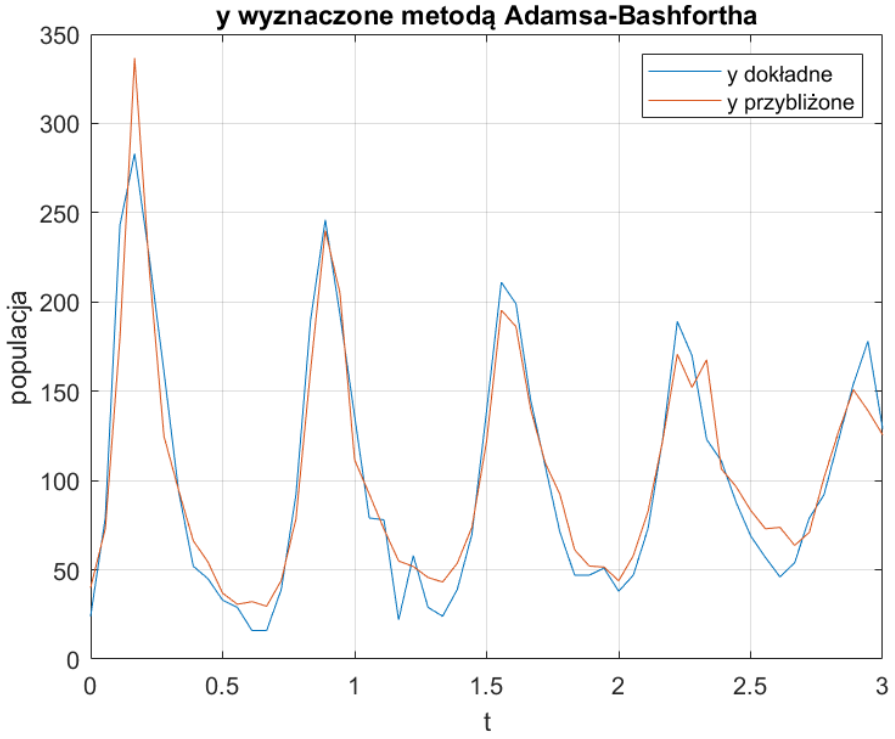

Rysunek 6: Przybliżenie y niejawną metodą Eulera

Jak widzimy dopasowanie $y(t)$ do modelu jest nieco gorsze niż $x(t)$, nie mniej jednak wydaje się być całkiem satysfakcjonujące.

### 3.2.2 Metoda Adamsa-Bashfortha



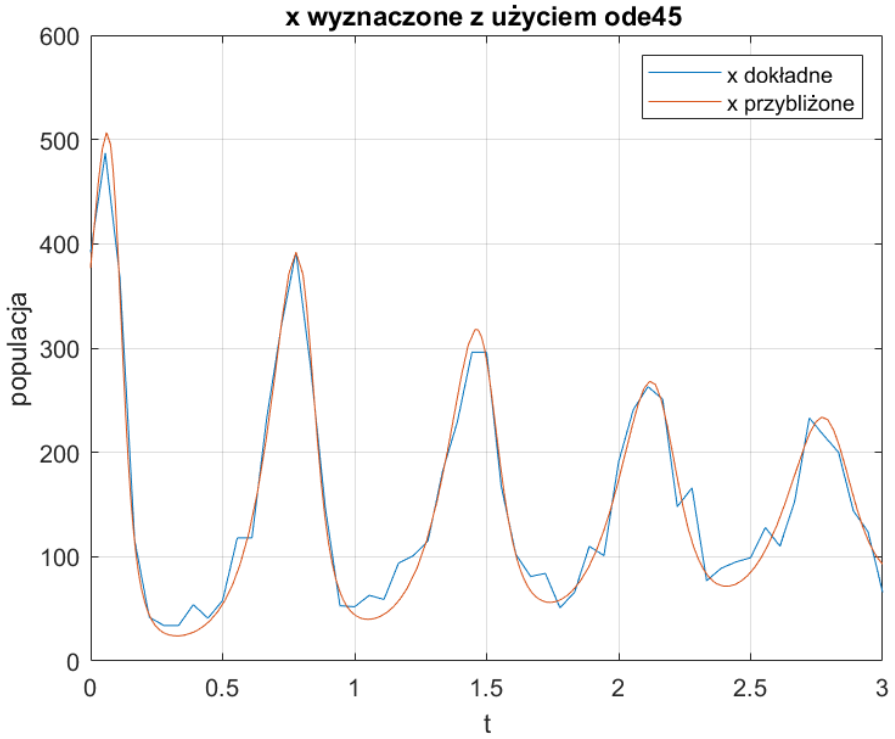

Rysunek 7: Przybliżenie x metodą Adama-Bashfortha



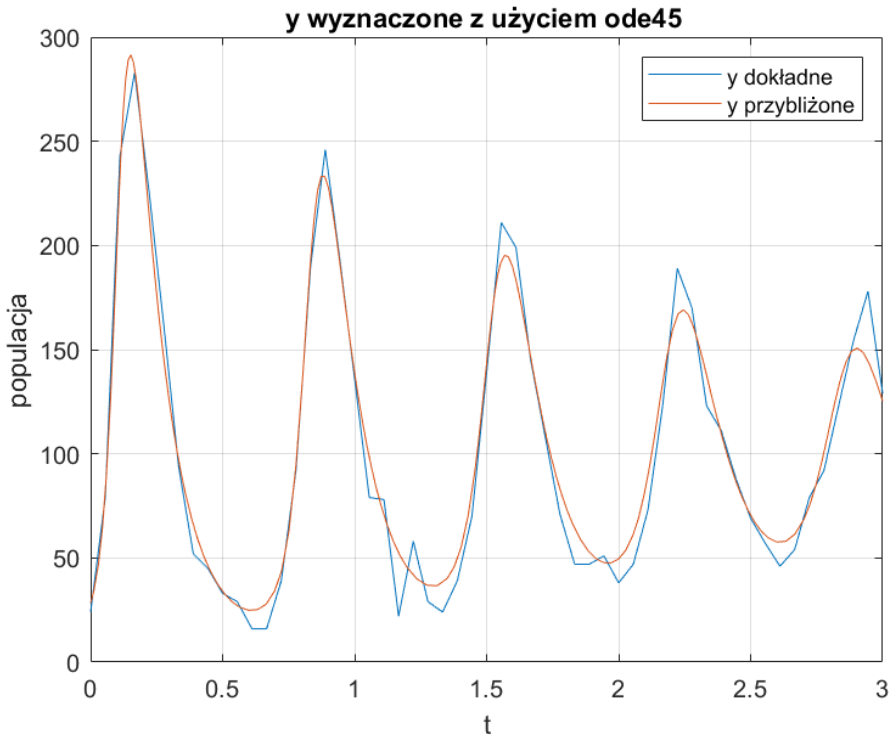

Rysunek 8: Przybliżenie y metodą Adamsa-Bashfortha

Jak widać metoda Adamsa-Bashfortha daje bardzo dobre dopasowanie.

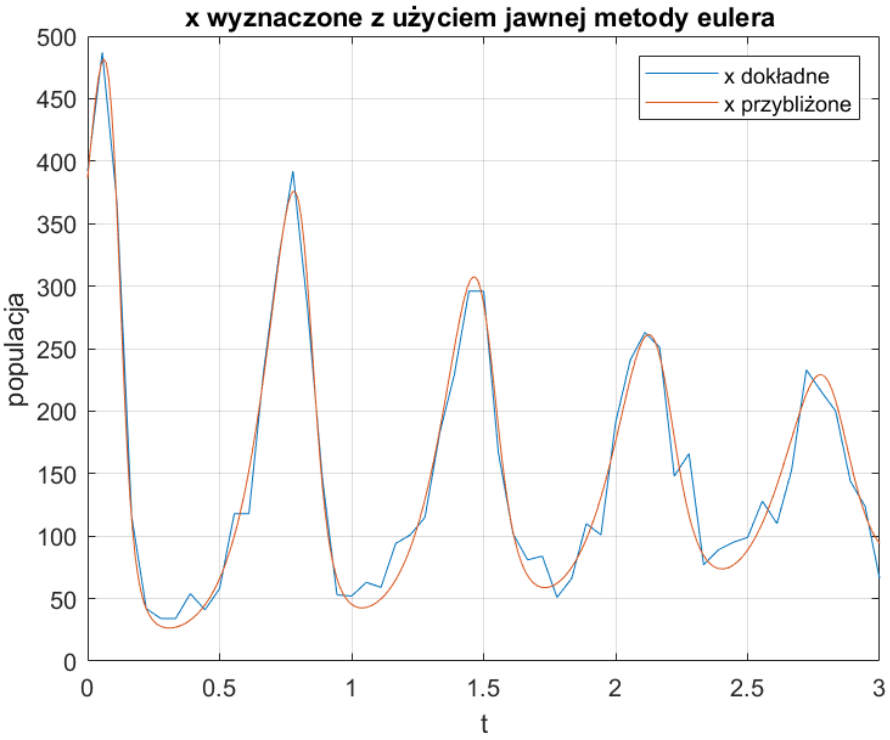
### 3.3 Zadanie 3



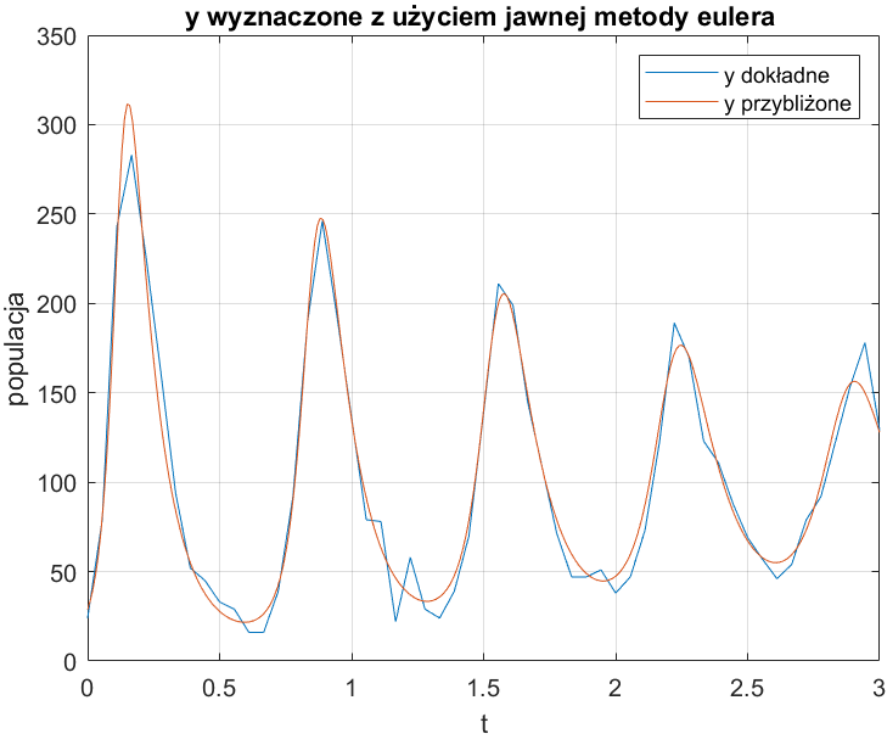

Rysunek 9: Przybliżenie x funkcja ode45




Rysunek 10: Przybliżenie y funkcja ode45



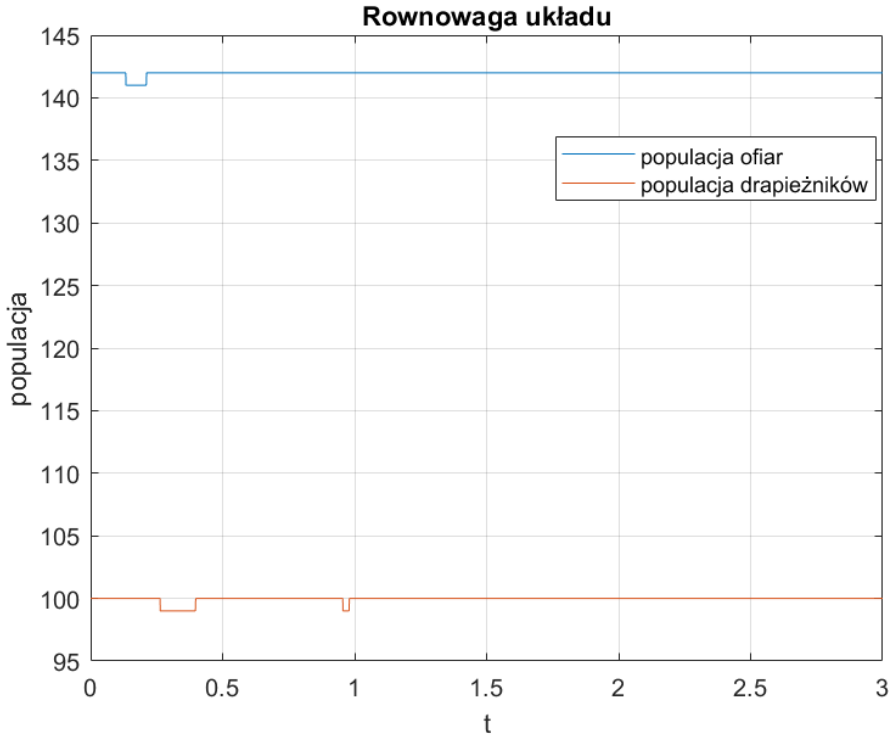

Rysunek 11: Przybliżenie x jawna metoda eulera




Rysunek 12: Przybliżenie y jawna metoda eulera

Po użyciu jawnej metody eulera lub funkcji ode45 do rozwiązywania układu równań, zastosowaniu parametru dopasowania modelu 2 i użyciu fminsearch, jesteśmy w stanie znaleźć bardzo dokładne współczynniki w równaniu 1 i wartości początkowe modelu.

## 3.4 Zadanie 4

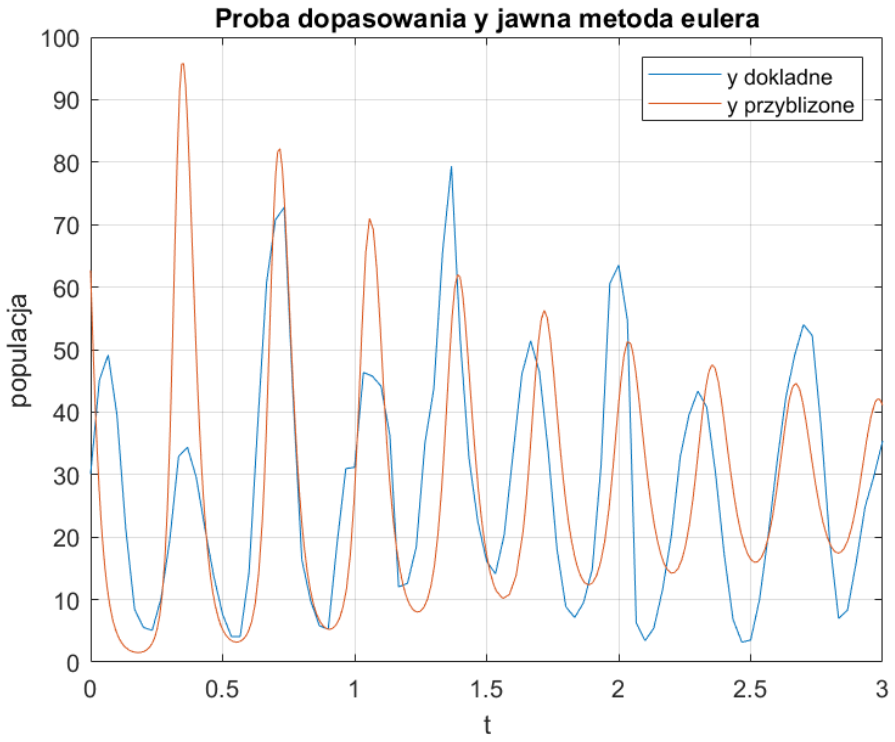



Rysunek 13: Stan równowagi układu

Jak widać dla obliczonych wartości początkowych $x_1$ i $y_1$ układ nie jest całkowicie stabilny, występują różnice wielkości jednego przedstawiciela gatunku. Różnice te Wynikają z konieczności zaokrąglenia rozwiązania układu (14) do liczby całkowitej.

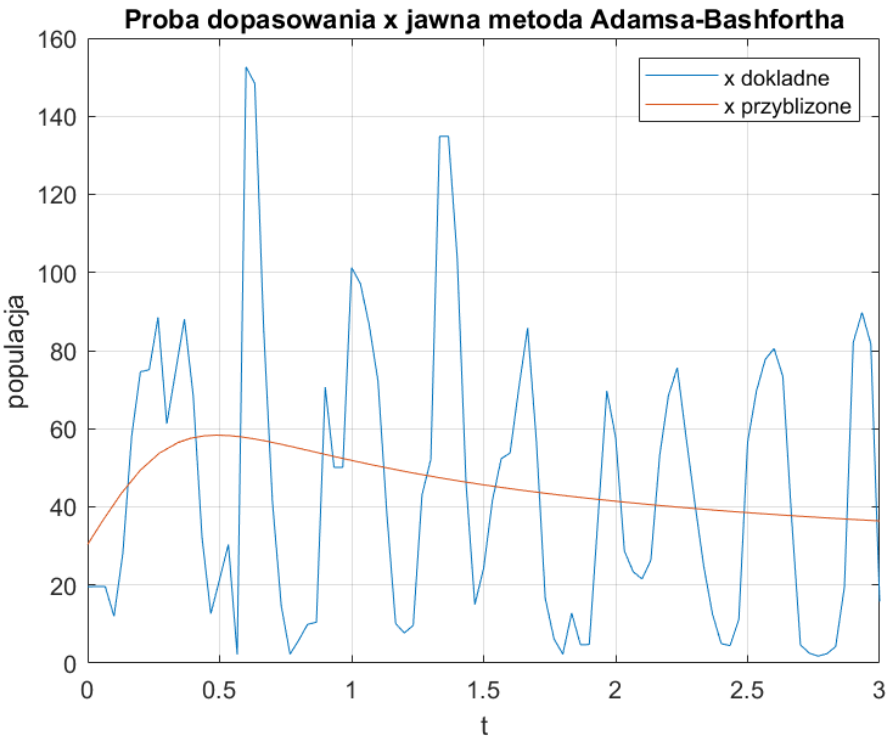
## 3.5 Zadanie 5


Rysunek 14: Próba dopasowania modelu jawna metoda eulera

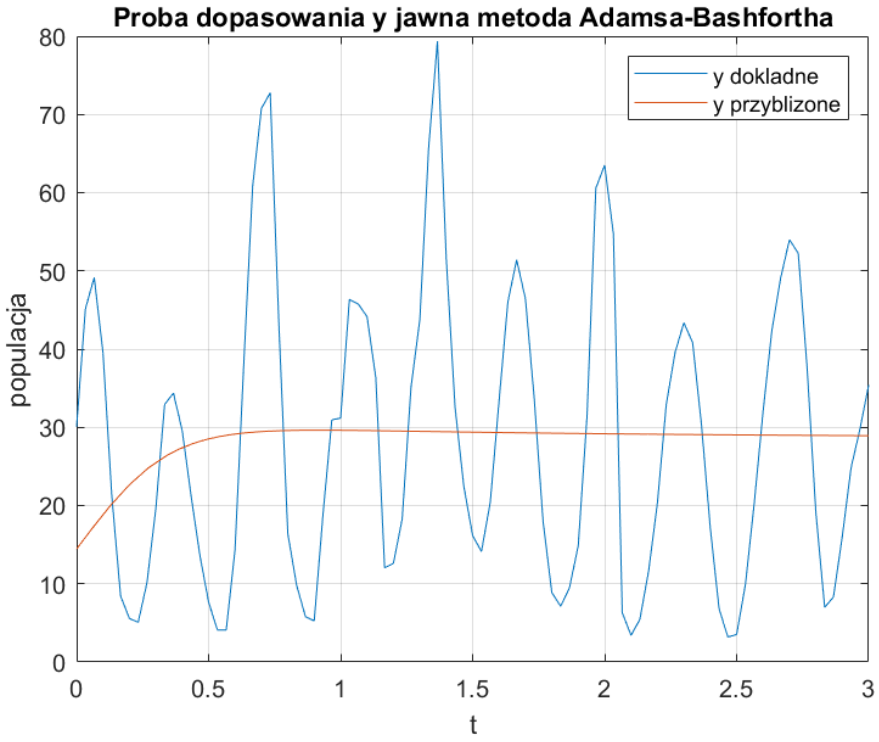



Rysunek 15: Próba dopasowania modeul jawna metoda eulera

Jak widać próba dopasowania modelu, jawną metodą eulera nie powiodłą się, wykresy są bardzo rozbieżne.

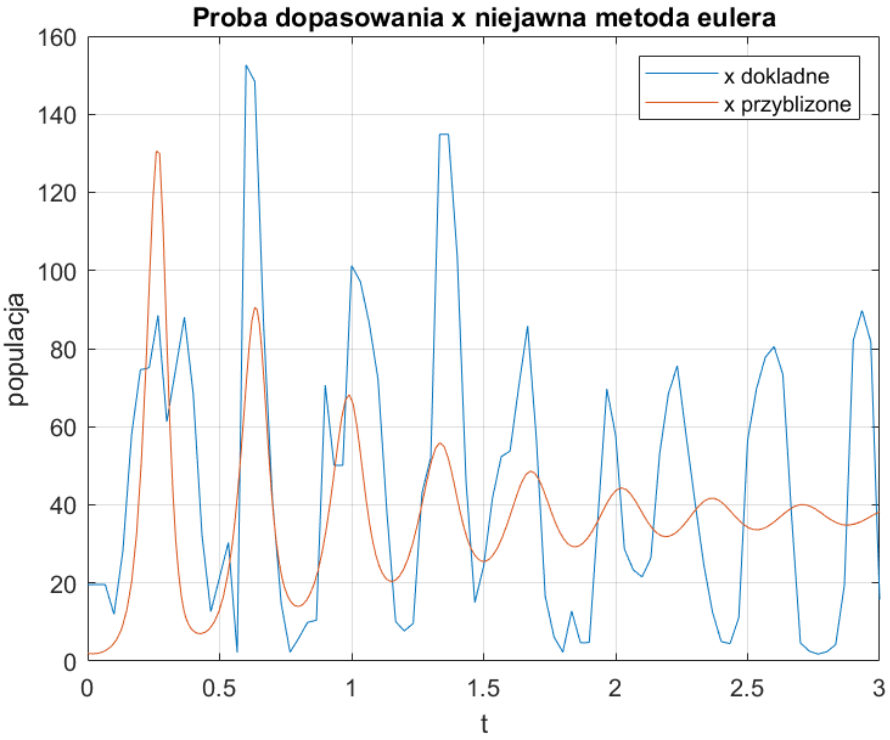



Rysunek 16: Próba dopasowania modelu metodą Adamsa-Bashfortha




Rysunek 17: Próba dopasowania modeul metodą Adamsa-Bashfortha

Metoda Adamsa-Bashfortha nie poradziła sobie z dopasowaniem modelu do danych rzeczywistych




Rysunek 18: Próba dopasowania modelu niejawna metoda eulera


Rysunek 19: Próba dopasowania modeul niejawna metoda eulera

Niejawna metoda Eulera również nie dopasowało modelu do danych.
Powyższe trzy metody nie dały rady znaleźć dobrego dopasowania. Ze wszystkich trzech, co zaskakujące najbliżej dokładnych danych jest wykres uzyskany dzięki jawnej metodzie eulera, ale i tak bardzo daleki od ideału.
Wszytkie otrzymane wyniki nie nadają się do zastosowaniach praktycznych, są zbyt dalekie od faktycznego stanu populacji gatunków.

Dlaczego więc, tak się dzieje? Jest kilka przyczyn trudność w dopasowaniu modelu do danych:

1. Brak pewności co do zebranych danych, W niektórych przypadkach może być brak dokładnych danych dotyczących populacji i czynników wpływających na nią. Precyzyjne dane są kluczowe do efektywnego dopasowania modelu.

2. Niewystarczająca liczba uwzględnionych czynników w modelu. Model Lotki-Volterry jest stosunkowo prostym modelem i zakłada kilka uproszczeń, które mogą nie odzwierciedlać wszystkich skomplikowanych interakcji i czynników wpływających na populacje. Rzeczywiste ekosystemy są bardziej złożone, a wpływ czynników takich jak dostępność pożywienia, choroby, zmiany klimatyczne czy ludzka działalność nie jest uwzględniony w ramach tego modelu.

3. Zmienność parametrów w modelu. Rzeczywiste środowisko jest dynamiczne i zmienne w czasie. Model matematyczny Lotki-Volterry, zakłada stałość parametrów, co może być niewłaściwe w przypadku zmieniających się warunków środowiskowych.

4. Długoterminowe Zmiany. Zmiany w ekosystemie, takie jak zmiany w użytkowaniu gruntów, urbanizacja czy zmiany klimatyczne, mogą wpływać na populacje w sposób, który nie jest uwzględniony w tym modelu.

5. Brak uwzględnienia innych gatunków w modelu. Ten model Lotki-Volterry skupia się na relacji drapieżnik - ofiara tylko między dwoma gatunkami, musimy pamiętać, że w rzeczywistości są jeszcze inne gatunki zamieszkujące dany ekosystem, które wpływają na siebie.

# 4 Wnioski

Eksperymenty wskazują na to, że model Lotki-Volterry jest bardzo wrażliwy na zmiany wartości poszczególnych parametrów, takich jak tempo wzrostu, wpływ drapieżników, czy interakcje wewnątrzgatunkowe.Jak przekonaliśmy się w zadaniu 3, dobranie właściwych parametrów w równaniu(1) wymaga bardzo wielu iteracji funkcji fminsearch, aby model dawał zadowalające rezultaty i by można było na nim oprzeć prognozy zmian populacji gatunków.

Jak przekonaliśmy się w zadaniu 5, model nie dopasowuje się dobrze do rzeczywistych danych dotyczących populacji zajęcy amerykańskich i rysi kanadyjskich, może to sugerować, że model Lotki-Volterry nie uwzględnia wszystkich istotnych czynników wpływających na dynamikę tych populacji. W przypadku kiedy chcemy pracować na rzeczywistych danych, konieczne jest rozważenie bardziej złożonych modeli z większą ilością parametrów wziętych pod uwagę.

W naszym projekcie kożystaliśmy z funkcji fminsearch i parametru dopasowania modelu do danych: 2, jednak nasz model nie dopasowywał się dobrze do danych rzeczywistych. Być może gdyby użyć innych technik optymalizacji i parametrów dopasowania modelu, udałoby się nasz model, mimo tego że uwzględnia niewilką ilość parametrów, dopasować lepiej.

# 5 Listing

Listing 1: funkcja opisujaca pojedyńcze równanie w układzie

```
function [out] = fun(y,A)
% funkjca reprezentujaca pojedyncze rownanie w ukladzie rownan
% rozniczkowych
% WEJSCIE:
% y - dwuelementowy wektor zawierajacy wartosci pierwszej i drugiej
%     funkcji w rownaniu
% A - Wektor wspolczynnikow w rownaniu
% WYJSCIE:
% out - obliczona wartosc pochodnej
out = A*[y(1);y(1)*y(2);y(1)*y(1)];
end % function
```

Listing 2: Metoda jawna Eulera

```
function [x] = Euler(A,t,y,x_init,fun)
% Funkcja realizujaca jawna metode Eulera
% WEJSCIE:
% A      - wspolczynniki w rownaniu
% t      - wektor czasu
% x_init - wartosc poczatkowa
```

```
% fun     - uchwyt do funkcji
% WYJSCIE:
% x       - wektor obliczonych wartosci
x = zeros(length(t),1);
x(1) = x_init;
for i=2:length(t)
    delta_t = t(i)-t(i-1);
    x(i) = x(i-1) + fun([x(i-1),y(i-1)],A)*delta_t;
end
end % function
```

Listing 3: Zadanie 1

```
    % Czytanie z pliku
dane = readmatrix("dane22.csv");
t = dane(:,1);
x_dok = dane(:,2);
y_dok = dane(:,3);

% Przedzialy
r_xSpan=[0,40];
r_xySpan = [-1,0];
r_xxSpan = [-0.1,1];
x_Span = [100,1000];

% Ilosc punktow testowych
n_r_x = 15;
n_r_xy = 10;
n_r_xx = 10;
n_x = 30;

Ax_best=zeros(1,3); % Najlepsze wspolczynniki w pierwszym rownaniu
x_init_best = 0; % Najlepsza wartosc poczatkowa x
Jx_min = Inf;

for r_x = linspace(r_xSpan(1),r_xSpan(end),n_r_x)
for r_xy = linspace(r_xySpan(1),r_xySpan(end),n_r_xy)
for r_xx = linspace(r_xxSpan(1),r_xxSpan(end),n_r_xx)
for x = linspace(x_Span(1),x_Span(end),n_x)

    A = [r_x,r_xy,r_xx]; % wspolczynniki w rownaniu

    x_przyb = Euler(A,t,y_dok,x,@fun);
    Jx = sum((x_dok - x_przyb).^2); % Blad

    if (Jx<Jx_min)
        Ax_best = A;
        x_init_best = x;
        Jx_min = Jx;
    end

end
end
end
end
figure;
plot(t,x_dok)
hold on;
plot(t,Euler(Ax_best,t,y_dok,x_init_best,@fun));
hold off;
title("Przyblizenie x uzyciem jawnej metody Eulera przed fminsearch
   ");
```

```
legend("x dokladne", "x przyblizone");
xlabel("t");
ylabel("populacja");
grid("on");

figure;
plot(t,x_dok)
hold on;

% Funkcja bledu
Fun_Jx = @(args)...
    sum((x_dok - Euler(args(1:3),t,y_dok,args(end),@fun)).^2);

% Optymalne argumenty
args_opt = fminsearch(Fun_Jx,[Ax_best,x_init_best]);

Ax_best = args_opt(1:3);
x_init_best = args_opt(end);

plot(t,Euler(Ax_best,t,y_dok,x_init_best,@fun))
hold off;
grid("on")
title("Przyblizenie x z uzyciem jawnej metody Eulera po fminsearch
   ")
legend("x dokladne", "x przyblizone");
xlabel("t");
ylabel("populacja");

% b) ----------------------------
% Przedzialy
r_ySpan=[-40,0];
r_yxSpan = [0,1];
r_yySpan = [-0.1,0];
y_Span = [10,200];

% Ilosc punktow testowych
n_r_y = 15;
n_r_yx = 10;
n_r_yy = 10;
n_y = 30;

Ay_best=zeros(1,3);
y_init_best = 0;
Jy_min = Inf;

for r_y = linspace(r_ySpan(1),r_ySpan(end),n_r_y)
for r_yx = linspace(r_yxSpan(1),r_yxSpan(end),n_r_yx)
for r_yy = linspace(r_yySpan(1),r_yySpan(end),n_r_yy)
for y = linspace(y_Span(1),y_Span(end),n_y)

    A = [r_y,r_yx,r_yy]; % Wspolczynniki w rownaniu

    y_przyb = Euler(A,t,x_dok,y,@fun);
    Jy = sum((y_dok - y_przyb).^2); % Blad

    if (Jy<Jy_min)
        Ay_best = A;
        y_init_best = y;
        Jy_min = Jy;
    end
end
```

```
end
end
end
figure
plot(t,y_dok)
hold on
plot(t,Euler(Ay_best,t,x_dok,y_init_best,@fun))
hold off
title("Przyblizenie y uzyciem jawnej metody Eulera przed fminsearch
    ");
legend("y dokladne", "y przyblizone");
xlabel("t");
ylabel("populacja");
grid("on");

figure
plot(t,y_dok)
hold on;
% Funkcja dokladnosci dopasowania
Fun_Jy = @(args)...
    sum((y_dok - Euler(args(1:3),t,x_dok,args(end),@fun)).^2);

args_opt = fminsearch(Fun_Jy,[Ay_best,y_init_best]);

Ay_best = args_opt(1:3);
y_init_best = args_opt(end);

plot(t,Euler(Ay_best,t,x_dok,y_init_best,@fun))
hold off;
grid("on")
title("Przyblizenie y z uzyciem jawnej metody Eulera po fminsearch
    ")
legend("y dokladne", "y przyblizone");
xlabel("t");
ylabel("populacja");
```

Listing 4: Euler2

```
function [x] = Euler2(A,t,y,x_init)
% funkcja realizujaca niejawna metode eulera
% WEJSCIE:
% WEJSCIE:
% A      - wspolczynniki w rownaniu
% t      - wektor czasu
% x_init - wartosc poczatkowa
% WYJSCIE:
% x      - wektor obliczonych wartosci
x = zeros(length(t),1);
x(1) = x_init;
for i = 2:length(t)
    del_t = t(i) - t(i-1);
    a = del_t*A(3);
    b = del_t*(A(1)+A(2)*y(i))-1;
    c = x(i-1);
    x(i) = (-b - sqrt(b*b - 4*a*c))/(2*a);
end
end % function
```

Listing 5: Zadanie 2a

```
% Czytanie z pliku
dane = readmatrix("dane22.csv");
```

```
t = dane(:,1);
x_dok = dane(:,2);
y_dok = dane(:,3);

% Przedzialy
r_xSpan=[0,40];
r_xySpan = [-1,0];
r_xxSpan = [-0.1,1];
x_Span = [100,1000];

% Ilosc podprzedzialow
n_r_x = 15;
n_r_xy = 10;
n_r_xx = 10;
n_x = 30;

fms_options = optimset("MaxFunEvals",4000,"MaxIter",4000);

Ax_best=zeros(1,3); % Najlepsze wspolczynniki w pierwszym rownaniu
x_init_best = 0; % Najlepsza wartosc poczatkowa x
Jx_min = Inf;

for r_x = linspace(r_xSpan(1),r_xSpan(end),n_r_x)
for r_xy = linspace(r_xySpan(1),r_xySpan(end),n_r_xy)
for r_xx = linspace(r_xxSpan(1),r_xxSpan(end),n_r_xx)
for x = linspace(x_Span(1),x_Span(end),n_x)

    A = [r_x,r_xy,r_xx]; % wspolczynniki w rownaniu

    x_przyb = Euler2(A,t,y_dok,x);
    Jx = sum((x_dok - x_przyb).^2); % Blad

    if (Jx<Jx_min) && (isreal(x_przyb))
        Ax_best = A;
        x_init_best = x;
        Jx_min = Jx;
    end

end
end
end
end

figure;
plot(t,x_dok)
hold on;

% Funkcja bledu
Fun_Jx = @(args)...
    sum((x_dok - Euler2(args(1:3),t,y_dok,args(end))).^2);

% Optymalne argumenty
 args_opt = fminsearch(Fun_Jx,[Ax_best,x_init_best],...
     fms_options);

 Ax_best = args_opt(1:3);
 x_init_best = args_opt(end);

plot(t,Euler2(Ax_best,t,y_dok,x_init_best));
hold off;
title("x wyznaczone niejawna metoda Eulera ")
```

```
xlabel("t")
ylabel("populacja")
legend("x dokladne", "x przyblizone")
grid("on")

% 2) ----------------------------------------

r_ySpan=[-40,0];
r_yxSpan = [0,1];
r_yySpan = [-0.1,0];
y_Span = [10,200];

n_r_y = 30;
n_r_yx = 10;
n_r_yy = 10;
n_y = 40;

Ay_best=zeros(1,3);
y_init_best = 0;
Jy_min = Inf;

for r_y = linspace(r_ySpan(1),r_ySpan(end),n_r_y)
for r_yx = linspace(r_yxSpan(1),r_yxSpan(end),n_r_yx)
for r_yy = linspace(r_yySpan(1),r_yySpan(end),n_r_yy)
for y = linspace(y_Span(1),y_Span(end),n_y)

    A = [r_y,r_yx,r_yy]; % Wspolczynniki w rownaniu

    y_przyb = Euler2(A,t,x_dok,y);
    Jy = sum((y_dok - y_przyb).^2); % Blad

    if (Jy<Jy_min)
        Ay_best = A;
        y_init_best = y;
        Jy_min = Jy;
    end
end
end
end
end

figure
plot(t,y_dok)
hold on;
Fun_Jy = @(args)...
    sum((y_dok - Euler2(args(1:3),t,x_dok,args(end))).^2);

args_opt = fminsearch(Fun_Jy,[Ay_best,y_init_best], ...
    fms_options);

Ay_best = args_opt(1:3);
y_init_best = args_opt(end);

plot(t,Euler2(Ay_best,t,x_dok,y_init_best))
hold off;
title("y wyznaczone niejawna metoda Eulera")
xlabel("t")
ylabel("populacja")
legend("y dokladne", "y przyblizone")
grid("on")
```

Listing 6: Adam-Bashforth

```
function [x] = AdamB(A,t,y,x_init,fun)
% Funkcja realizujaca metode Adama-Bashfortha
% WEJSCIE:
% A      - wspolczynniki w rownaniu
% t      - wektor czasu
% x_init - wartosc poczatkowa
% fun    - uchwyt do funkcji
% WYJSCIE:
% x      - wektor obliczonych wartosci
x = zeros(length(t),1);
x(1) = x_init;
x(2) = x(1) + fun([x(1),y(1)],A)*(t(2)-t(1));
for i=3:length(t)
    delta_t = t(i)-t(i-1);
    x(i) = x(i-1) + (3/2*fun([x(i-1),y(i-1)],A) ...
        -1/2 * fun([x(i-2),y(i-2)],A))*delta_t;
end
end % function
```

Listing 7: Zad3a

```
% Wspolczynniki i wartosci poczatkowe z zadania 1
A = [10.2943   -0.0630   -0.0228; -5.0148    0.0584   -0.0275];
x1 = 428.4096;
y1 = 47.3206;

dane = readmatrix("dane22.csv");
t = dane(:,1);
x_dok = dane(:,2);
y_dok = dane(:,3);

fms_options = optimset("MaxFunEvals",8000,"MaxIter",8000);

% Uchwyt do funkci dopasowania
J_handle = @(args) J(args,t,x_dok,y_dok);

% Znalezienie najlepszych wspolczynnikow i wartosci poczatkowych
args_opt = fminsearch(J_handle,[A,[x1;y1]],fms_options);

% Rozwiazanie ukladu dla najlepszych wspolczynnikow
odefun = @(t,y) [args_opt(1,1:3)*[y(1);y(1)*y(2);y(1)*y(1)];...
    args_opt(2,1:3)*[y(2);y(1)*y(2);y(2)*y(2)]];
[t_ode, y_ode] = ode45(odefun,[0,3],[args_opt(1,end);args_opt(2,end
    )]);

figure
plot(t, x_dok);
hold on
plot(t_ode,y_ode(:,1));
hold off
title("x wyznaczone z uzyciem ode45 ")
xlabel("t")
ylabel("populacja")
legend("x dokladne", "x przyblizone")
grid("on")

figure
plot(t, y_dok);
hold on
plot(t_ode,y_ode(:,2));
```

```
hold off
title("y wyznaczone z uzyciem ode45 ")
xlabel("t")
ylabel("populacja")
legend("y dokladne", "y przyblizone")
grid("on")

function [err] = J (args,t,x_dok,y_dok)
% funkcja dokladnosci dopasowania modelu
% WEJSCIE:
% args  - wspolczynniki w ukladzie rownan rozniczkowych
% t     - wektor czasu w jakich dokonano dokladnych pomiarow
% x_dok - wektor dokladnych pomiarow populacji ofiar
% y-dok - wektor dokladnych pomiarow populcaji drapieznikow
% WYJSCIE:
% err   - dokladnosc dopasowania modelu

odefun = @(t,y) [args(1,1:3)*[y(1);y(1)*y(2);y(1)*y(1)];...
    args(2,1:3)*[y(2);y(1)*y(2);y(2)*y(2)]];

[t_ode, y_ode] = ode45(odefun,[0,3],[args(1,end);args(2,end)]);

x_przyb = interp1(t_ode,y_ode(:,1),t);
y_przyb = interp1(t_ode,y_ode(:,2),t);

err = sum((x_przyb - x_dok).^2) + sum((y_przyb- y_dok).^2);

end % function
```

Listing 8: Zad3b

```
% Wspolczynniki wyznaczone w zadaniu 2b)
A = [7.0844   -0.0677   -0.0045; -8.3415    0.0591   -0.0011];
x1 = 367.9547;
y1 = 40.0113;

dane = readmatrix("dane22.csv");
t = dane(:,1);
x_dok = dane(:,2);
y_dok = dane(:,3);

fms_options = optimset("MaxFunEvals",8000,"MaxIter",8000);

% Uchwyt do funkci dopasowania modelu do danych
J_handle = @(args) J(args,t,x_dok,y_dok);

% Znalezienie najlepszych parametrow i wartosci poczatkowych
args_opt = fminsearch(J_handle,[A,[x1;y1]],fms_options);

% Rozwiazanie ukladu dla najlepszych wspolczynnikow
[t_eul, y_eul] = ...
    Euler_zad3(args_opt(:,1:3),[0,3],0.01,[args_opt(1,end);args_opt
       (2,end)]);

figure
plot(t, x_dok);
hold on
plot(t_eul,y_eul(1,:));
hold off
title("x wyznaczone z uzyciem jawnej metody eulera ")
xlabel("t")
ylabel("populacja")
```

```
legend("x dokladne", "x przyblizone")
grid("on")

figure
plot(t, y_dok);
hold on
plot(t_eul,y_eul(2,:));
hold off
title("y wyznaczone z uzyciem jawnej metody eulera ")
xlabel("t")
ylabel("populacja")
legend("y dokladne", "y przyblizone")
grid("on")

function [err] = J (args,t,x_dok,y_dok)
% funkcja dokladnosci dopasowania modelu
% WEJSCIE:
% args  - wspolczynniki w ukladzie rownan rozniczkowych
% t     - wektor czasu w jakich dokonano dokladnych pomiarow
% x_dok - wektor dokladnych pomiarow populacji ofiar
% y-dok - wektor dokladnych pomiarow populcaji drapieznikow
% WYJSCIE:
% err   - dokladnosc dopasowania modelu
[t_eul, y_eul] = ...
    Euler_zad3(args(:,1:3),[0,3],0.01,[args(1,end);args(2,end)]);

x_przyb = interp1(t_eul,y_eul(1,:),t);
y_przyb = interp1(t_eul,y_eul(2,:),t);

err = sum((x_przyb - x_dok).^2) + sum((y_przyb- y_dok).^2);

end % function
```

Listing 9: Zad4

```
% Wspolczynniki z zadania 3 a)
A = [11.5414557952839, -0.113284961890873,-0.00173440186216480;...
    -8.03168152879862, 0.0606278266074272,-0.00574306691813943];

syms x y
eq1 = A(1,:)*[x; x*y; x*x] == 0;
eq2 = A(2,:)*[y; x*y; y*y] == 0;

sol = solve([eq1,eq2],x,y);
x_init = round(double(sol.x(2,1)));
y_init = round(double(sol.y(2,1)));

[t_eul, y_eul] = Euler_zad3(A,[0,3],0.001,[x_init, y_init]);

figure()
plot(t_eul,round(y_eul,0,"decimals",TieBreaker="tozero"))
title("Rownowaga ukladu")
legend("populacja ofiar", "populacja drapieznikow");
grid("on")
xlabel("t")
ylabel("populacja")
```

Listing 10: Zad5

```
    HudsonBay = readmatrix("HudsonBay.csv");
t = HudsonBay(:,1);
x_dok = HudsonBay(:,2);
```

```
y_dok = HudsonBay(:,3);
t = t-t(1);
t = 3*t/t(end);

r_xSpan=[0,40];
r_xySpan = [-1,0];
r_xxSpan = [-0.1,1];
x_Span = [10,100];

% Ilosc podprzedzialow
n_r_x = 15;
n_r_xy = 10;
n_r_xx = 10;
n_x = 30;

Ax_best=zeros(1,3); % Najlepsze wspolczynniki w pierwszym rownaniu
x_init_best = 0; % Najlepsza wartosc poczatkowa x
Jx_min = Inf;

for r_x = linspace(r_xSpan(1),r_xSpan(end),n_r_x)
for r_xy = linspace(r_xySpan(1),r_xySpan(end),n_r_xy)
for r_xx = linspace(r_xxSpan(1),r_xxSpan(end),n_r_xx)
for x = linspace(x_Span(1),x_Span(end),n_x)

    A = [r_x,r_xy,r_xx]; % wspolczynniki w rownaniu

    x_przyb = Euler(A,t,y_dok,x,@fun);
    Jx = sum((x_dok - x_przyb).^2); % Blad

    if (Jx<Jx_min)
        Ax_best = A;
        x_init_best = x;
        Jx_min = Jx;
    end

end
end
end
end

% Funkcja bledu
Fun_Jx = @(args)...
    sum((x_dok - Euler(args(1:3),t,y_dok,args(end),@fun)).^2);

% Optymalne argumenty
args_opt = fminsearch(Fun_Jx,[Ax_best,x_init_best]);

Ax_best = args_opt(1:3);
x_init_best = args_opt(end);

% b) ----------------------------
r_ySpan=[-40,0];
r_yxSpan = [0,1];
r_yySpan = [-0.1,0];
y_Span = [10,50];

n_r_y = 15;
n_r_yx = 10;
n_r_yy = 10;
n_y = 30;
```

```
Ay_best=zeros(1,3);
y_init_best = 0;
Jy_min = Inf;

for r_y = linspace(r_ySpan(1),r_ySpan(end),n_r_y)
for r_yx = linspace(r_yxSpan(1),r_yxSpan(end),n_r_yx)
for r_yy = linspace(r_yySpan(1),r_yySpan(end),n_r_yy)
for y = linspace(y_Span(1),y_Span(end),n_y)

    A = [r_y,r_yx,r_yy]; % Wspolczynniki w rownaniu

    y_przyb = Euler(A,t,x_dok,y,@fun);
    Jy = sum((y_dok - y_przyb).^2); % Blad

    if (Jy<Jy_min)
        Ay_best = A;
        y_init_best = y;
        Jy_min = Jy;
    end
end
end
end
end

Fun_Jy = @(args)...
    sum((y_dok - Euler(args(1:3),t,x_dok,args(end),@fun)).^2);

args_opt = fminsearch(Fun_Jy,[Ay_best,y_init_best]);

Ay_best = args_opt(1:3);
y_init_best = args_opt(end);

A = [Ax_best,x_init_best; Ay_best,y_init_best];

fms_options = optimset("MaxFunEvals",8000,"MaxIter",8000);

J_handle = @(args) J(args,t,x_dok,y_dok);

args_opt = fminsearch(J_handle,A,fms_options);

odefun = @(t,y) [args_opt(1,1:3)*[y(1);y(1)*y(2);y(1)*y(1)];...
    args_opt(2,1:3)*[y(2);y(1)*y(2);y(2)*y(2)]];
[t_ode, y_ode] = ode45(odefun,[0,3],[args_opt(1,end);args_opt(2,end
   )]);

figure;
plot(t,x_dok);
hold on
plot(t_ode,y_ode(:,1))
hold off
grid on
title("Proba dopasowania x jawna metoda eulera")
legend("x dokladne", "x przyblizone")
xlabel("t")
ylabel("populacja")

figure;
plot(t,y_dok)
hold on
plot(t_ode,y_ode(:,2))
grid on
```

```
title("Proba dopasowania y jawna metoda eulera")
legend("y dokladne", "y przyblizone")
xlabel("t")
ylabel("populacja")
```

Listing 11: Funkcja dopasowania

```
function [err] = J (args,t,x_dok,y_dok)
% funkcja dokladnosci dopasowania modelu
% WEJSCIE:
% args  - wspolczynniki w ukladzie rownan rozniczkowych i wartosci
%         poczatkowe
% t     - wektor czasu w jakich dokonano dokladnych pomiarow
% x_dok - wektor dokladnych pomiarow populacji ofiar
% y-dok - wektor dokladnych pomiarow populcaji drapieznikow
% WYJSCIE:
% err   - dokladnosc dopasowania modelu
odefun = @(t,y) [args(1,1:3)*[y(1);y(1)*y(2);y(1)*y(1)];...
    args(2,1:3)*[y(2);y(1)*y(2);y(2)*y(2)]];

[t_ode, y_ode] = ode45(odefun,[0,3],[args(1,end);args(2,end)]);

x_przyb = interp1(t_ode,y_ode(:,1),t);
y_przyb = interp1(t_ode,y_ode(:,2),t);

err = sum((x_przyb - x_dok).^2) + sum((y_przyb- y_dok).^2);

end % function
```

# Źródła


[1] https://www.mathworks.com/help/matlab/ref/fminsearch.html

[2] https://www.mathworks.com/help/matlab/ref/ode45.html

[3] https://www.mathworks.com/help/matlab/ref/interp1.html

[4] https://pl.wikipedia.org/wiki/Metoda_Eulera

[5] https://en.wikipedia.org/wiki/Linear_multistep_method

[6] Iwona Wróbel, Notatki do wykładu Metody Numeryczne 2

[7] A. Lotka, "Analytical note on certain rhythmic relations in organic systems," Proceedings of the National Academy of Sciences, vol. 6, pp. 410–415, 1920.