Politechnika Warszawska,

Wydział Elektroniki i Technik Informacyjnych

Metody numeryczne

Zadanie projektowe #3: Rozwiązywanie równań różniczkowych

Oświadczam, że niniejsza, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmioty Metody numeryczne została wykonana przeze mnie samodzielnie.

Anna Czarkowska

Spis treści:

I. Lista wykorzystanych oznaczeń

$A, D, R, \ldots$- macierze zmiennych skalarnych
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ - wektory zmiennych skalarnych
$h, a, b, \ldots$ - zmienne skalarne
$x_{n,v}, u_{n,v}, \ldots$ - element macierzy ($X, U, \ldots$) w n-tym wierszu i v-tej kolumnie
$u_{n,:}, u_{:,v}, \ldots$ – wszystkie elementy z wiersza n lub wszystkie elementy z kolumny v
$x_n, y_n, \ldots$ - n-ty element wektora
$x(0), y(0)$ – wartości początkowe dla rozwiązywanych równań różniczkowych
$f(t, y)$ – przykładowy zapis funkcji f zależnej od zmiennych t i y

LV – układ równań Lotki-Volterry

II. Wprowadzenie

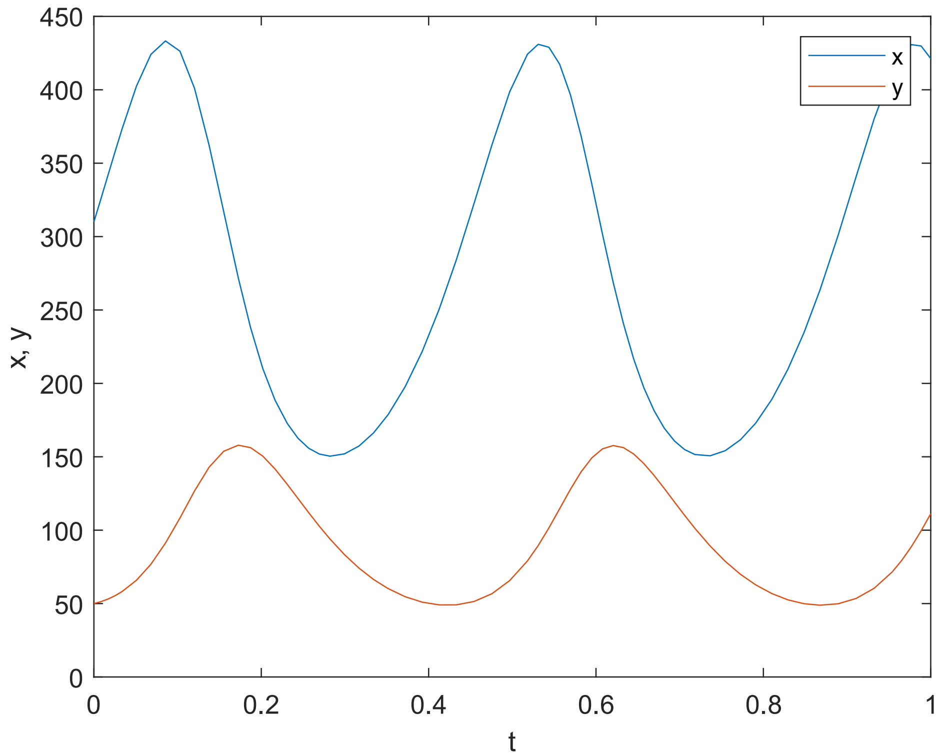
Tematyką tego projektu jest analiza wpływu dobranej metody rozwiązywania równań różniczkowych na końcowy wynik na podstawie układu równań Lotki-Volterry, który stanowi model opisu populacji drapieżników i ofiar. Sprawdzane są też zależności zagregowanego błędu względnego wyników od długości kroku całkowania h oraz parametry układu LV, dla których symulacja jest najbardziej optymalna.

III. Metodyka i wyniki doświadczeń

Zad. 1 - Rozwiązanie układu równań Lotki-Volterry przy użyciu funkcji *ode45*
Aby wyznaczyć rozwiązanie podanego w zadaniu układu równań LV, należy zdefiniować wektor dwuelementowy, w którym zawarte są prawe strony równań w postaci funkcji zależnych od czasu t oraz populacji x i y.
Następnie wykorzystana zostaje funkcja *ode45*[1], która przyjmuje następujące parametry:

- Wyznaczone w wektorze funkcje odpowiadające prawej stronie równań LV
- Wartości definiujące zakres dla uzyskiwanych wartości t
- Wartości początkowe x(0) i y(0)

Uzyskane wyniki to wartości t, x i y, które zostają zapisane w wektorze $\boldsymbol{t}$ i macierzy U, a następnie przedstawione na wykresie z Rys. 1.



Rys. 1 – Uzyskane rozwiązania układu LV (x(t) i y(t))

Zad. 2 – Wykorzystanie różnych metod do rozwiązania układu równań LV

Zamiast wykorzystywać funkcję MATLAB-a do wyznaczenia rozwiązania, w tym zadaniu sprawdzane będą inne metody, którymi można uzyskać wyniki o różnej dokładności. Dla każdego rozwiązania wartości na osi x wykresów są określone jako t z zakresu od 0 do 1, rozłożone równomiernie co krok h. Ze wzorów obliczane są obie wartości populacji x i y.

Pierwszą sprawdzaną metodą jest **otwarta metoda Eulera**[2] zdefiniowana wzorem:

$$y_n = y_{n-1} + h \cdot f(t_{n-1}, y_{n-1}) \tag{1}$$

Rys. 2.1 przedstawia rozwiązanie układu wyznaczonego powyższym wzorem dla kolejnych elementów wektorów x i y.

Druga wykorzystywana metoda to **zamknięta metoda Eulera**[3], którą przedstawia wzór:

$$y_n = y_{n-1} + h \cdot f(t_n, y_n) \tag{2}$$

Do wyznaczenia kolejnych rozwiązań w kodzie użyta zostaje funkcja *fsolve*[4]. Wyniki otrzymane tą metoda są przedstawione na Rys. 2.

Otwarta metoda punktu środkowego[5], na podstawie której uzyskiwany jest kolejny wykres z Rys. 2.3, opisana jest wzorem:

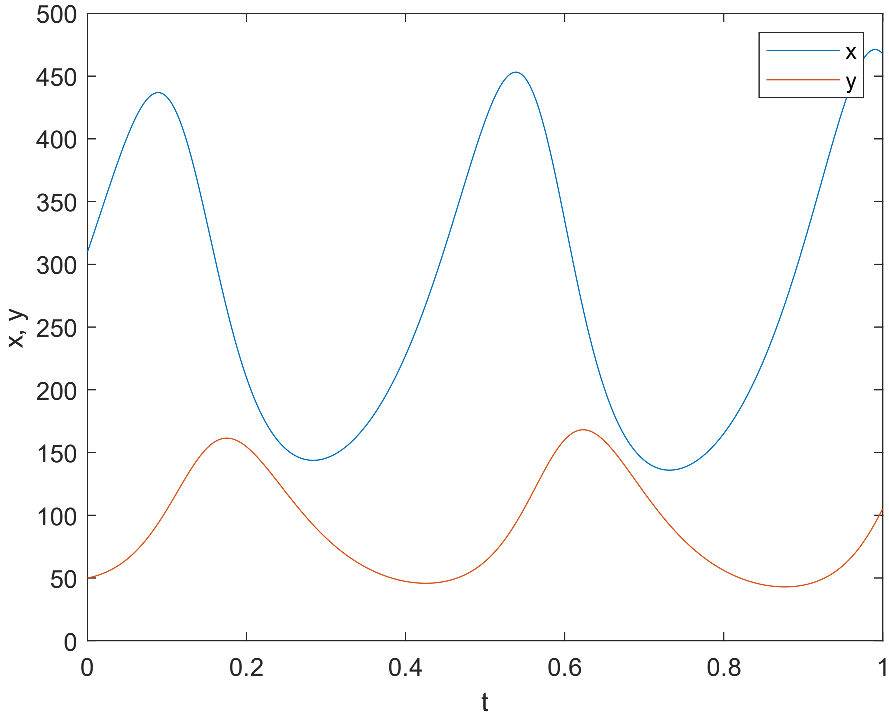
$$y_n = y_{n-1} + h \cdot f\left(t_{n-\frac{1}{2}}, y_{n-\frac{1}{2}}\right), \tag{3}$$

gdzie $y_{n-\frac{1}{2}} = y_{n-1} + \frac{1}{2} \cdot h \cdot f(t_{n-1}, y_{n-1})$ i $t_{n-\frac{1}{2}} = t_{n-1} + \frac{1}{2} \cdot h$.

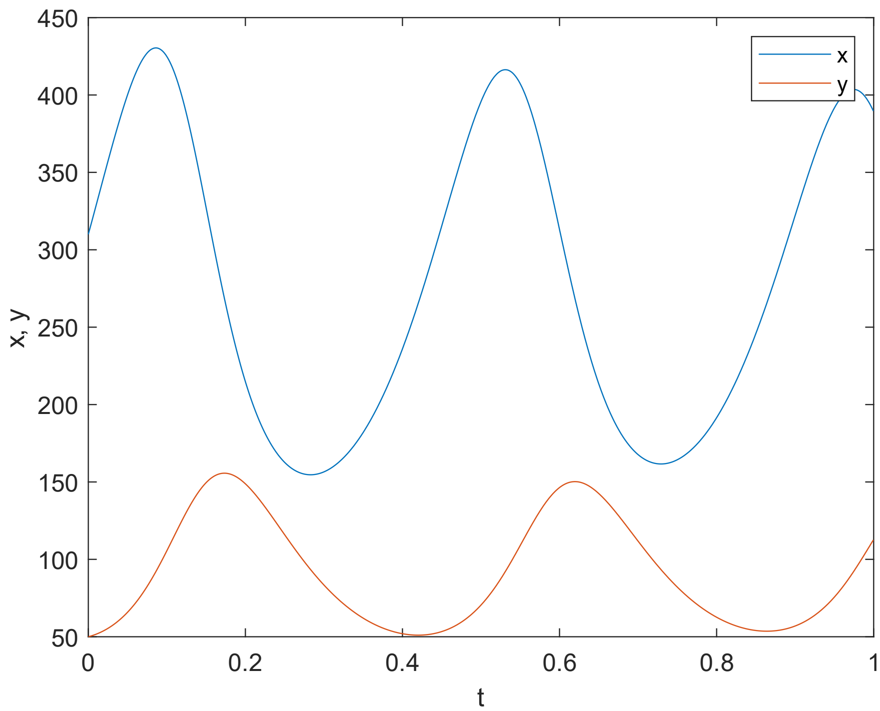
Ostatnią użytą metodą jest **metoda Adamsa-Moultona 3. rzędu**[6], a jej wzór prezentuje się następująco:

$$y_n = y_{n-1} + \frac{5}{12} \cdot h \cdot f(t_n, y_n) + \frac{8}{12} \cdot h \cdot f(t_{n-1}, y_{n-1}) - \frac{1}{12} \cdot h \cdot f(t_{n-2}, y_{n-2}) \quad (4)$$

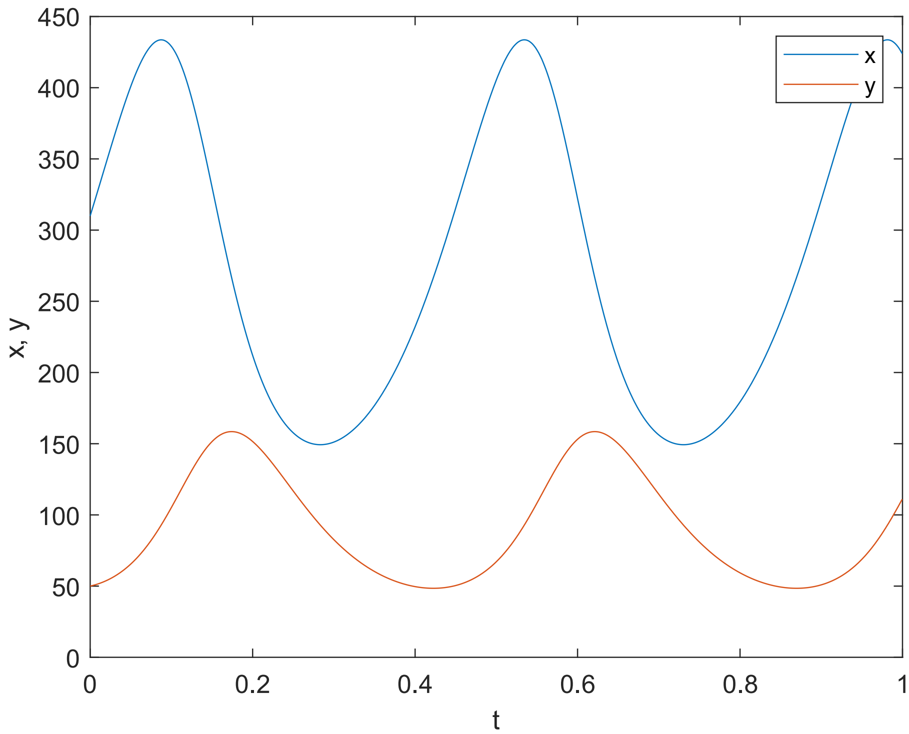
Wynik wyznaczony tą metodą przedstawiony jest na Rys. 2.4.



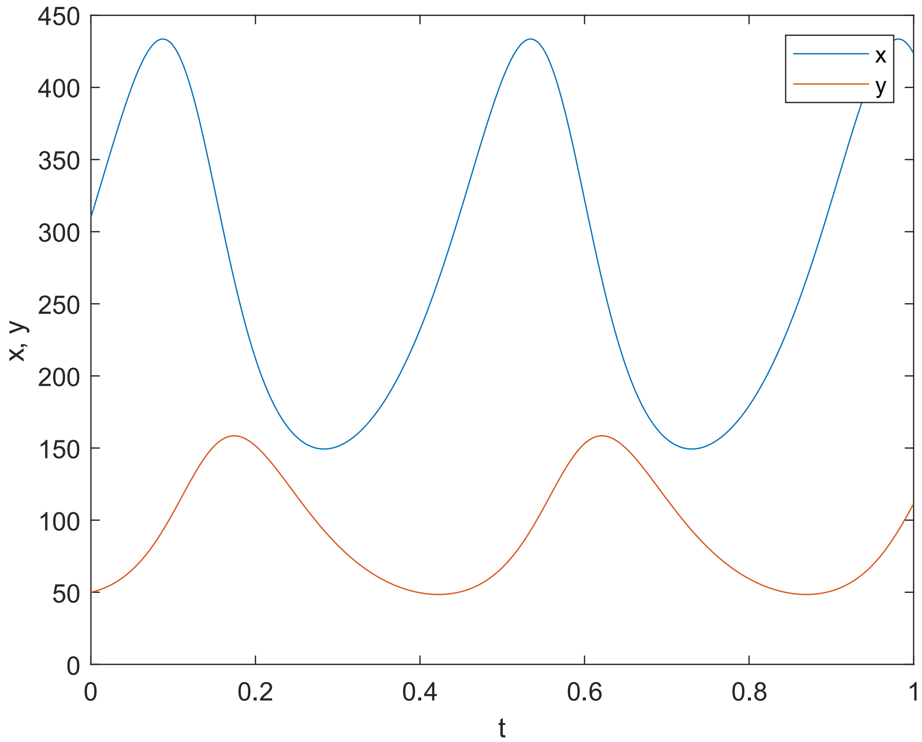
Rys. 2.1 – Rozwiązanie układu LV otwartą metodą Eulera



Rys. 2.2 – Rozwiązanie układu LV zamkniętą metodą Eulera



Rys. 2.3 – Rozwiązanie układu LV otwartą metodą punktu środkowego



Rys. 2.4 – Rozwiązanie układu LV metodą Adamsa-Moultona trzeciego rzędu

Zad. 3 – Wyznaczenie zagregowanego błędu względnego rozwiązań poszczególnymi metodami

Pierwszym krokiem tego zadania jest określenie wartości referencyjnych pierwszej części rozwiązań układu LV, aby móc wykorzystać je do wyliczenia błędu. Obliczane są ponownie przy pomocy funkcji *ode45*, jednak tym razem podawane są dwa dodatkowe parametry określające względną i bezwzględną tolerancję, a za zakres służy nam wektor t, w którym znajdują się wartości od 0 do 1 co krok h.

Otrzymane zagregowane błędy bezwzględne dla każdej z metod z zadania drugiego podane są w Tab.1.

Zagregowany błąd względny δ_x	
Otwarta metoda Eulera	0.03910
Zamknięta metoda Eulera	0.03485
Otwarta metoda punktu środkowego	0.00024
Metoda Adamsa-Moultona 3. rzędu	0.00009

Tab. 1 – Zagregowany błąd względny wyników wyznaczonych przez różne metody z zadania drugiego (zaokrąglone do 5 liczb po przecinku)

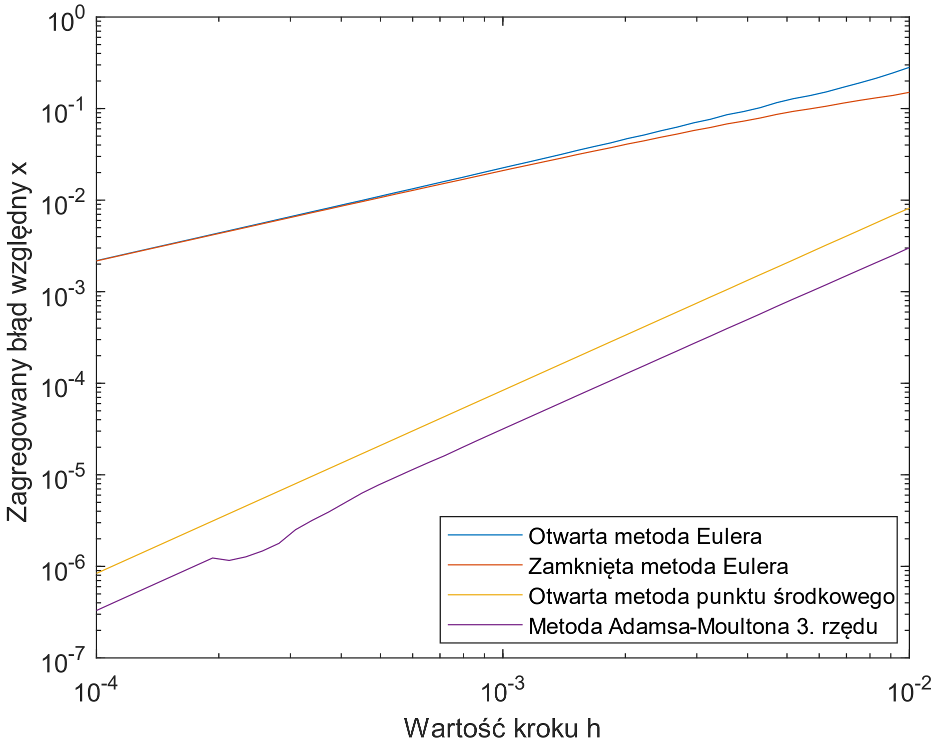
Zad. 4 – Wyznaczenie zależności zagregowanego błędu względnego od kroku h

Czynności wykonywane w zadaniu drugim i trzecim są powtarzane w pętli dla każdego elementu wektora zawierającego długości skoku całkowania h

z zakresu od 10^{-4} do 10^{-2}. Wektor utworzony zostaje poprzez użycie funkcji *logspace*[7].

Wartości referencyjne są ponownie wyznaczane dla każdej nowej długości kroku h, ponieważ liczba wartości t zmniejsza się wraz ze wzrostem tej długości, a ich zakres nie ulega zmianie.

Otrzymane charakterystyki przedstawione są na Rys. 4.



Rys. 3. – Zależność zagregowanego błędu bezwzględnego wyników x od wartości kroku całkowania dla czterech różnych metod rozwiązywania równań różniczkowych

Zad. 5 – Znalezienie parametrów układu LV dla minimalnej wartości kryterium

Pierwszym krokiem do wykonania jest wczytanie podanej tabeli danych oraz zapisanie wartości początkowych parametrów podanych w zadaniu pierwszym. Następnie utworzona zostaje funkcja przyjmująca dane z tabeli i potrzebne parametry oraz zwracająca wartość określonego w zadaniu kryterium. W funkcji znajduje się również wykorzystanie *ode45*, dzięki której obliczane są wartości referencyjne.

Końcowe parametry wyszukiwane są z użyciem funkcji *fminsearch*[8], do której podawana jest wyznaczona funkcja, w której zmienną staje się p, oraz wartości początkowe parametrów zapisanych na początku.

Otrzymane wyniki znajdują się w Tab. 5.

	p_1	p_2	p_3	p_4
Wyznaczona wartość p:	-16.21971	0.32167	0.11215	17.23729

Tab. 2 – Wartości uzyskanych parametrów układu LV minimalizujących wartość kryterium z zadania piątego (zaokrąglone do piątego miejsca po przecinku)

Dyskusja wyników eksperymentów numerycznych

Zadanie 4

Zależność błędu δ_x od kroku całkowania h jest liniowa i wprost proporcjonalna dla wszystkich wykorzystywanych metod. W przypadku obu metod Eulera, czyli sprawdzanych metod jednokrokowych, błąd jest większy od błędów pozostałych metod, które są z kolei wielokrokowe. Te różnice akurat nie są zależne od kroku całkowania.

Wpływ na charakterystyki ma również to, czy metoda Eulera jest zamknięta czy otwarta – dla coraz większych wartości kroku ich błąd jest stosunkowo mniejszy od błędu wyników z metod otwartych. Dla otwartej metody punktu środkowego i zamkniętej metody Adamsa niewidoczna jest podobna zależność.

IV. Wnioski

Wybór metody rozwiązywania równań różniczkowych wpływa na dokładność i złożoność obliczeń. Metody otwarte są łatwiejsze i szybsze do obliczenia niż metody zamknięte, jednak ich uzyskiwane wyniki zawierają większy błąd względny dla dużych długości kroku całkowania. Kolejnym istotnym aspektem jest liczba kroków danej metody – im jest większa, tym lepsze można otrzymać przybliżenie wyników dokładnych.

Największy wpływ na dokładność rozwiązania ma krok całkowania h. Im jest mniejszy, tym więcej punktów jest uwzględnionych w przybliżeniu, co daje dokładniejsze przybliżenie przebiegu funkcji. Problemem jest jednak duże wydłużenie czasu obliczeń przy bardzo małych h, co można zaobserwować przy uruchamianiu kodu do zadania 4. Tak mały krok nie musi być konieczny przy niektórych obliczeniach, dlatego najlepiej jest dostosować go do potrzebnej dokładności.

V. Bibliografia

[1] https://www.mathworks.com/help/matlab/ref/ode45.html#d126e1117397

[2] P. Mazurek, slajd nr 193 z wykładu Metod Numerycznych (MNUB), Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, semestr letni 2023/2024

[3] P. Mazurek, slajd nr 196 z wykładu Metod Numerycznych (MNUB), Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, semestr letni 2023/2024

[4] https://www.mathworks.com/help/optim/ug/fsolve.html

[5] P. Mazurek, slajd nr 193 z wykładu Metod Numerycznych (MNUB), Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, semestr letni 2023/2024

[6] P. Mazurek, slajd nr 209 z wykładu Metod Numerycznych (MNUB), Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, semestr letni 2023/2024
[7] https://www.mathworks.com/help/matlab/ref/logspace.html
[8] https://www.mathworks.com/help/matlab/ref/fminsearch.html

VI. Listing programów

1. Plik *zad1.m*

```
clear
close all

P = [13 0.14 0.06 16];
Lotka_Volterra = @(t,u) [u(1).*(P(1)-P(2)*u(2)); u(2).*(P(3)*u(1)-P(4))];

tspan = [0 1];
u0 = [310; 50];
[t,u] = ode45(Lotka_Volterra, tspan, u0);

figure;
plot(t, u);
xlabel("t")
ylabel("x, y")
legend("x","y")
```

2. Plik *zad2.m*

```
clear
close all

h = 0.0017;
P = [13 0.14 0.06 16];
fx = @(x,y) x*(P(1)-P(2)*y);
fy = @(x,y) y*(P(3)*x-P(4));

t = 0:h:1;
N = length(t);
xa = zeros(N, 1);
ya = zeros(N, 1);
xa(1) = 310;
ya(1) = 50;
xb = xa;
yb = ya;
xc = xa;
yc = ya;
xd = xa;
yd = ya;

%otwarta metoda Eulera
for n = 2:N
    xa(n) = xa(n-1) + h*fx(xa(n-1),ya(n-1));
    ya(n) = ya(n-1) + h*fy(xa(n-1),ya(n-1));
end
```

```
figure("Name","Otwarta metoda Eulera");
plot (t, xa, t, ya)
legend("x", "y")
xlabel("t")
ylabel("x, y")

%zamknięta metoda Eulera
for n = 1:N-1
    F = @(u) [xb(n) - u(1) + h*fx(u(1),u(2));
        yb(n) - u(2) + h*fy(u(1),u(2))];

    u_temp = fsolve(F, [xb(n); yb(n)]);

    xb(n+1) = u_temp(1);
    yb(n+1) = u_temp(2);
end

figure("Name","Zamknięta metoda Eulera");
plot(t, xb, t, yb);
legend("x", "y")
xlabel("t")
ylabel("x, y")

%otwarta metoda punktu środkowego
for n = 2:N
    x_temp = xc(n-1) + 1/2*h*fx(xc(n-1),yc(n-1));
    y_temp = yc(n-1) + 1/2*h*fy(xc(n-1),yc(n-1));
    xc(n) = xc(n-1) + h*fx(x_temp,y_temp);
    yc(n) = yc(n-1) + h*fy(x_temp,y_temp);
end

figure("Name","Otwarta metoda punktu środkowego");
plot(t, xc, t, yc);
legend("x", "y")
xlabel("t")
ylabel("x, y")

%metoda Adamsa-Moultona 3. rzędu
F = @(u) [xd(1) - u(1) + h*fx(u(1),u(2));
        yd(1) - u(2) + h*fy(u(1),u(2))];
u_temp = fsolve(F, [xd(1); yd(1)]);
xd(2) = u_temp(1);
yd(2) = u_temp(2);

for n = 2:N-1
    F = @(u) [xd(n) - u(1) + 5/12*h*fx(u(1),u(2)) + 2/3*h*fx(xd(n),yd(n)) -
1/12*h*fx(xd(n-1),yd(n-1));
        yd(n) - u(2) + 5/12*h*fy(u(1),u(2)) + 2/3*h*fy(xd(n),yd(n)) -
1/12*h*fy(xd(n-1),yd(n-1))];

    u_temp = fsolve(F, [xd(n); yd(n)]);

    xd(n+1) = u_temp(1);
    yd(n+1) = u_temp(2);
end

figure("Name", "Metoda Adamsa-Moultona 3. rzędu");
plot(t, xd, t, yd);
legend("x", "y")
```

```
xlabel("t")
ylabel("x, y")
```

3. zad3.m

```
clear
run ("zad2.m")
close all

P = [13 0.14 0.06 16];
Lotka_Volterra = @(t,u) [u(1).*(P(1)-P(2)*u(2)); u(2).*(P(3)*u(1)-P(4))];

u0 = [310; 50];
options = odeset('RelTol',1e-8, 'AbsTol', 1e-12);
[tref,u] = ode45(Lotka_Volterra, t, u0, options);
xref = u(:,1);
yref = u(:,2);

error_xa = sqrt(sum((xa - xref).^2))/sqrt(sum(xref.^2));
error_xb = sqrt(sum((xb - xref).^2))/sqrt(sum(xref.^2));
error_xc = sqrt(sum((xc - xref).^2))/sqrt(sum(xref.^2));
error_xd = sqrt(sum((xd - xref).^2))/sqrt(sum(xref.^2));
```

4. zad4.m

```
clear
close all

H = logspace(-4, -2);
P = [13 0.14 0.06 16];

Lotka_Volterra = @(t,u) [u(1).*(P(1)-P(2)*u(2)); u(2).*(P(3)*u(1)-P(4))];
fx = @(x,y) x*(P(1)-P(2)*y);
fy = @(x,y) y*(P(3)*x-P(4));
options = odeset('RelTol',1e-8, 'AbsTol', 1e-12);
u0 = [310; 50];

error_xa = zeros(length(H), 1);
error_xb = zeros(length(H), 1);
error_xc = zeros(length(H), 1);
error_xd = zeros(length(H), 1);

idx = 1;
for h = H
    t = 0:h:1;
    N = length(t);
    xa = zeros(N, 1);
    ya = zeros(N, 1);
    xa(1) = u0(1);
    ya(1) = u0(2);
    xb = xa;
    yb = ya;
    xc = xa;
    yc = ya;
    xd = xa;
    yd = ya;
```

```
    [tref,u] = ode45(Lotka_Volterra, t, u0, options);
    xref = u(:,1);
    yref = u(:,2);

    %otwarta metoda Eulera
    for n = 2:N
        xa(n) = xa(n-1) + h*fx(xa(n-1),ya(n-1));
        ya(n) = ya(n-1) + h*fy(xa(n-1),ya(n-1));
    end

    %zamknięta metoda Eulera
    for n = 1:N-1
        F = @(uf) [xb(n) - uf(1) + h*fx(uf(1),uf(2));
            yb(n) - uf(2) + h*fy(uf(1),uf(2))];

        u_temp = fsolve(F, [xb(n); yb(n)]);

        xb(n+1) = u_temp(1);
        yb(n+1) = u_temp(2);
    end

    %otwarta metoda punktu środkowego
    for n = 2:N
        x_temp = xc(n-1) + 1/2*h*fx(xc(n-1),yc(n-1));
        y_temp = yc(n-1) + 1/2*h*fy(xc(n-1),yc(n-1));
        xc(n) = xc(n-1) + h*fx(x_temp,y_temp);
        yc(n) = yc(n-1) + h*fy(x_temp,y_temp);
    end

    %metoda Adamsa-Moultona 3. rzędu
    F = @(uf) [xd(1) - uf(1) + h*fx(uf(1),uf(2));
            yd(1) - uf(2) + h*fy(uf(1),uf(2))];
    u_temp = fsolve(F, [xd(1); yd(1)]);
    xd(2) = u_temp(1);
    yd(2) = u_temp(2);

    for n = 2:N-1
        F = @(uf) [xd(n) - uf(1) + 5/12*h*fx(uf(1),uf(2)) + 2/3*h*fx(xd(n),yd(n))
- 1/12*h*fx(xd(n-1),yd(n-1));
            yd(n) - uf(2) + 5/12*h*fy(uf(1),uf(2)) + 2/3*h*fy(xd(n),yd(n)) -
1/12*h*fy(xd(n-1),yd(n-1))];

        u_temp = fsolve(F, [xd(n); yd(n)]);

        xd(n+1) = u_temp(1);
        yd(n+1) = u_temp(2);
    end

    %zagregowany błąd względny x dla każdej metody
    error_xa(idx) = sqrt(sum((xa - xref).^2))/sqrt(sum(xref.^2));
    error_xb(idx) = sqrt(sum((xb - xref).^2))/sqrt(sum(xref.^2));
    error_xc(idx) = sqrt(sum((xc - xref).^2))/sqrt(sum(xref.^2));
    error_xd(idx) = sqrt(sum((xd - xref).^2))/sqrt(sum(xref.^2));
    idx = idx + 1;
end

loglog(H, error_xa, H, error_xb, H, error_xc, H, error_xd)
```

```
legend("Otwarta metoda Eulera", "Zamknięta metoda Eulera", "Otwarta metoda punktu środkowego", "Metoda Adamsa-Moultona 3. rzędu")
xlabel("Wartość kroku h")
ylabel("Zagregowany błąd względny x")
```

5. zad4.m

```
clear
close all
T = readtable ("MNUB_24L_P3_dane08.csv");

x = T{:,2};
y = T{:,3};
P0 = [13 0.14 0.06 16];
u0 = [x(1), y(1)];

P_est = fminsearch(@(P) Jfun(P, T{:,1}, x, y, u0), P0);

function J = Jfun(P, t, x, y, u0)
    Lotka_Volterra = @(t,u,P) [u(1).*(P(1)-P(2)*u(2)); u(2).*(P(3)*u(1)-P(4))];

    [~, uest] = ode45(@(t, u) Lotka_Volterra(t, u, P), t, u0);
    xest = uest(1);
    yest = uest(2);

    J = sum((xest - x).^2) + sum((yest - y).^2);
end
```