11.12.2023 Warszawa

Marcin Gronicki

Politechnika Warszawska  
Wydział Matematyki i Nauk Informacyjnych

Zadanie projektowe nr 2: Estymacja parametrów modelu Lotki-Volterry  
semestr zimowy 2023/24

Modelowanie Matematyczne dr hab. inż. Kajetana Snopek prof. uczelni

Spis treści

[1. Symbole matematyczne i akronimy 2](#_Toc153357618)

[2. Wprowadzenie 3](#_Toc153357619)

[2.1 Równanie Lotki-Volterry 3](#_Toc153357620)

[2.2 Jawna metoda Eulera 3](#_Toc153357621)

[2.3 Niejawna metoda Eulera 3](#_Toc153357622)

[2.4 Metoda Heuna 4](#_Toc153357623)

[2.5 Wbudowana w środowisko Matlab funkcja ode45 4](#_Toc153357624)

[3. Metodyka i wyniki doświadczeń 4](#_Toc153357625)

[3.1 Wyznaczenie optymalnych parametrów dopasowania modelu do danych z wykorzystaniem jawenej metody eulera 4](#_Toc153357626)

[3.2 Wyznaczenie optymalnych parametrów dopasowania modelu do danych z wykorzystaniem niejawnej metody Eulera i metody Heuna 5](#_Toc153357627)

[3.3 Wyznaczenie optymalnych parametrów dopasowania modelu do danych z wykorzystaniem ode45 i jawnej metody Eulera. 5](#_Toc153357628)

[3.4 Wyznaczanie wartości x, y dla których populacje osiągają stan równowagi. 5](#_Toc153357629)

[4. Dyskusja wyników eksperymentów numerycznych 6](#_Toc153357630)

[4.1 Wyniki wyznaczenia optymalnych parametrów dopasowania modelu do danych z wykorzystaniem jawenej metody eulera. 6](#_Toc153357631)

[4.2 Wyniki wyznaczenia optymalnych parametrów dopasowania modelu do danych z wykorzystaniem niejawnej metody Eulera i metody Heuna 7](#_Toc153357632)

[4.3 Wyniki wyznaczenia optymalnych parametrów dopasowania modelu do danych z wykorzystaniem ode45 i jawnej metody Eulera. 10](#_Toc153357633)

[4.4 Znalezione wartości x, y dla których populacje osiągają stan rówonowagi. 10](#_Toc153357634)

# Symbole matematyczne i akronimy

# 2. Wprowadzenie

## Równanie Lotki-Volterry

Celem tego projektu jest znalezienie optymalnych parametrów układu równań Lotki-Volterry (1), dla których model zdefiniowany równaniem (1), w najbardziej dokładny sposób będzie odwzorowywał liczność populacji x i y.

Równanie

Równanie Lotki-Volterry jest układem dynamicznym, używanym do symulowania liczności populacji ofiar i drapieżników w danym ekosystemie.

W niniejszym projekcie zadanie znalezienia optymalnych parametrów zostanie zaimplementowane przy użyciu różnych metod rozwiązywania układów równań różniczkowych, zaimplementowanych przez autora a także dostępnych w środowisku matlab, w początkowych eksperymentach metody zostaną sprawdzone na danych syntetycznych, a w końcowym eksperymencie zostaną sprawdzone na danych obrazujących rzeczywisty ekosystem zatoki Hudson.

## Jawna metoda Eulera

Metoda jawna Eulera jest jedną z prostszych technik numerycznego rozwiązywania równań różniczkowych. Jej idea opiera się na przybliżeniu wartości funkcji pochodnej. Użyta w tym projekcie jawna metoda Eulera wyraża się wzorem (1)

Wzór

Gdzie oznacza uzyskane w n-tym kroku przybliżenie liczności populacji x lub y, a oznacza dokładną zmierzoną wartość populacji x lub y w n-tym kroku, oznacza krok czasowy.

Metoda Eulera jest stosunkowo prosta w implementacji, ale jej dokładność jest niska, szczególnie dla dużych wartości kroku czasowego.

## Niejawna metoda Eulera

Metod niejawna Eulera jest zmodyfikowaną wersją jawnej metody Eulera, gdzie różnica polega na wykorzystywaniu obliczanej wartości w momencie jej obliczania, w przypadku implementacji tej metody w tym programie skorzystałem z faktu możliwości przedstawienia wzoru (2), jako funkcji kwadratowej i obliczeniu jej pierwiastków a następnie wykorzystaniu większego.

Wzór

## 2.4 Metoda Heuna

Metoda Heuna jest ulepszoną metodą Eulera, do jej zaimplementowania posłużą wzory (3), (4), (5), (6)

Wzór

Wzór

Wzór

## Wbudowana w środowisko Matlab funkcja ode45

Funkcja ode45 znajduje numerycznie rozwiązania równań różniczkowych, opiera się na algorytmie Rungego-Kutty rzędu (4,5). Ode45 przyjmuje uchwyt do funkcji reprezentującej równanie/układ równań różniczkowych, tspan – wektor rozmiaru 2 zawierającą czas początkowy i końcowy, a następnie y0 – wektor warunków początkowych. Jako wynik zwraca wektor o rozmiarze 2 zawierający t – wektor próbek czasu i y – wektor wyliczonych wartości w chwili odpowiedniej próbki z t.

# 3. Metodyka i wyniki doświadczeń

## 3.1 Wyznaczenie optymalnych parametrów dopasowania modelu do danych z wykorzystaniem jawenej metody eulera

Początkowym eksperymentem była próba znalezienia optymalnych parametrów minimalizujących wzór (6) gdzie oznacza dokładną zmierzoną liczność populacji.

Wzór

Następnie znalezienie parametrów minimalizujących wzór (7).

Wzór

W celu znalezienia tych parametrów przeszukałem przy użyciu wbudowanej w Matlab R2023b funkcji *combinations* generującej kombinacje punktu startowego i parametrów , gdzie z podanych przedziałów wybrano dla 50, 20, 10, 20 wartości położonych w odstępach liniowych. Następnie wybrałem kombinację dającą najmniejszą wartość i wywołałem dla tej kombinacji elementów wbudowaną w Matlab funkcję *fimnsearch* która znajduje parametry dające najmniejszą wartość funkcji.

Analogicznie postąpiłem dla populacji y, z tym że parametry startowe dla funkcji *fminsearch* zostały wybrane z przedziałów i parametrów . Następne kroki zostały wykonane w sposób identyczny jak w przypadku populacji x, oprócz tego że funkcja *fminsearch* znajdowała optymalne rozwiązanie dla .

## 3.2 Wyznaczenie optymalnych parametrów dopasowania modelu do danych z wykorzystaniem niejawnej metody Eulera i metody Heuna

W następnym etapie projektu w celuz znalezienia optymalnych parametrów została wykorzystana niejawna metoda Eulera (2.3) i metoda Heuna (2.4). W celu wykonania tego zadania procedura była analogiczna do tej zdefiniowanej w podrozdziale (3.1). Jedyną istotną różnicą było zastępienie jawnej metody Eulera, metodą niejawną i Heuna. W celu implementacji niejawnej metody Eulera została ona przekształcona do postaci funkcji kwadratowej odpowiednio wzór (8) i (9). Następnie znalazłem miejsca zerowe tych funkcji przy użyciu zaaimplementowanej przeze mnie funkcji *findRoot* (obliczającą delte, a następnie pierwiastki z wykorzystaniem wzoru Vieta). W każdej iteracji program poszukiwał miejsc zerowych a następnie wybierał większe.

Wzór

Wzór

## 3.3 Wyznaczenie optymalnych parametrów dopasowania modelu do danych z wykorzystaniem ode45 i jawnej metody Eulera.

W tym eksperymenice wyznaczyłem parametry które są jednocześnie optymalne dla populacji x i y, to znaczy minimalizujace wzór (10) . Wykorzystałem wyznaczone w pierwszym eksperymencie wartości startowe dla optymalizacji wzorów (6) i (7).

Wzór

Następnie zaimplementowałem metody rozwiązywaia URRZ: metodę ode45 i jawną metodę Eulera dla których uruchomiłem optymalizacje przy użyciu *fminsearch.* Dodatkowo funkcje implementujące poprzednio wspomniane metody numeryczne rozszerzyłem o interpolowanie uzyskanych przez te metody wyników w punktach . Interpolacje wykonowałem używając wbudowanej w środowisko funkcji *interp1.*

## 3.4 Wyznaczanie wartości x, y dla których populacje osiągają stan równowagi.

Celeme niniejszego eksperymentu było oblicznenie liczności populacji x i y > 0 dla których układ osiąga stan równowagi (tzn. x(t) i y(t) nie zmienia się w czasie). W tym celu użyłem funkcji *solve* znajdującej się w Matlab Symbolic toolbox, która rozwiązuje układy równań. Jednka najpierw nie zbędne było zmodyfikowanie równania (1), poprzez podstawienie za obie pochodne 0, a pod parametry r podstawiłem optymalne wartości znalezione w zadaniu 3. Tak zmodyfikowane równanie zostało rozwiązane symbolicznie przy użyciy *solve*.

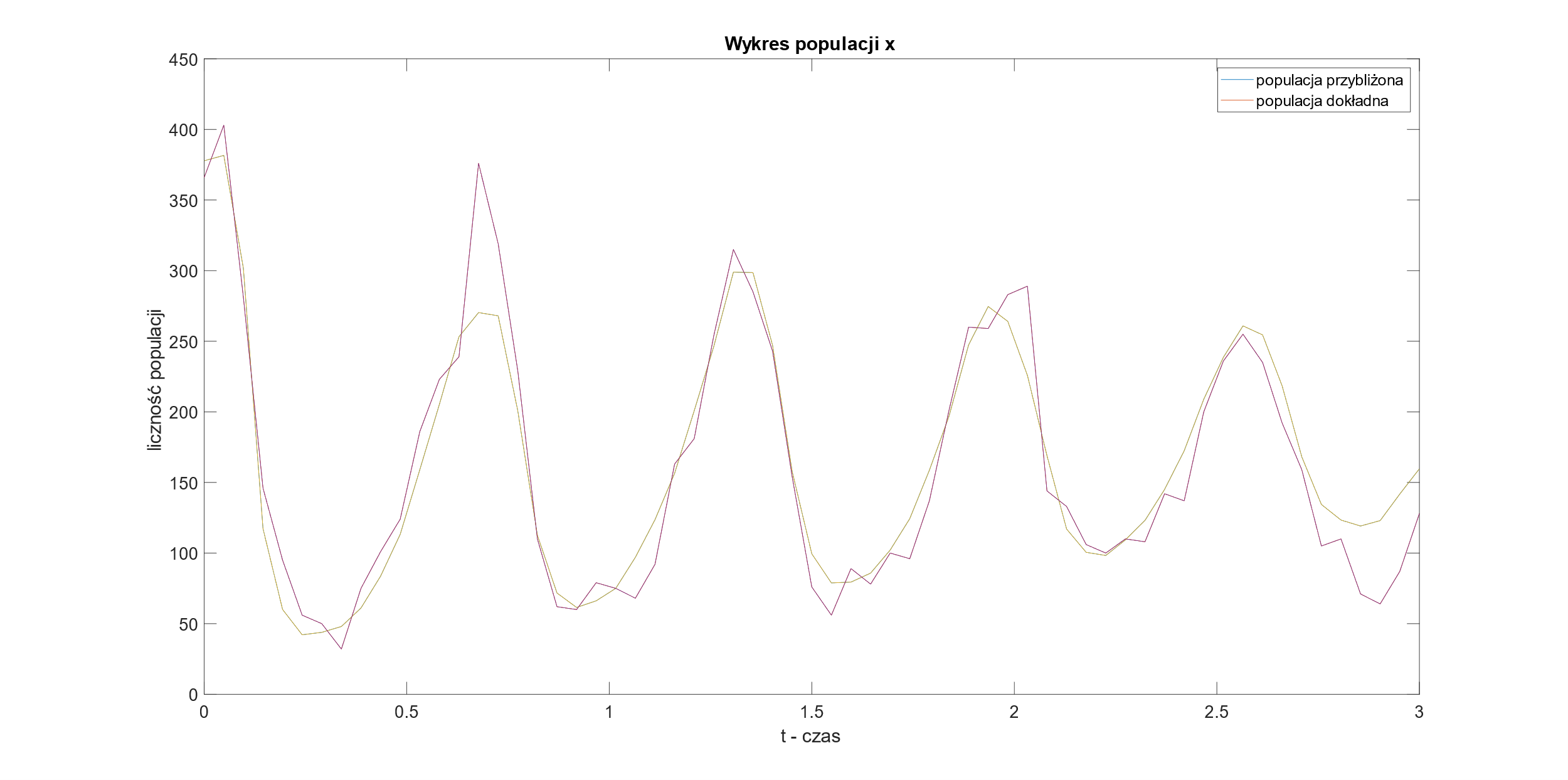
# 4. Dyskusja wyników eksperymentów numerycznych

## 4.1 Wyniki wyznaczenia optymalnych parametrów dopasowania modelu do danych z wykorzystaniem jawenej metody eulera.

W niniejszym podrozdziale przedstawię otrzymane parametry dla populacji x i y, a także zobrazuję ich jakość na wykresie (1) i (2), w porówaniu do dokładnych zmierzonych liczności populacji.

Dla populacji x otrzymałem wartości parametrów

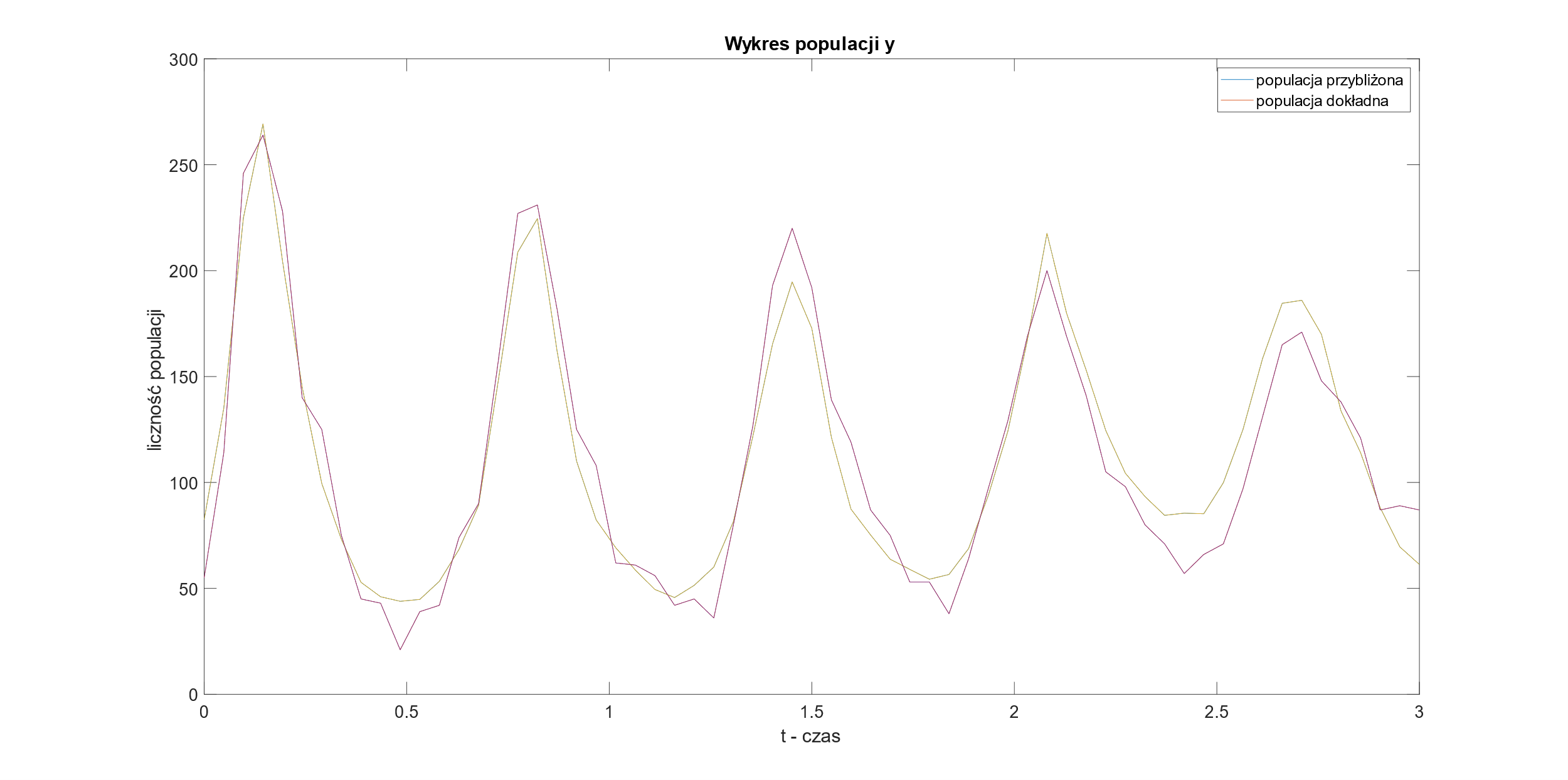
I minimalną wartość



Wykres

Dla populacji y otrzymałem parametry:

Natomiast minimalna wartość wyniosła 1.7088e+04.

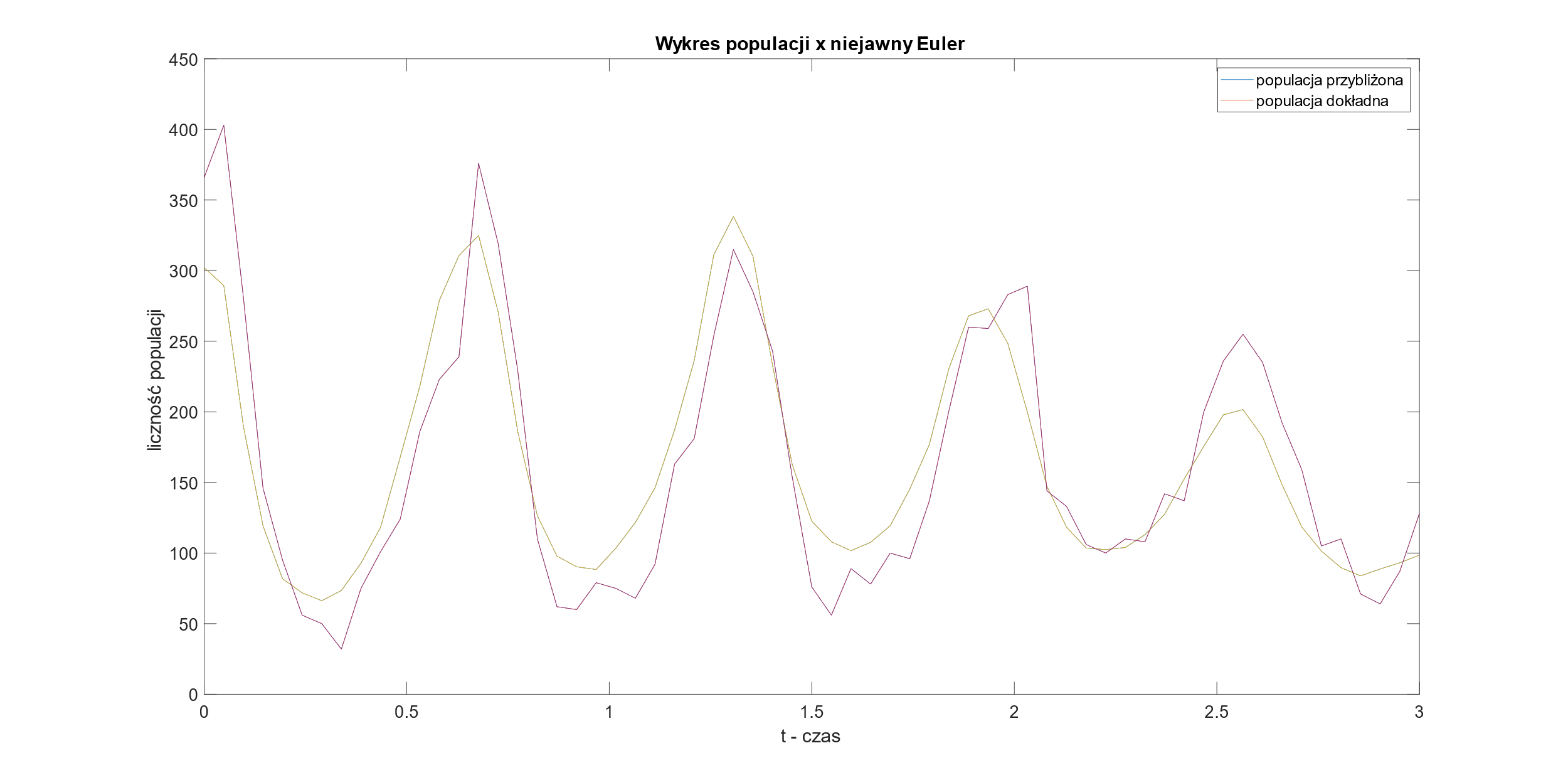


Wykres

## 4.2 Wyniki wyznaczenia optymalnych parametrów dopasowania modelu do danych z wykorzystaniem niejawnej metody Eulera i metody Heuna

W tej części raportu przedstawię optymalne wartości parametrów uzyskanych przy użyciu niejawnej metody Eulera i Heuna. W obu przypadkach niezbędna była minimalizacja wzorów (6) i (7). W przypadku obliczeń wykonywanych przy użyciu niejawnej metody Eulera otrzymałem dla populacji x następujące wartości parametrów:

a wartość dla tych parametrów wyniosła 9.9245e+04. Na wykresie (3) jest przedstawiona liczność populacji x uzyskana przy pomocy przybliżonej metody w porównaniu do dokładnych wartości.

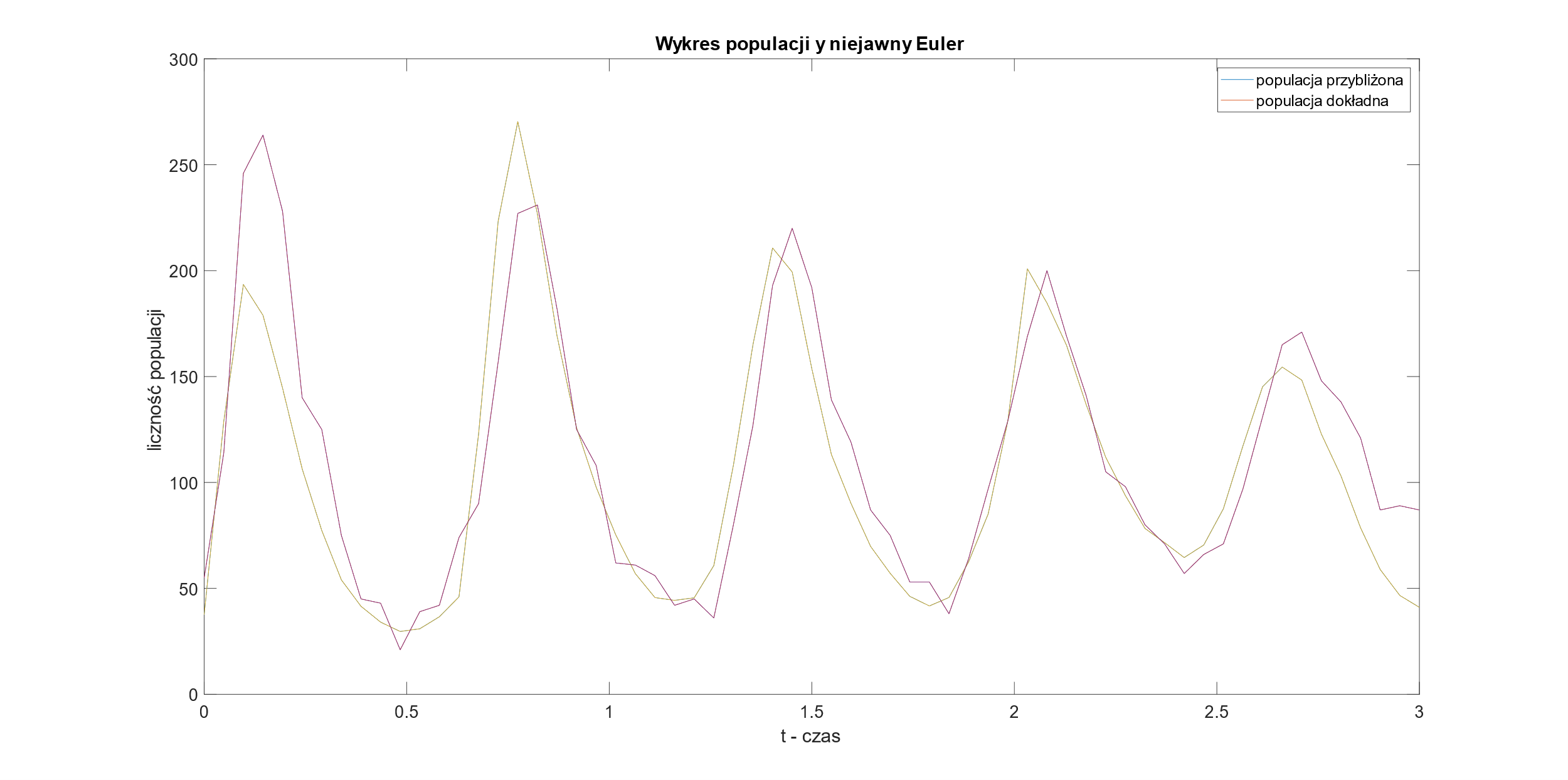


Wykres

Dla populacji y optymalne parametry wyniosły:

Z kolei minimalna wartość współczynnika wyniosła 9.9245e+04

Wykres (4) przedstawia uzyskany przybliżony przebieg zmienność populacji w porównaniu do dokładnej liczności populacji y.

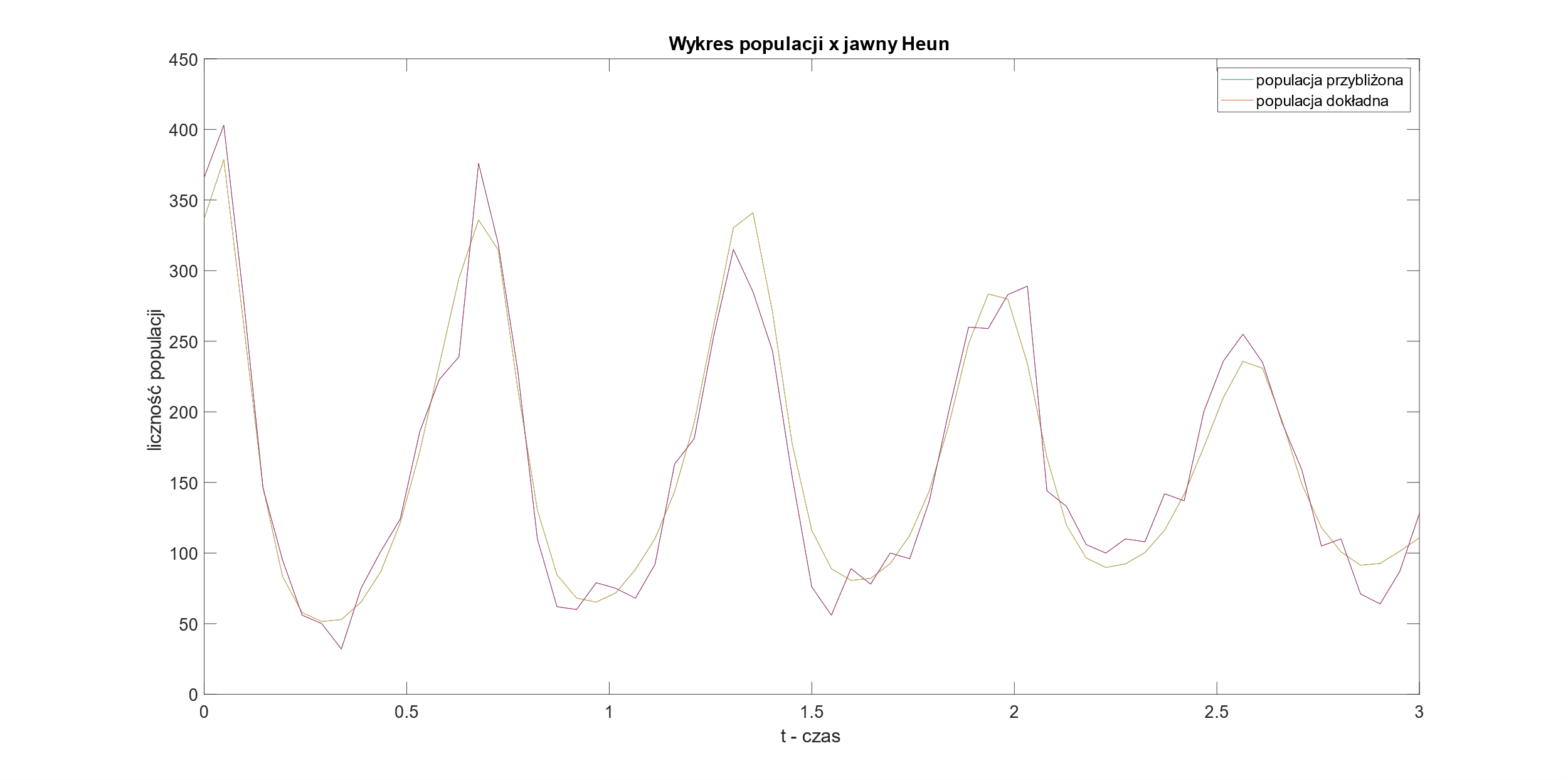


Wykres

Następnie wyznaczyłem optymalne parametry wykorzystując metodę Heuna. Dla populacji x optymalne parametry wyniosły:

Natomiast wartość współczynnika wyniosła 2.7790e+04

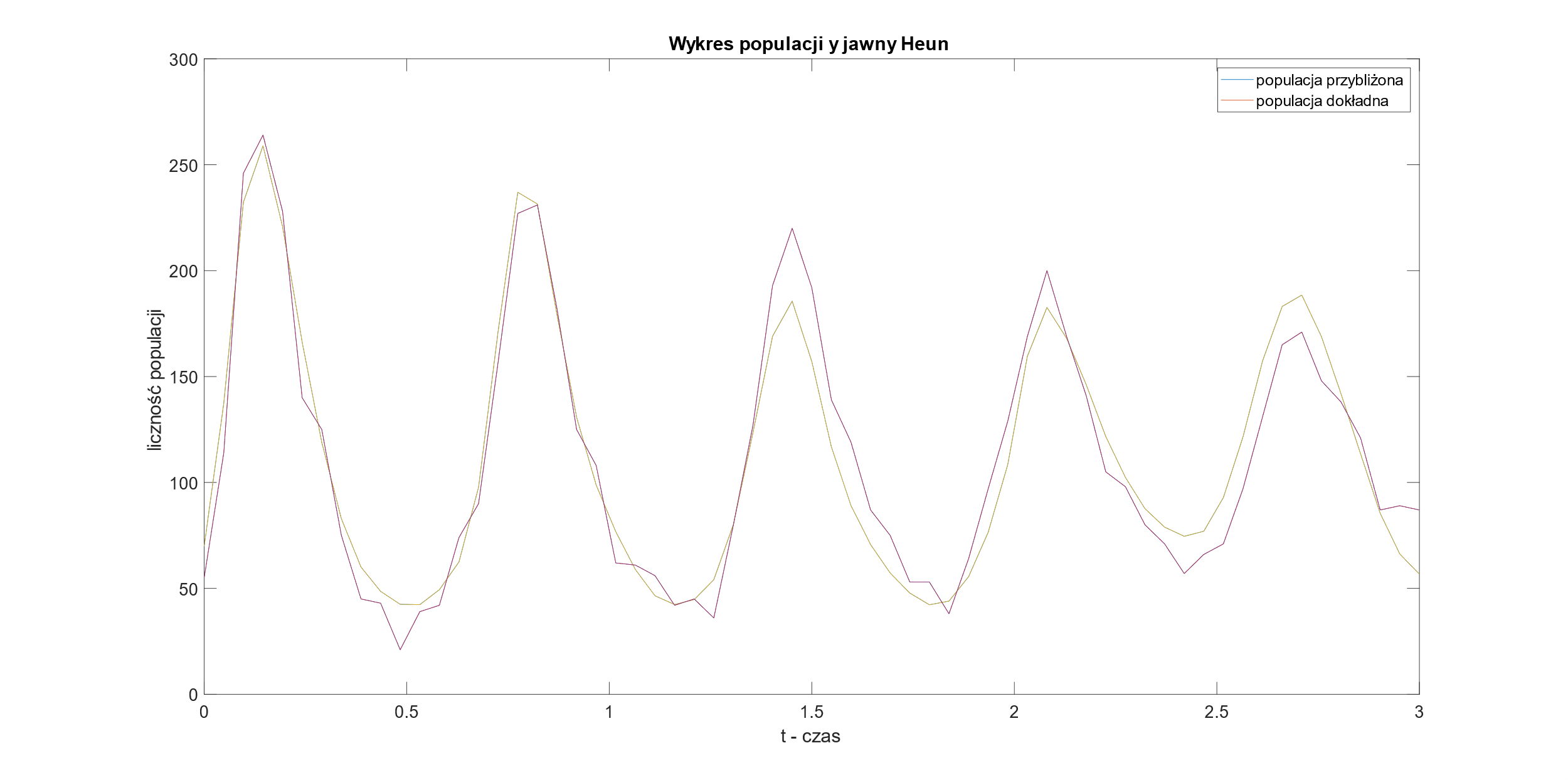
Na wykresie (5) został przedstawiony wynik w analogiczny do poprzednich wykresów sposób.



Wykres

Optymalne parametry dla modelu populacji y wyniosły:

a minimalna wartość współczynnika wyniosła 1.5550e+04. Wykres (6) przedstawia populację y wyznaczoną metodą Heuna.

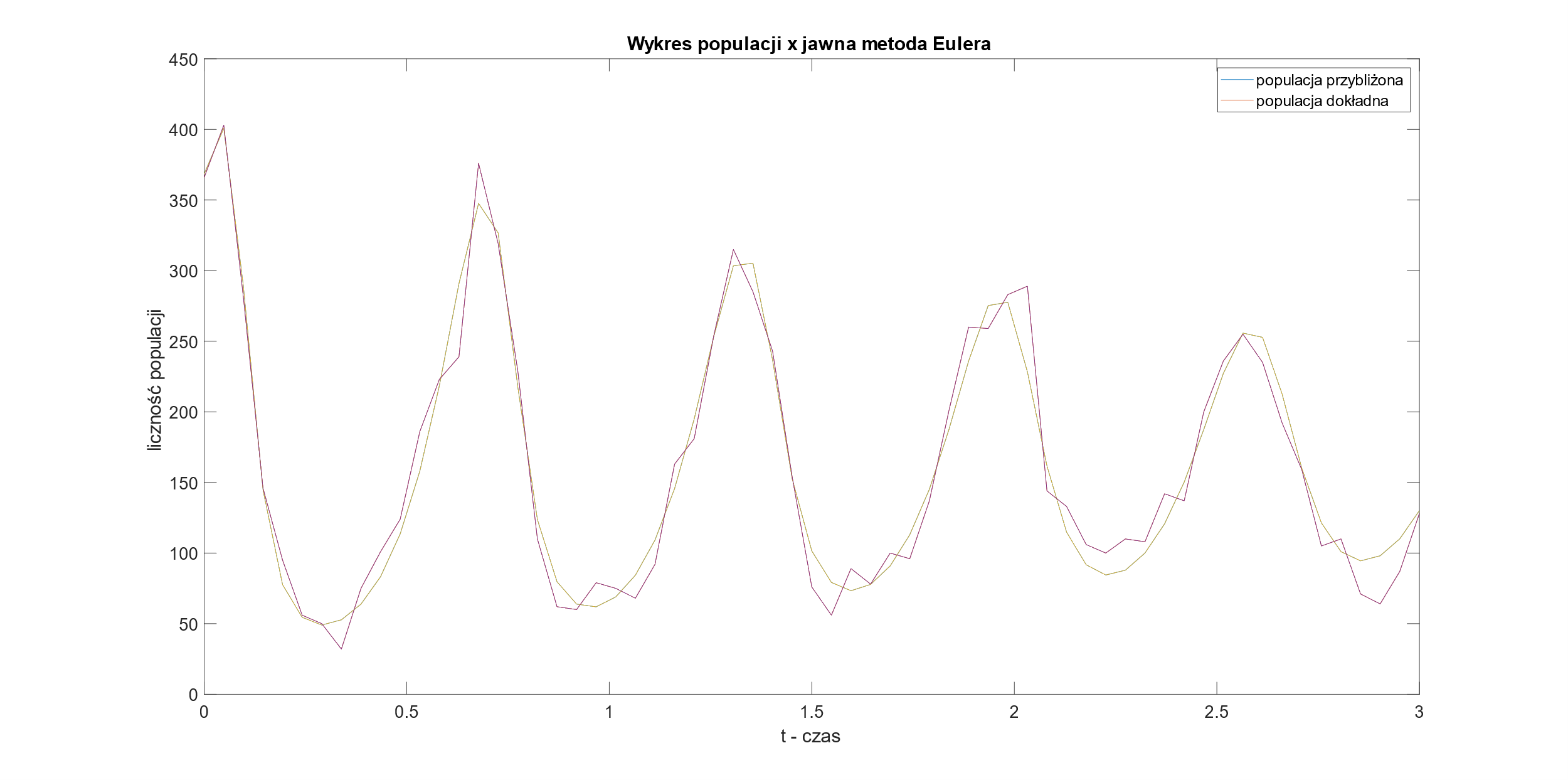


Wykres

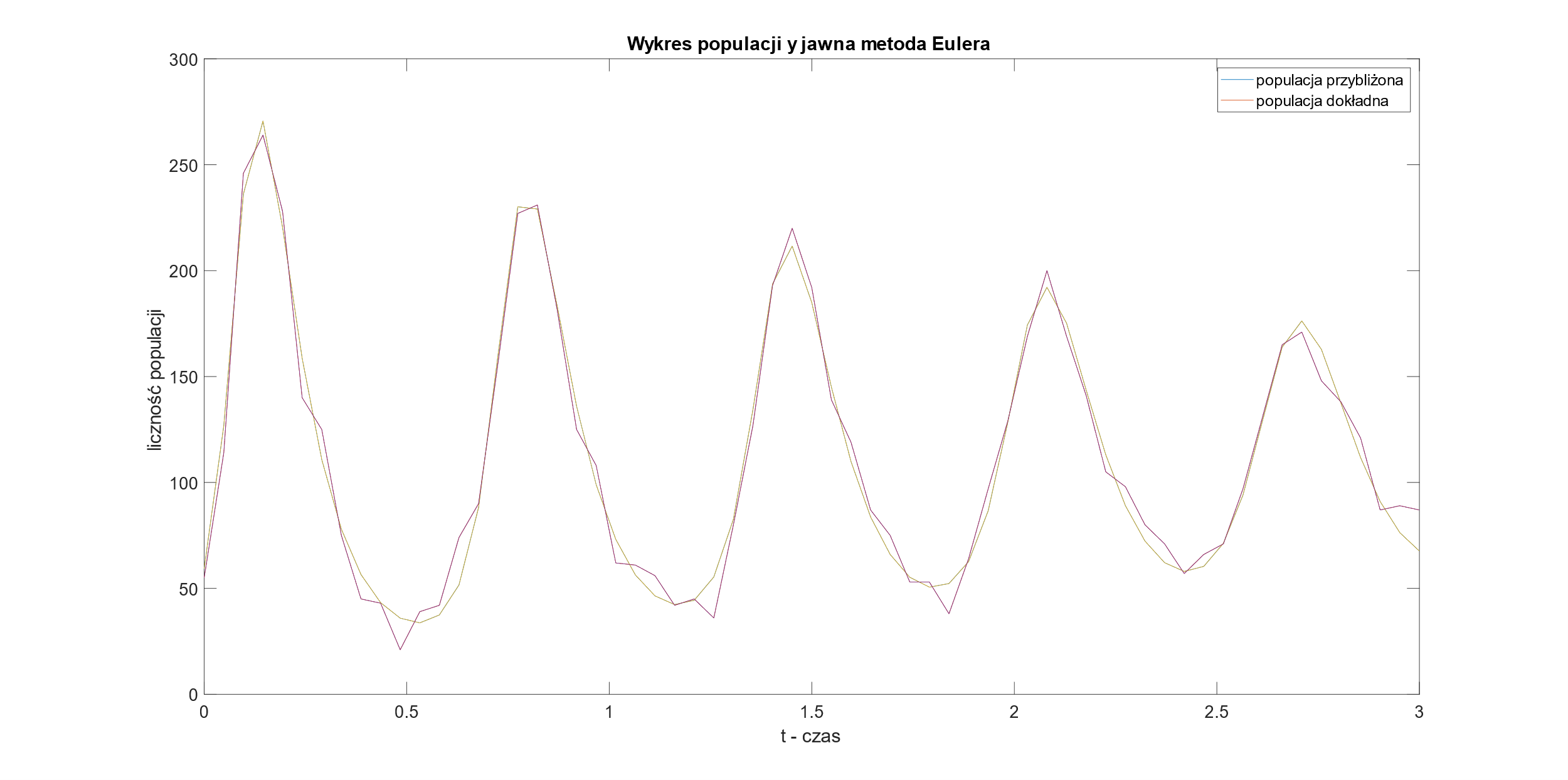
## 4.3 Wyniki wyznaczenia optymalnych parametrów dopasowania modelu do danych z wykorzystaniem ode45 i jawnej metody Eulera.

W tym rozdziale przedstawie otrzymane wartości parametrów gdzie do obliczeń wykorzystałem wbudowaną funkcje w matlab *ode45* i jawną metodę Eulera. W przypadku pierwszej funkcji otrzymałem wartości wszystkich parametrów:

Wartość wyniosła 8.6068e+04, a wykres (7) i (8) przedstawia odpowiednio populacje x i y uzyskane przy użyciu ode45.



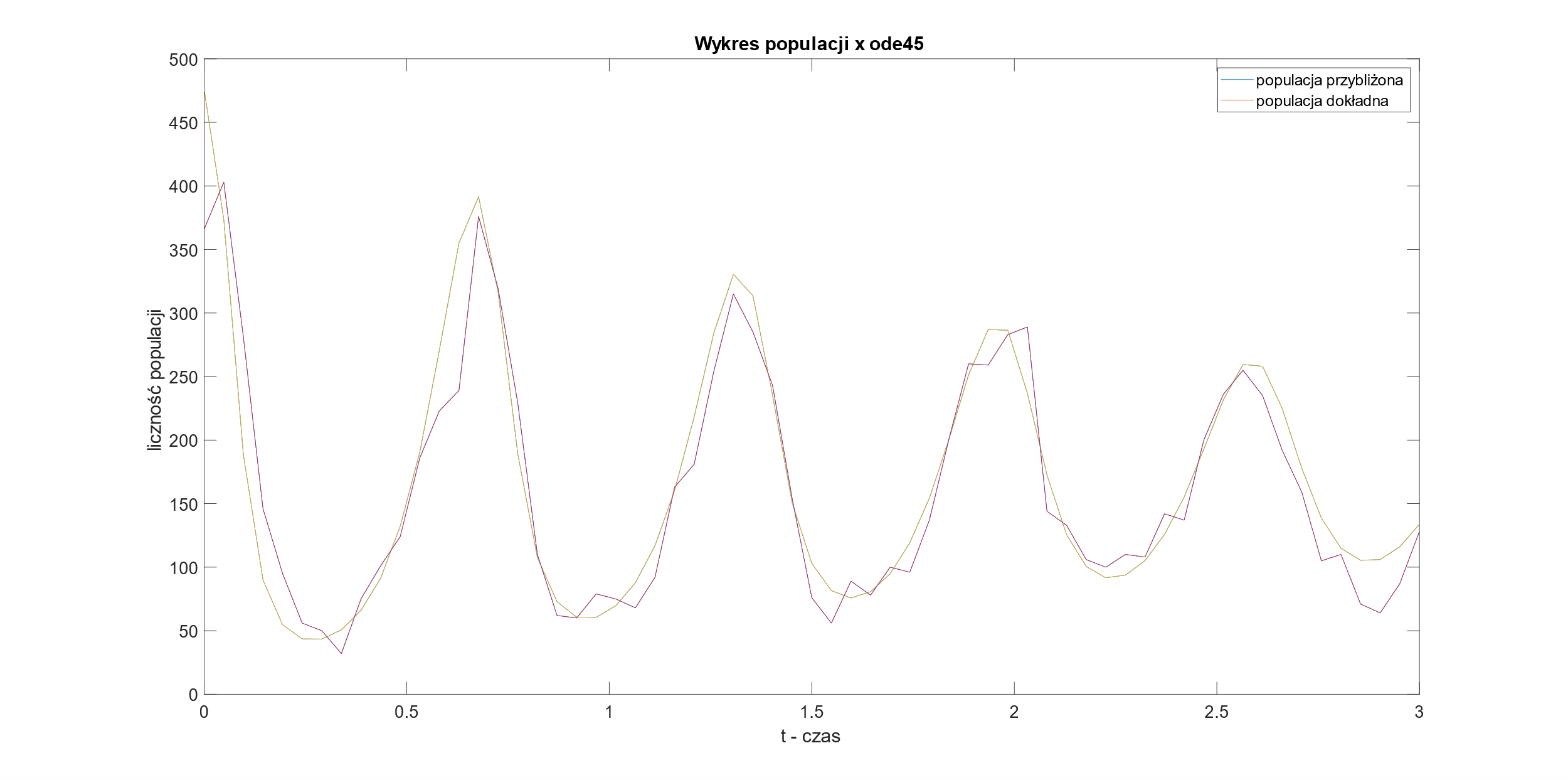
Wykres



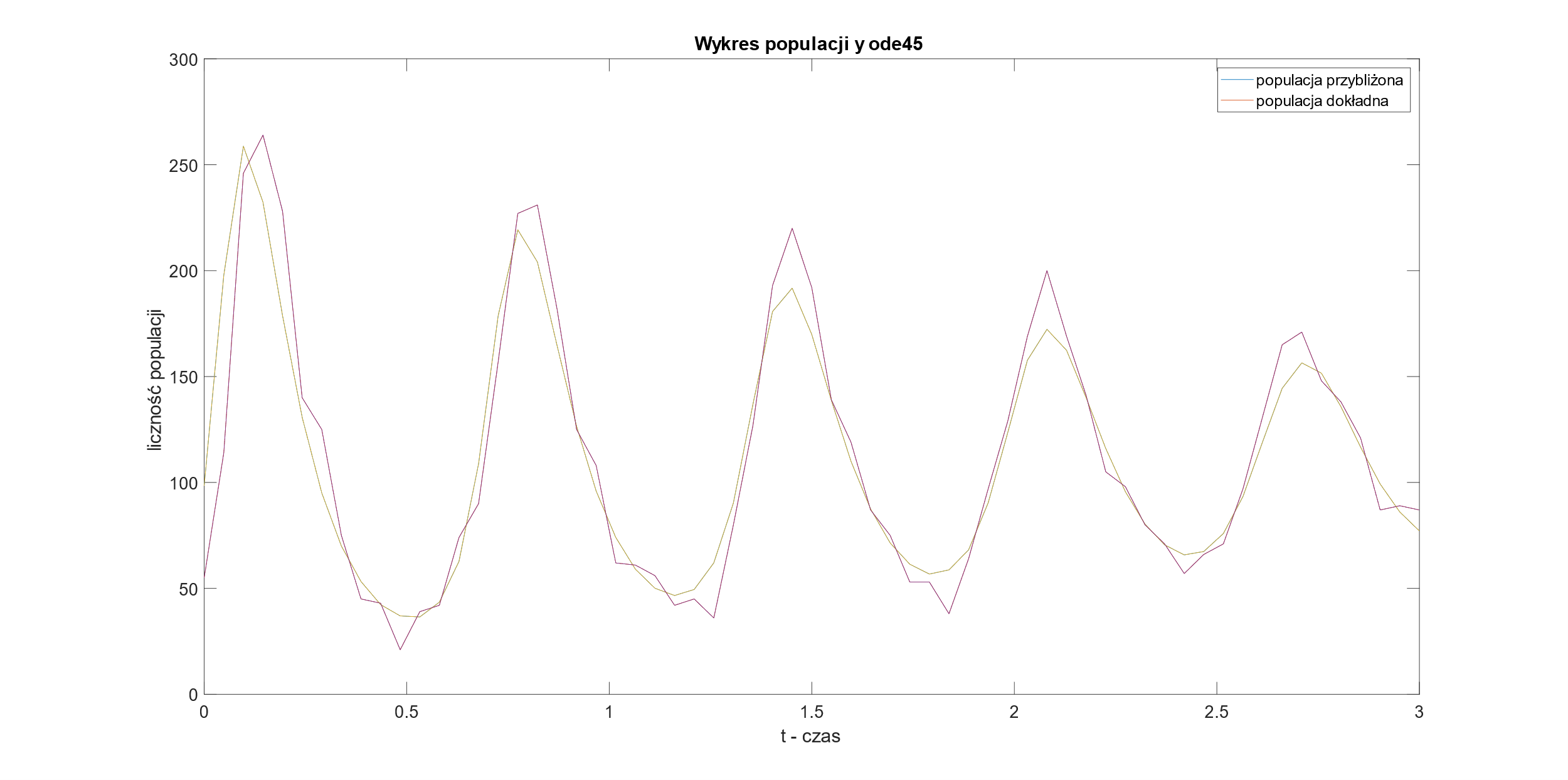
Wykres

Wykorzystując jawną metodę Eulera otrzymałem parametry:

Wartość wyniosła 2.5538e+04, a wykres (9) i (10) przedstawia odpowiednio populacje x i y uzyskane przy użyciu jawnej metody Eulera.



Wykres



Wykres

## 4.4 Znalezione wartości x, y dla których populacje osiągają stan rówonowagi.

Postępując zgodnie z instrukcjami przedstawionymi w rozdziale 3.4 otrzymałem 4 pary rozwiązań, z czego tylko jedna para miała dla x i y wartości większe od zera. Dla x szukana liczność populacji wyniosła , a dla populacji y wyniosła .

Możemy podać odpowiednie interpretacje parametrów w kontekście Lotki-Volterra:

* ​: Wzrost naturalny populacji ofiary (αα w klasycznym Lotka-Volterra).
* ​: Wpływ drapieżnika na ofiarę (czynnik ograniczający populację ofiary z powodu drapieżnictwa).
* ​: Interakcje między osobnikami w obrębie populacji ofiary (samoregulacja populacji ofiary).
* ​: Utrata energii przez drapieżnika (czynnik redukujący liczebność drapieżnika).
* ​: Wpływ ofiary na drapieżnika (czynnik zwiększający liczebność drapieżnika w wyniku dostępu do pokarmu).
* ​: Interakcje między osobnikami w obrębie populacji drapieżnika (samoregulacja populacji drapieżnika).

## 4.5 Optymalne wartości parametrów dla populacji zajęcy i rysiów w ekosystemie Hudson Bay na przestrzeni lat 1845-1935.

# 5. Wnioski

# 6. Źródła

# 7. Listing Programów

W rozwiązaniu nie ma programu głównego, do każdego zadania do realizacji danej metody stworzyłem jeden skrypt plus funkcje pomocnicze.

## 7.1 Lisiting zadanie 1

W przypadku zadanie pierwszego plik zad1v2.m był skryptem głównym i korzystał z funkcji pomocniczych zad1Euler.m zad1EulerY.m implementującą metodę jawną Eulera.

zad1v2.m

clear

% Wczytanie danych

dane = readtable("dane16.csv");

tDane = double(dane.t);

xDane = double(dane.x);

yDane = double(dane.y);

% Początkowe Parametry

rx = linspace(0,40,20);

ry = linspace(-1,0,10);

rxx = linspace(-0.1,0,20);

x0 = linspace(100,1000,50);

ry2 = linspace(-40,0,20);

ryx = linspace(0,1,10);

ryy = linspace(-0.1,0,20);

y0 = linspace(10,200,50);

combs = combinations(x0,rx,ry,rxx);

combs = table2array(combs);

combs2 = combinations(y0,ry2,ryx,ryy);

combs2 = table2array(combs2);

% Funkckcja do minimalizacji

Jx = @(xEuler) sum((xEuler - xDane).^2);

Jx2 = @(xEuler) sum((xEuler - yDane).^2);

sols = zeros(length(rx),1);

sols2 = zeros(length(ry2),1);

for i=1:length(combs)

sols(i) = Jx(zad1Euler(combs(i,1),tDane,yDane,combs(i,2:end)));

sols2(i) = Jx2(zad1EulerY(combs2(i,1),tDane,xDane,combs2(i,2:end)));

end

minimum = min(sols);

Rminimum = combs(sols == minimum,:);

tempFunc = @(x) Jx(zad1Euler(x(1),tDane,yDane,x(2:end)));

[opt\_PopX,val] = fminsearch(tempFunc,Rminimum)

minimum2 = min(sols2);

Rminimum2 = combs2(sols2 == minimum2,:);

tempFunc2 = @(x) Jx2(zad1EulerY(x(1),tDane,xDane,x(2:end)));

[opt\_PopY,val] = fminsearch(tempFunc2,Rminimum2)

figure(1)

plot(tDane,zad1Euler(opt\_PopX(1),tDane,yDane,opt\_PopX(1,2:end)))

hold on

plot(tDane,xDane)

title("Wykres populacji x")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

figure(2)

plot(tDane,zad1EulerY(opt\_PopY(1),tDane,xDane,opt\_PopY(1,2:end)))

hold on

plot(tDane,yDane)

title("Wykres populacji y")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

zad1Euler.m:

function xVect = zad1Euler(x0,t,y,R)

xn = zeros(length(t),1);

xn(1) = x0;

rx = R(1);

rxy = R(2);

rxx = R(3);

f = @(x,y) rx \* x + rxy \* x \* y + rxx \* x \* x;

for i=2:length(t)

tn = t(i) - t(i - 1);

xn(i) = xn(i-1) + f(xn(i-1),y(i-1)) \* tn;

end

xVect = xn;

function yVect = zad1EulerY(y0,t,x,R)

yn = zeros(length(t),1);

yn(1) = y0;

ry = R(1);

ryx = R(2);

ryy = R(3);

f = @(x,y) ry \* y + ryx \* x \* y + ryy \* y \* y;

for i=2:length(t)

tn = t(i) - t(i - 1);

yn(i) = yn(i-1) + f(x(i-1),yn(i-1)) \* tn;

end

yVect = yn;

## 7.2 Lisiting zadanie 2

Zadanie drugie realizują dwa skrypty zad2EulerWykres.m i zad2HeunWykres.m. Gdzie pierwszy skrypt wykorzystuje funkcje findRoots i zad2ImplictEulerY.m, a drugi funkcje Heun.m i HeunY.m

zad2EulerWykres.m:

clear

% Wczytanie danych

dane = readtable("dane16.csv");

tDane = double(dane.t);

xDane = double(dane.x);

yDane = double(dane.y);

% Początkowe Parametry

rx = linspace(0,40,20);

ry = linspace(-1,0,10);

rxx = linspace(-0.1,0,20);

x0 = linspace(100,1000,50);

ry2 = linspace(-40,0,20);

ryx = linspace(0,1,10);

ryy = linspace(-0.1,0,20);

y0 = linspace(10,200,50);

combs = combinations(x0,rx,ry,rxx);

combs = table2array(combs);

combs2 = combinations(y0,ry2,ryx,ryy);

combs2 = table2array(combs2);

% Funkckcja do minimalizacji

Jx = @(xEuler) sum((xEuler - xDane).^2);

Jx2 = @(xEuler) sum((xEuler - yDane).^2);

sols = zeros(length(rx),1);

sols2 = zeros(length(ry2),1);

for i=1:length(combs)

sols(i) = Jx(zad2implicitEulerY(combs(i,1),tDane,yDane,combs(i,2:end)));

sols2(i) = Jx2(zad2implicitEulerY(combs2(i,1),tDane,xDane,combs2(i,2:end)));

end

options = optimset('fminsearch');

options.MaxIter = 4000;

options.MaxFunEvals = 5000;

minimum = min(sols);

Rminimum = combs(sols == minimum,:);

tempFunc = @(x) Jx(zad2implicitEulerY(x(1),tDane,yDane,x(2:end)));

[opt\_PopX,fval] = fminsearch(tempFunc,Rminimum,options)

minimum2 = min(sols2);

Rminimum2 = combs2(sols2 == minimum2,:);

tempFunc2 = @(x) Jx2(zad2implicitEulerY(x(1),tDane,xDane,x(2:end)));

[opt\_PopY,fval] = fminsearch(tempFunc2,Rminimum2,options)

figure(1)

plot(tDane,zad2implicitEulerY(opt\_PopX(1),tDane,yDane,opt\_PopX(1,2:end)))

hold on

plot(tDane,xDane)

title("Wykres populacji x niejawny Euler")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

figure(2)

plot(tDane,zad2implicitEulerY(opt\_PopY(1),tDane,xDane,opt\_PopY(1,2:end)))

hold on

plot(tDane,yDane)

title("Wykres populacji y niejawny Euler")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

zad2implicitEulerY.m:

function xVect = zad2implicitEulerY(x0,t,y,R)

xn = zeros(length(t),1);

xn(1) = x0;

rx = R(1);

rxy = R(2);

rxx = R(3);

for i=2:length(t)

tn = t(i) - t(i - 1);

[x1, x2] = findRoots([rxx \* tn, rx \* tn + rxy \* y(i) \* tn...

- 1, xn(i-1)]);

xn(i) = max(x1,x2);

end

xVect = xn;

findRoots.m

function xVect = zad2implicitEulerY(x0,t,y,R)

xn = zeros(length(t),1);

xn(1) = x0;

rx = R(1);

rxy = R(2);

rxx = R(3);

for i=2:length(t)

tn = t(i) - t(i - 1);

[x1, x2] = findRoots([rxx \* tn, rx \* tn + rxy \* y(i) \* tn...

- 1, xn(i-1)]);

xn(i) = max(x1,x2);

end

xVect = xn;

zad2HeunWykres.m:

clear

% Wczytanie danych

dane = readtable("dane16.csv");

tDane = double(dane.t);

xDane = double(dane.x);

yDane = double(dane.y);

% Początkowe Parametry

rx = linspace(0,40,20);

ry = linspace(-1,0,10);

rxx = linspace(-0.1,0,20);

x0 = linspace(100,1000,50);

ry2 = linspace(-40,0,20);

ryx = linspace(0,1,10);

ryy = linspace(-0.1,0,20);

y0 = linspace(10,200,50);

combs = combinations(x0,rx,ry,rxx);

combs = table2array(combs);

combs2 = combinations(y0,ry2,ryx,ryy);

combs2 = table2array(combs2);

% Funkckcja do minimalizacji

Jx = @(xEuler) sum((xEuler - xDane).^2);

Jx2 = @(xEuler) sum((xEuler - yDane).^2);

sols = zeros(length(rx),1);

sols2 = zeros(length(ry2),1);

for i=1:length(combs)

sols(i) = Jx(Heun(combs(i,1),tDane,yDane,combs(i,2:end)));

sols2(i) = Jx2(HeunY(combs2(i,1),tDane,xDane,combs2(i,2:end)));

end

minimum = min(sols);

Rminimum = combs(sols == minimum,:);

tempFunc = @(x) Jx(Heun(x(1),tDane,yDane,x(2:end)));

[opt\_PopX,fval] = fminsearch(tempFunc,Rminimum)

minimum2 = min(sols2);

Rminimum2 = combs2(sols2 == minimum2,:);

tempFunc2 = @(x) Jx2(HeuYn(x(1),tDane,xDane,x(2:end)));

[opt\_PopY,fval] = fminsearch(tempFunc2,Rminimum2)

figure(1)

plot(tDane,Heun(opt\_PopX(1),tDane,yDane,opt\_PopX(1,2:end)))

hold on

plot(tDane,xDane)

title("Wykres populacji x jawny Heun")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

figure(2)

plot(tDane,HeunY(opt\_PopY(1),tDane,xDane,opt\_PopY(1,2:end)))

hold on

plot(tDane,yDane)

title("Wykres populacji y jawny Heun")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

Heun.m:

function vect = Heun(x0,t,ydok,R)

xn = zeros(length(t),1);

xn(1) = x0;

rx = R(1);

rxy = R(2);

rxx = R(3);

f = @(x,y) rx \* x + rxy \* x \* y + rxx \* x \* x;

for i=2:length(t)

tn = t(i) - t(i - 1);

param = xn(i-1) + f(xn(i-1),ydok(i-1)) \* tn;

xn(i) = xn(i-1) + 1 / 2 \* tn \* ...

(f(xn(i-1),ydok(i-1)) + f(param,ydok(i)));

end

vect = xn;

HeunY.m:

function vect = HeunY(y0,t,xdok,R)

yn = zeros(length(t),1);

yn(1) = y0;

ry = R(1);

ryx = R(2);

ryy = R(3);

f = @(x,y) ry \* y + ryx \* x \* y + ryy \* y \* y;

for i=2:length(t)

tn = t(i) - t(i - 1);

param = yn(i-1) + f(xdok(i-1),yn(i-1)) \* tn;

yn(i) = yn(i-1) + 1 / 2 \* tn \* ...

(f(xdok(i-1),yn(i-1)) + f(xdok(i),param));

end

vect = yn;

## 7.3 Lisiting zadanie 3

W zadaiu wykorzystałem skrypty zad3.m i zad3Euler.m

zad3.m:

clear

% Wczytanie danych

dane = readtable("dane16.csv");

tDane = double(dane.t);

xDane = double(dane.x);

yDane = double(dane.y);

% Początkowe Parametry

rx = linspace(0,40,20);

ry = linspace(-1,0,10);

rxx = linspace(-0.1,0,20);

x0 = linspace(100,1000,50);

ry2 = linspace(-40,0,20);

ryx = linspace(0,1,10);

ryy = linspace(-0.1,0,20);

y0 = linspace(10,200,50);

combs = combinations(x0,rx,ry,rxx);

combs = table2array(combs);

combs2 = combinations(y0,ry2,ryx,ryy);

combs2 = table2array(combs2);

% Funkckcja do minimalizacji

Jx = @(xEuler) sum((xEuler - xDane).^2);

Jx2 = @(xEuler) sum((xEuler - yDane).^2);

sols = zeros(length(rx),1);

sols2 = zeros(length(ry2),1);

for i=1:length(combs)

sols(i) = Jx(zad1Euler(combs(i,1),tDane,yDane,combs(i,2:end)));

sols2(i) = Jx2(zad1Euler(combs2(i,1),tDane,xDane,combs2(i,2:end)));

end

minimum = min(sols);

Rminimum = combs(sols == minimum,:);

tempFunc = @(x) Jx(zad1Euler(x(1),tDane,yDane,x(2:end)));

opt\_PopX = fminsearch(tempFunc,Rminimum);

minimum2 = min(sols2);

Rminimum2 = combs2(sols2 == minimum2,:);

tempFunc2 = @(x) Jx2(zad1Euler(x(1),tDane,xDane,x(2:end)));

opt\_PopY = fminsearch(tempFunc2,Rminimum2);

JxAll = @(Est) sum(sum((Est - [xDane,yDane]).^2));

AllParams = [opt\_PopX,opt\_PopY];

minimizeFunc = @(x) JxAll(odeSolver(x,tDane));

options = optimset('fminsearch');

options.MaxIter = 4000;

options.MaxFunEvals = 5000;

[optAll,fVal] = fminsearch(minimizeFunc,AllParams,options)

Est = odeSolver(optAll,tDane);

figure(1)

plot(tDane,Est(:,1))

hold on

plot(tDane,xDane')

title("Wykres populacji x ode45")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

figure(2)

plot(tDane,Est(:,2))

hold on

plot(tDane,yDane)

title("Wykres populacji y ode45")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

function Est = odeSolver(AllParams,tDane)

optX = AllParams(1:4);

optY = AllParams(5:end);

startVal = [optX(1), optY(1)];

URRZ = @(t,y) [optX(2) \* y(1) + optX(3) \* y(1) \* y(2) + ...

optX(4) \* y(1) \* y(1); optY(2) \* y(2) + optY(3)...

\* y(1) \* y(2) + optY(4) \* y(2) \* y(2)];

[t,y] = ode45(URRZ,[0 3],startVal);

xEst = interp1(t, y(:,1),tDane);

yEst = interp1(t, y(:,2),tDane);

Est = [xEst,yEst];

end

zad3Euler.m:

clear

% Wczytanie danych

dane = readtable("dane16.csv");

tDane = double(dane.t);

xDane = double(dane.x);

yDane = double(dane.y);

% Początkowe Parametry

rx = linspace(0,40,20);

ry = linspace(-1,0,10);

rxx = linspace(-0.1,0,20);

x0 = linspace(100,1000,50);

ry2 = linspace(-40,0,20);

ryx = linspace(0,1,10);

ryy = linspace(-0.1,0,20);

y0 = linspace(10,200,50);

combs = combinations(x0,rx,ry,rxx);

combs = table2array(combs);

combs2 = combinations(y0,ry2,ryx,ryy);

combs2 = table2array(combs2);

% Funkckcja do minimalizacji

Jx = @(xEuler) sum((xEuler - xDane).^2);

Jx2 = @(xEuler) sum((xEuler - yDane).^2);

sols = zeros(length(rx),1);

sols2 = zeros(length(ry2),1);

for i=1:length(combs)

sols(i) = Jx(zad1Euler(combs(i,1),tDane,yDane,combs(i,2:end)));

sols2(i) = Jx2(zad1Euler(combs2(i,1),tDane,xDane,combs2(i,2:end)));

end

minimum = min(sols);

Rminimum = combs(sols == minimum,:);

tempFunc = @(x) Jx(zad1Euler(x(1),tDane,yDane,x(2:end)));

opt\_PopX = fminsearch(tempFunc,Rminimum);

minimum2 = min(sols2);

Rminimum2 = combs2(sols2 == minimum2,:);

tempFunc2 = @(x) Jx2(zad1Euler(x(1),tDane,xDane,x(2:end)));

opt\_PopY = fminsearch(tempFunc2,Rminimum2);

JxAll = @(Est) sum(sum((Est - [xDane,yDane]).^2));

AllParams = [opt\_PopX,opt\_PopY];

minimizeFunc = @(x) JxAll(EulerSolver(x,tDane));

options = optimset('fminsearch');

options.MaxIter = 4000;

options.MaxFunEvals = 5000;

[optAll,fval] = fminsearch(minimizeFunc,AllParams,options)

Est = EulerSolver(optAll,tDane);

figure(1)

plot(tDane,Est(:,1))

hold on

plot(tDane,xDane)

title("Wykres populacji x jawna metoda Eulera")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

figure(2)

plot(tDane,Est(:,2))

hold on

plot(tDane,yDane)

title("Wykres populacji y jawna metoda Eulera")

legend("populacja przybliżona","populacja dokładna")

xlabel("t - czas")

ylabel("liczność populacji")

function Est = EulerSolver(allParams,tDane)

h = 0.001;

t = min(tDane):h:max(tDane);

xO = allParams(1:4);

yO = allParams(5:end);

fx = @(x, y) xO(2) \* x + xO(3) \* x \* y + xO(4) \* x \* x;

fy = @(x, y) yO(2) \* y + yO(3) \* y \* x + yO(4) \* y \* y;

xEst = zeros(length(tDane),1);

xEst(1) = xO(1);

yEst = zeros(length(tDane),1);

yEst(1) = yO(1);

for i = 2:length(t)

tn = t(i) - t(i-1);

xEst(i) = xEst(i-1) + fx(xEst(i-1), yEst(i-1)) \* tn;

yEst(i) = yEst(i-1) + fy(xEst(i-1), yEst(i-1)) \* tn;

end % for

xEst = interp1(t, xEst, tDane);

yEst = interp1(t, yEst, tDane);

Est = [xEst,yEst];

End

## 7.4 Lisiting zadanie 4

zad4Dsolve.m:

dane = readtable("dane16.csv");

tDane = double(dane.t);

syms x y t

% wartości uzyskana w zadaniu 3

AllParams = [475.8032,12.0826,-0.1077,-0.0029,98.4847,...

-8.6214,0.0530,-0.0024];

xParams = AllParams(1:4);

yParams = AllParams(5:end);

x1 = xParams(1);

y1 = yParams(1);

rx = xParams(2);

rxy = xParams(3);

rxx = xParams(4);

ry = yParams(2);

ryx = yParams(3);

ryy = yParams(4);

eq1 = 0 == rx \* x + rxy \* x \* y + rxx \* x \* x;

eq2 = 0 == ry \* y + ryx \* x \* y + ryy \* y \* y;

sol = solve([eq1,eq2],[x,y]);

sol.x

sol.y

7.5 Lisiting zadanie 5

zad5.m

clear

% Wczytanie danych

dane = readtable("HudsonBay.csv");

tDane = double(dane.Year);

xDane = double(dane.Hares);

yDane = double(dane.Lynx);

tDane = tDane - tDane(1);

tDane = tDane / 30;

% Początkowe Parametry

rx = linspace(0,40,20);

ry = linspace(-1,0,10);

rxx = linspace(-0.1,0,20);

x0 = linspace(100,1000,50);

ry2 = linspace(-40,0,20);

ryx = linspace(0,1,10);

ryy = linspace(-0.1,0,20);

y0 = linspace(10,200,50);

combs = combinations(x0,rx,ry,rxx);

combs = table2array(combs);

combs2 = combinations(y0,ry2,ryx,ryy);

combs2 = table2array(combs2);

% Funkckcja do minimalizacji

Jx = @(xEuler) sum((xEuler - xDane).^2);

Jx2 = @(xEuler) sum((xEuler - yDane).^2);

sols = zeros(length(rx),1);

sols2 = zeros(length(ry2),1);

for i=1:length(combs)

sols(i) = Jx(zad1Euler(combs(i,1),tDane,yDane,combs(i,2:end)));

sols2(i) = Jx2(zad1Euler(combs2(i,1),tDane,xDane,combs2(i,2:end)));

end

minimum = min(sols);

Rminimum = combs(sols == minimum,:);

tempFunc = @(x) Jx(zad1Euler(x(1),tDane,yDane,x(2:end)));

opt\_PopX = fminsearch(tempFunc,Rminimum);

minimum2 = min(sols2);

Rminimum2 = combs2(sols2 == minimum2,:);

tempFunc2 = @(x) Jx2(zad1Euler(x(1),tDane,xDane,x(2:end)));

opt\_PopY = fminsearch(tempFunc2,Rminimum2);

JxAll = @(Est) sum(sum((Est - [xDane,yDane]).^2));

AllParams = [opt\_PopX,opt\_PopY];

minimizeFunc = @(x) JxAll(odeSolver(x,tDane));

options = optimset('fminsearch');

options.MaxIter = 4000;

options.MaxFunEvals = 5000;

[optAll,fval] = fminsearch(minimizeFunc,AllParams,options)

Est = odeSolver(optAll,tDane);

figure(1)

plot(tDane,Est(:,1))

hold on

plot(tDane,xDane')

title("Wykres populacji x")

legend("populacja przybliżona","populacja dokładna")

figure(2)

plot(tDane,Est(:,2))

hold on

plot(tDane,yDane)

title("Wykres populacji y")

legend("populacja przybliżona","populacja dokładna")

function Est = odeSolver(AllParams,tDane)

optX = AllParams(1:4);

optY = AllParams(5:end);

startVal = [optX(1), optY(1)];

URRZ = @(t,y) [optX(2) \* y(1) + optX(3) \* y(1) \* y(2) + ...

optX(4) \* y(1) \* y(1); optY(2) \* y(2) + optY(3)...

\* y(1) \* y(2) + optY(4) \* y(2) \* y(2)];

[t,y] = ode45(URRZ,[0 3],startVal);

xEst = interp1(t, y(:,1),tDane);

yEst = interp1(t, y(:,2),tDane);

Est = [xEst,yEst];

end