Metody Numeryczne

Projekt 4 – zadanie 4.3

Autor: Kamil Jabłonowski

Nr albumu: 271523

Prowadzący: dr hab. Inż. Andrzej Karbowski

Cel zadania

Celem zadania było obliczenie przebiegu trajektorii ruchu punktu na przedziale $[0, 20]$. Ruch opisany jest układem równań różniczkowych zwyczajnych pierwszego rzędu:

$$x_1'(t) = x_2(t) + x_1(t)(0{,}2 - x_1^2(t) - x_2^2(t))$$

$$x_2'(t) = -x_1(t) + x_2(t)(0{,}2 - x_1^2(t) - x_2^2(t))$$

W których nie występuje jawnie zmienna niezależna t , wobec czego w dalszych rozważaniach pominięto ją w zapisie i przyjęto $x_i = x_i(t)$.

Rozwiązania należało znaleźć dla następujących warunków początkowych:

a) $x_1(0) = 8, \quad x_2(0) = 7$
b) $x_1(0) = 0, \quad x_2(0) = 0{,}2$
c) $x_1(0) = 6, \quad x_2(0) = 0$
d) $x_1(0) = 0{,}01, \quad x_2(0) = 0{,}001$

Do znalezienia rozwiązań wykorzystano metody:

(1) Rungego-Kutty czwartego rzędu ze stałym krokiem
(2) Wielokrokową predyktor-korektor Adamsa czwartego rzędu ze stałym krokiem
(3) Rungego-Kutty czwartego rzędu ze zmiennym krokiem

Algorytmy zaimplementowane zostały w taki sposób, żeby dopuścić również możliwość ich wykorzystania w przypadku układów równań o dowolnej liczbie równań, dopuszczono również możliwość rozwiązywania równań, w których funkcjach prawych stron równań występuje jawnie zmienna niezależna.

Opis zagadnienia

Zagadnieniem, którego dotyczy opisane zadanie, jest numeryczne rozwiązanie układu równań różniczkowych zwyczajnych pierwszego rzędu z warunkami początkowymi. Układ taki jest postaci

$$\frac{d\boldsymbol{y}(x)}{dx} = \boldsymbol{f}(x, \boldsymbol{y}(x))$$

Gdzie $\boldsymbol{y}(x) = [y_1(x), y_2(x), \dots, y_m(x)]^T$ – wektor zmiennych zależnych (rozwiązań),

$\boldsymbol{f} = [f_1, f_2, \dots, f_m]^T$ – wektor funkcji prawych stron równań

x – zmienna niezależna

Oraz dane są warunki początkowe postaci $y_i(a) = y_{ia}$, $i = 1, \dots m$.

Numeryczne metody rozwiązywania układu równań różniczkowych są metodami różnicowymi, w których wartość przybliżona rozwiązania obliczana jest w kolejnych, dyskretnych punktach x_n: $a = x_0 \leq x_1 \leq x_2 \leq \cdots \leq x_n \leq \cdots \leq b$. Krokiem metody nazywamy $h_n = x_{n+1} - x_n$. krok może być stały lub zmienny. Metody dzielimy na jednokrokowe (takie w których do wyznaczenia rozwiązania y_n potrzebujemy jedynie informacji o rozwiązaniu z poprzedniego kroku y_{n-1}) oraz metody wielokrokowe (takie w których do wyznaczenia rozwiązania y_n potrzebujemy informacji o rozwiązaniach z kilku poprzednich kroków - $y_{n-1}, y_{n-2}, \dots, y_{n-j}$).

Metoda Rungego-Kutty czwartego rzędu ze stałym krokiem

Metody Rungego-Kutty to ogólny zbiór metod jednokrokowych, w których

$$y_{n+1} = y_n + h \cdot \sum_{i=1}^{m} w_i k_i$$

$$k_1 = f(x_n, y_n)$$

$$k_i = f\left(x_n + c_i h, y_n + h \cdot \sum_{j=1}^{i-1} a_{ij} k_j\right), \quad i = 2, 3, \dots m$$

Gdzie do wykonania jednego kroku tej metody, należy wartości prawych stron równań różniczkowych obliczyć m razy. Parametry w_i, a_{ij}, c_i dobiera się w taki sposób, aby rząd metody był możliwie wysoki.

W naszym zadaniu korzystaliśmy z metody RK4 (Rungego-Kutty 4 rzędu). W tym przypadku wzory dla tej metody wyglądają następująco

$$k_1 = f(x_n, y_n)$$

$$k_2 = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right)$$

$$k_3 = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2\right)$$

$$k_4 = f\left(x_n + h, y_n + \frac{1}{2}hk_3\right)$$

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$

Interpretacja tej metody jest następująca: wyznaczamy wartość k_1 pochodnej w punkcie (x_n, y_n), następnie wyznaczamy k_2 i k_3 wartości pochodnych w punktach na środku przedziału $[x_n, x_{n+1}]$ oraz k_4 – wartość pochodnej na końcu przedziału. Aproksymacja pochodnej w punkcie (x_{n+1}, y_{n+1}) wyznaczana jest jako średnia ważona z wagami 1 na krańcach i 2 w punkcie środkowym.

Najtrudniejszym elementem tej metody jest konieczność wyboru kroku w taki sposób, aby był on wystarczająco niewielki do uzyskania odpowiedniej dokładności, ale jednocześnie nie powinien być dużo mniejszy, ponieważ mniejszy krok zwiększa ilość wykonywanych iteracji, a więc też czas pracy algorytmu.

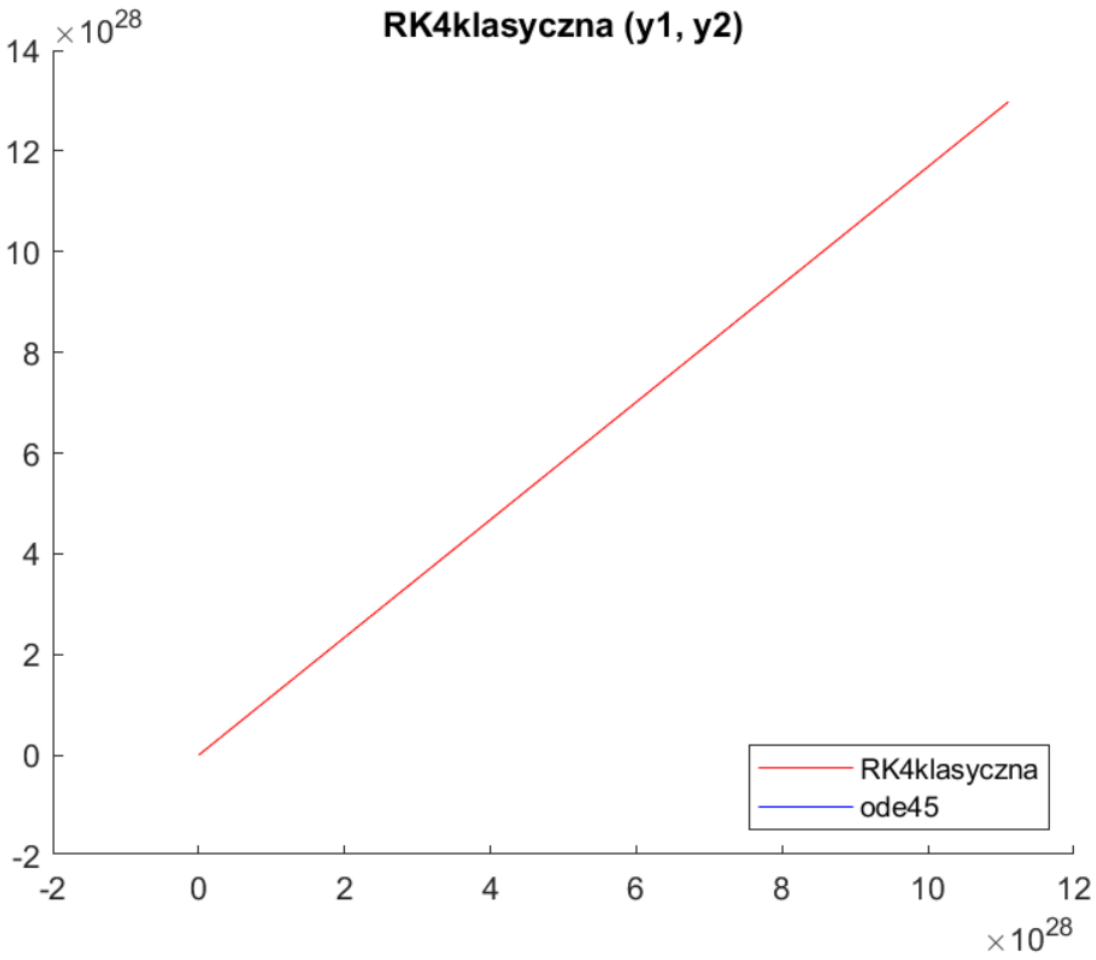
Implementacja w języku MATLAB

```
function [x, y] = RK4klasyczna(f, y0, a, h, ~)
    n = length(y0);              % wymiarowość układu
    x = (a(1):h:a(2))';          % chwile czasu w których wyznaczane będą wartości
    k = zeros(4, n);             % macierz na wartości k używane w metodzie
    y = zeros(length(x), n);     % macierz na rozwiązania y(i,j) - wartość j-tej funkcji dla xi
    y(1, :) = y0;
    for i = 1:(length(x) - 1)
        k(1, :) = f(x(i), y(i, :));
        k(2, :) = f(x(i) + 0.5 * h, y(i, :) + 0.5 * h * k(1, :));
        k(3, :) = f(x(i) + 0.5 * h, y(i, :) + 0.5 * h * k(2, :));
        k(4, :) = f(x(i) + h, y(i, :) + h * k(3, :));

        y(i + 1, :) = y(i, :) + (1 / 6) * h * (k(1, :) + 2 * k(2, :) + 2 * k(3, :) + k(4, :));
    end
end
```

Wyniki

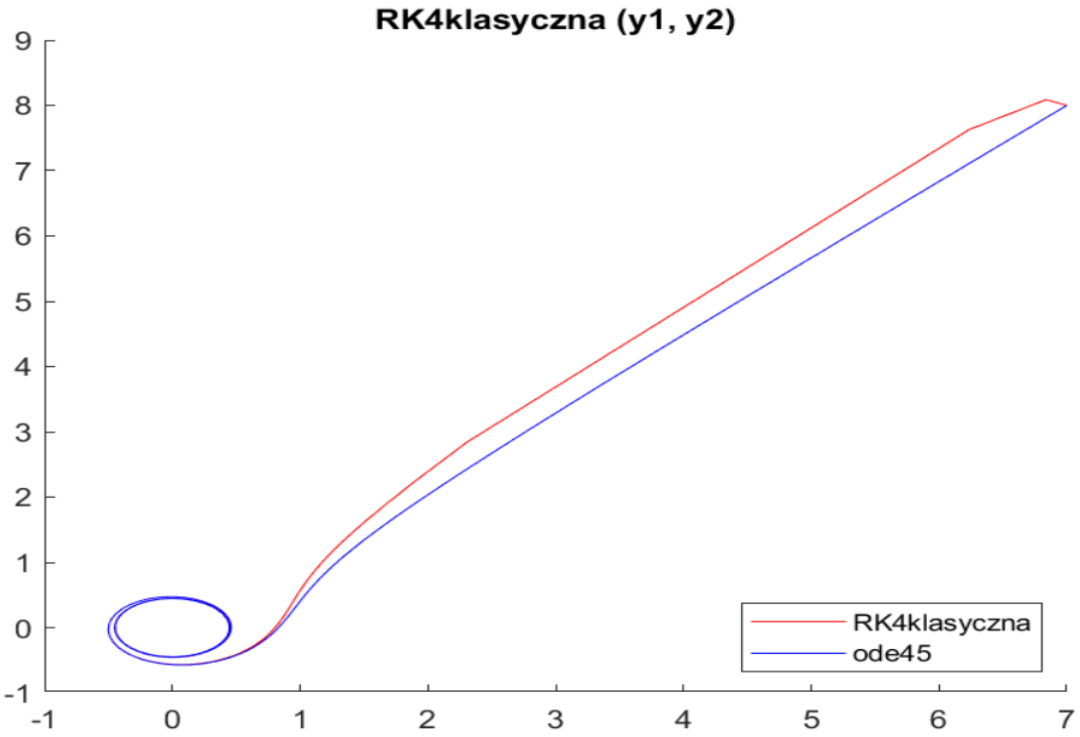
a) Punkty startowe $x_1(0) = 7, x_2(0) = 8$

$$h = 0{,}1$$



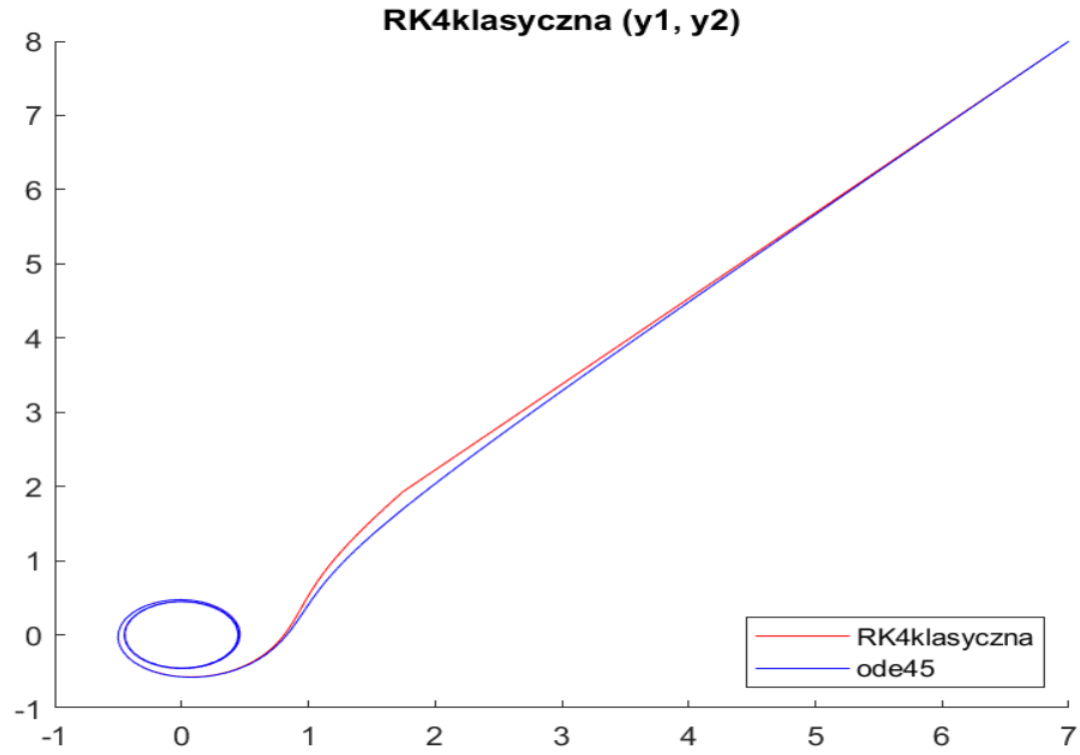
Metoda w tym przypadku okazała się rozbieżna.

$$h = 0{,}02116$$



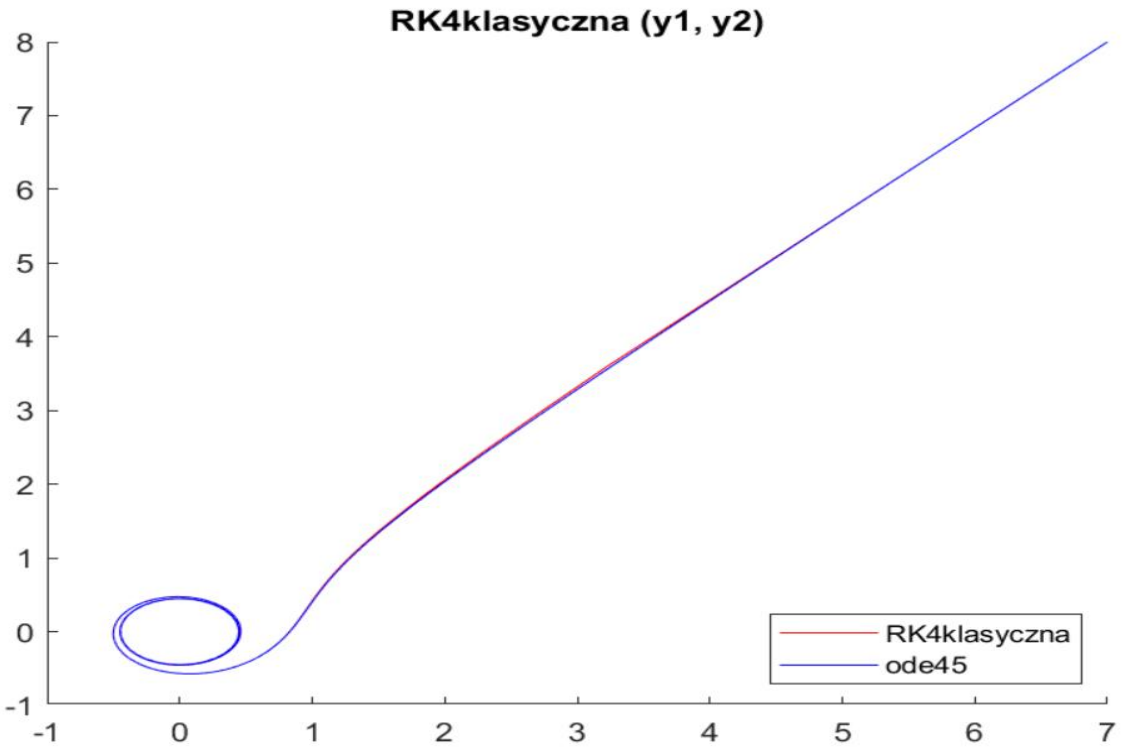
Jest to pierwsza znaleziona wartość kroku dla której metoda jest zbieżna. Widać jednak, że szczególnie na przedziale [1,7] błąd względem metody ode45 jest bardzo duży

$$h = 0{,}015$$



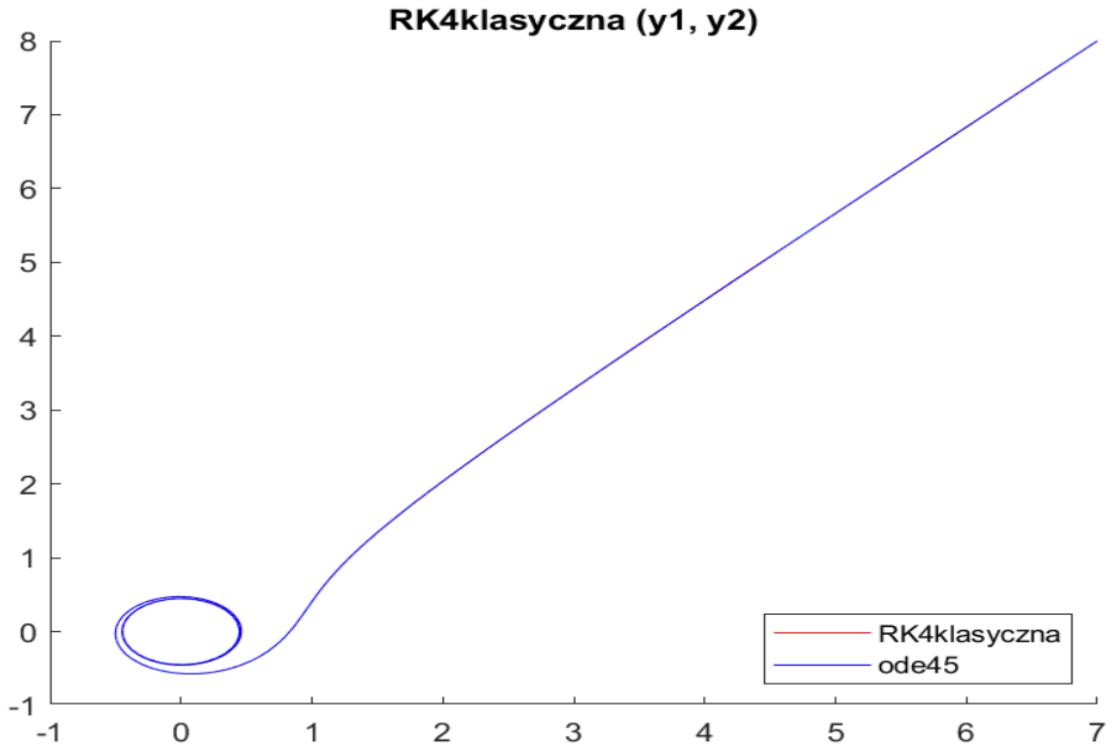
Dla kroku równego 0,015 widać poprawę, jednak błąd nadal jest duży na odcinku $[1, 3]$.

$h = 0{,}01$



Widzimy już stosunkowo zadowalające rozwiązanie

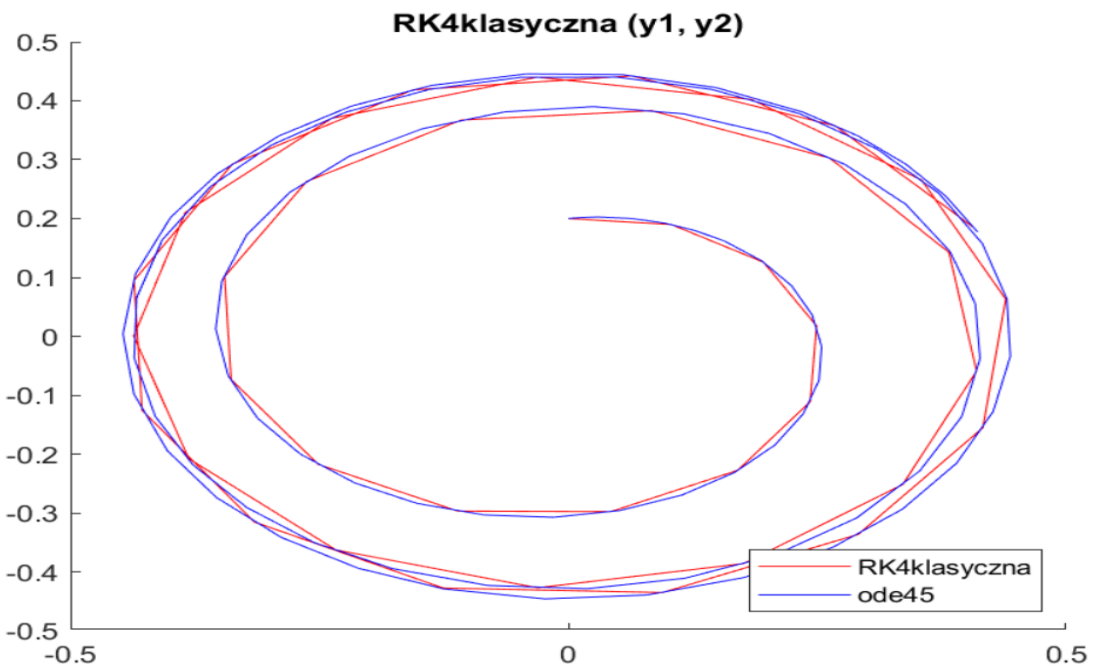
$h = 0{,}007$



Przy takim kroku widać że rozwiązanie tą metodą jest bardzo zbliżone do rozwiązania uzyskanego metodą ode45, wobec tego uznano że $h = 0{,}007$ jest optymalną wartością kroku

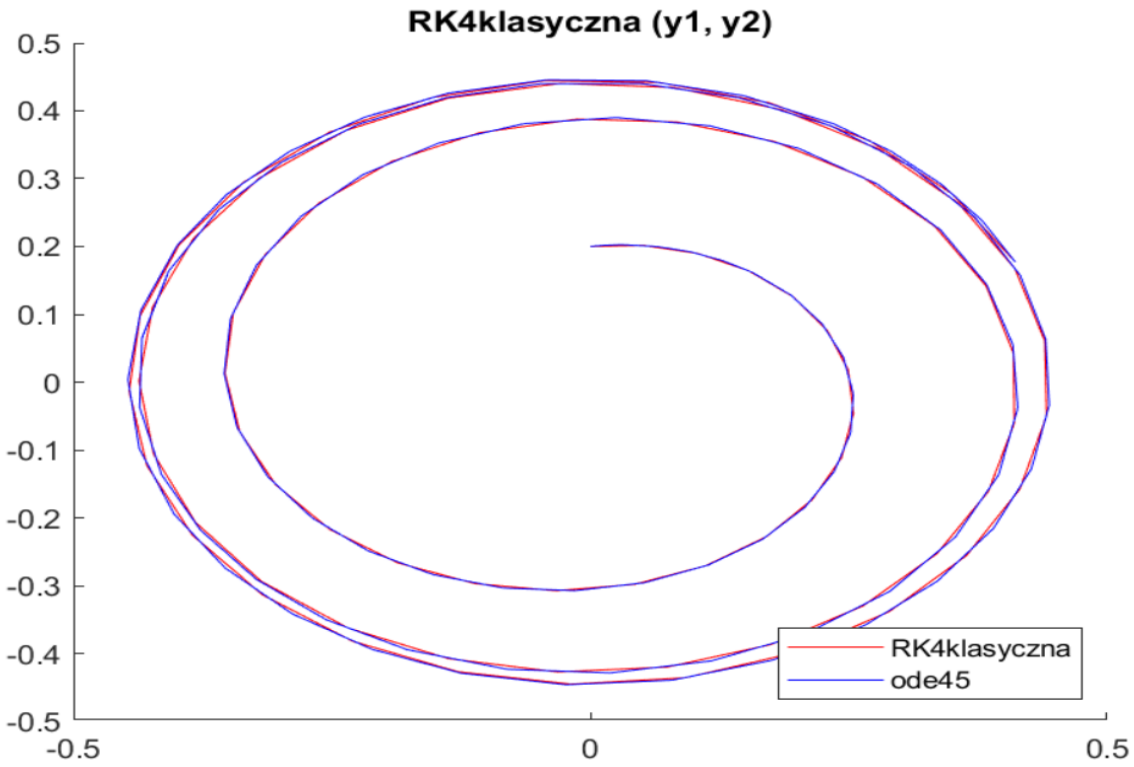
b) $x_1(0) = 0, x_2(0) = 0{,}2$

$$h = 0{,}5$$



Dla takiego kroku rozwiązanie nie jest jeszcze odpowiednio dobre, ale da się po nim określić kształt.

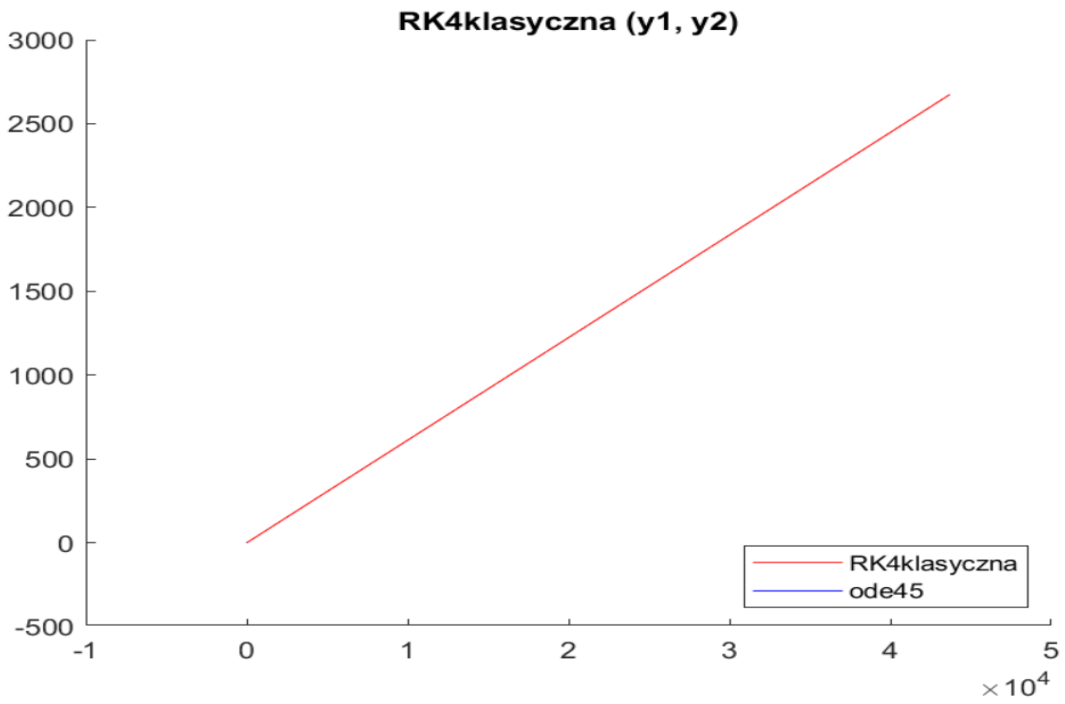
$$h = 0{,}25$$



Przy takim kroku mamy już całkiem dobre rozwiązanie, zbliżone do rozwiązania metodą ode45 i takie rozwiązanie uznamy za optymalne dla tego przypadku

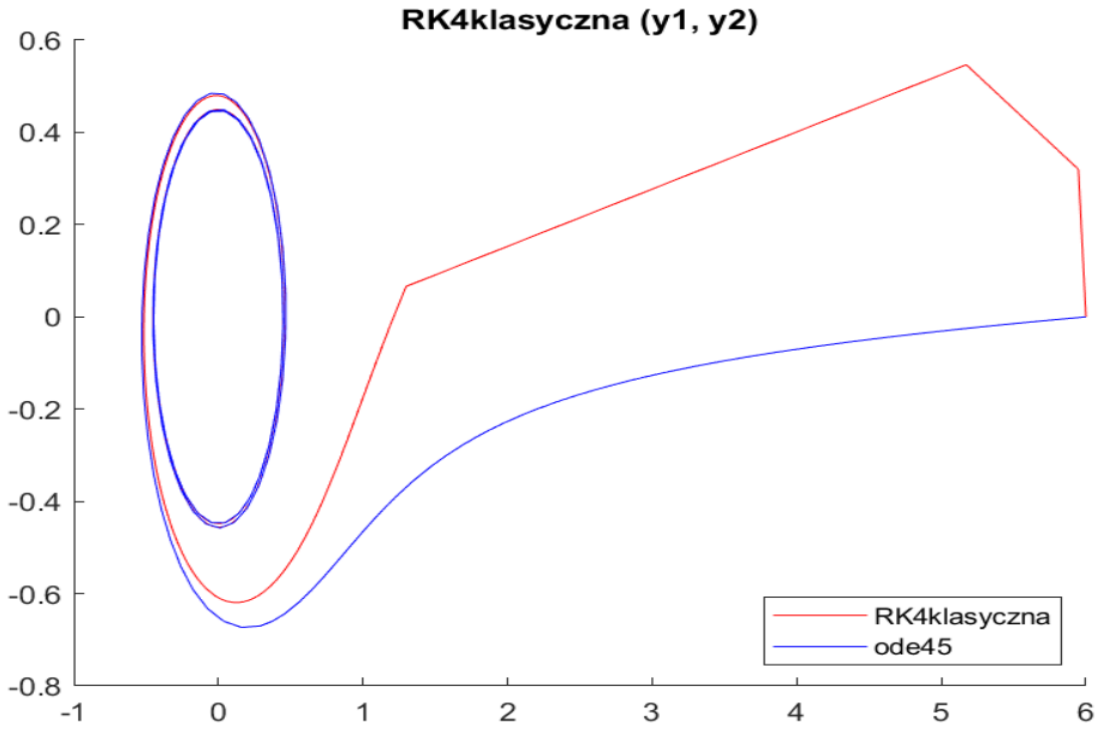
c) $x_1(0) = 6, x_2(0) = 0$

$$h = 0{,}1$$



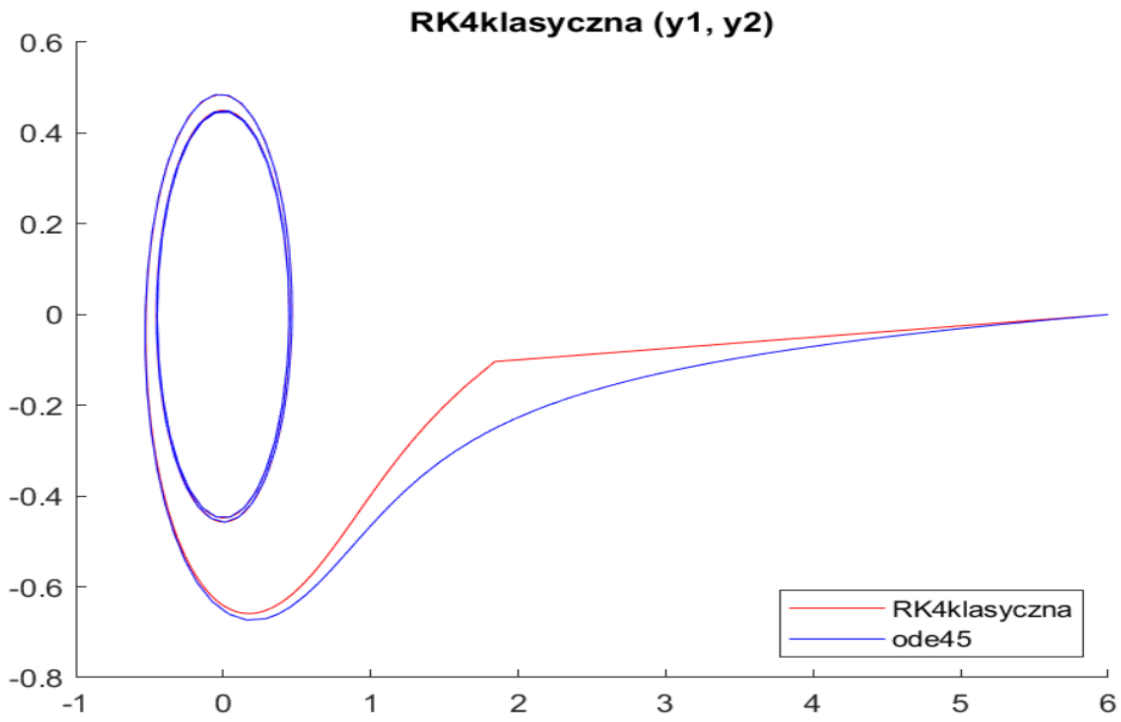
Dla tej wartości kroku metoda jest rozbieżna

$$h = 0{,}0668$$



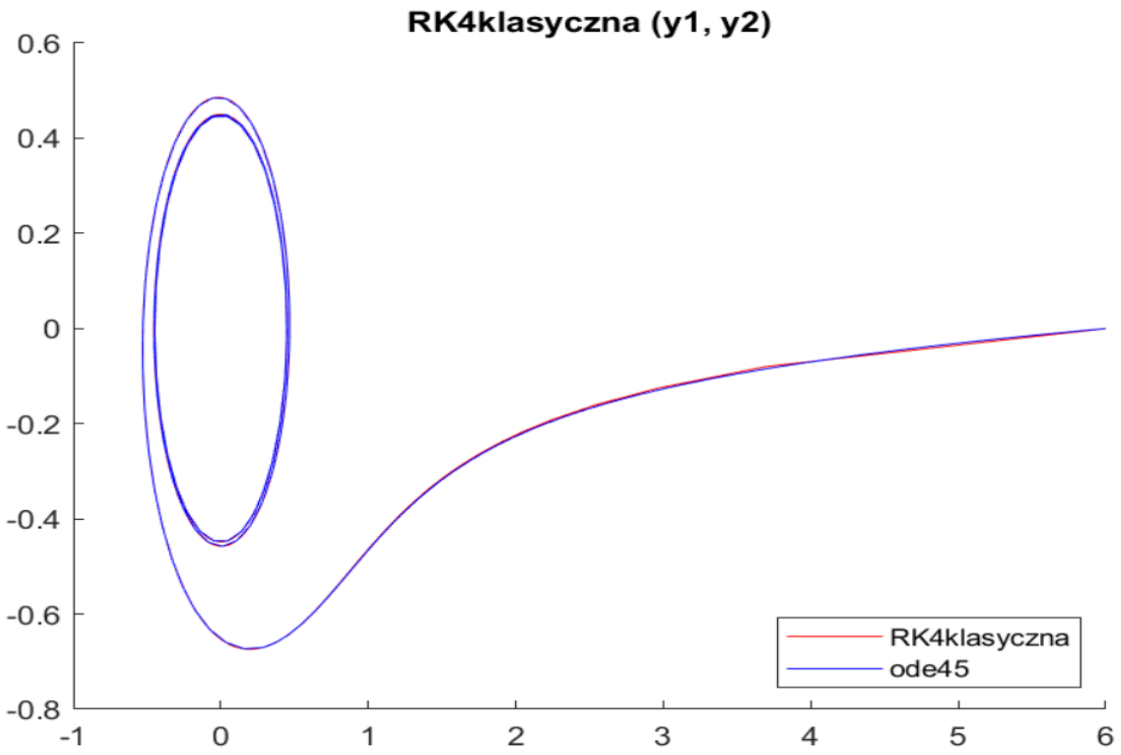
Jest to pierwsza znaleziona wartość kroku dla której metoda nie jest rozbieżna. Widać jednak bardzo duży błąd.

$$h = 0{,}04$$



Widzimy poprawę rozwiązania względem poprzedniej wartości, nadal jednak nie jest ono zadowalające

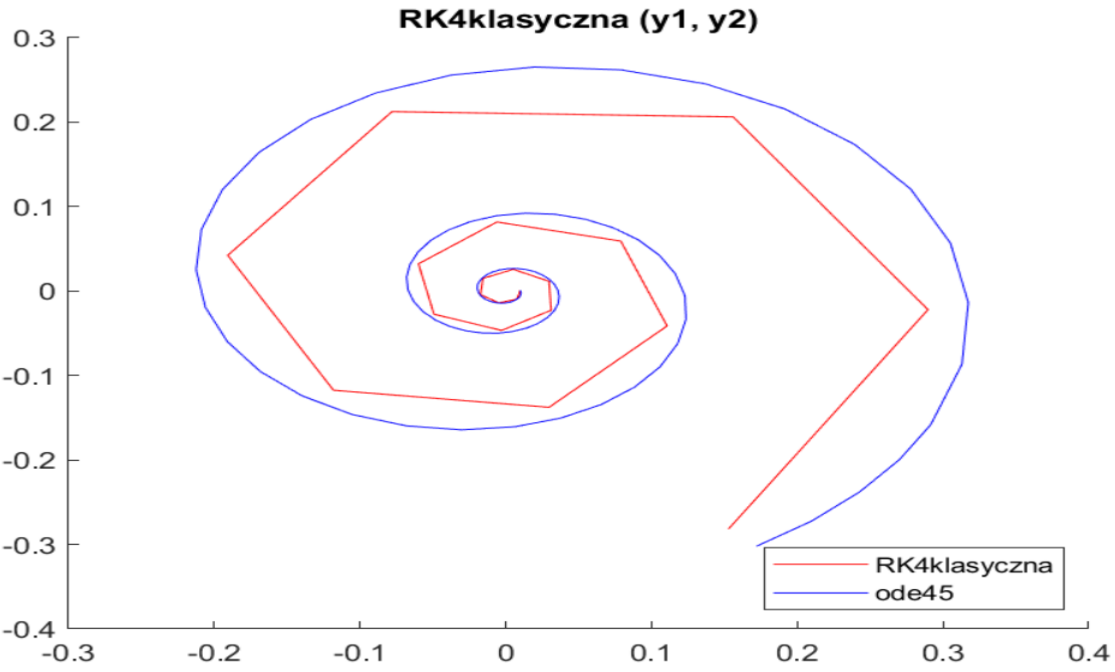
$$h = 0{,}021$$



Dla takiego kroku rozwiązanie jest już bardzo podobne jak to uzyskane referencyjną metodą ode45. W związku z tym optymalną wartością kroku w tym przypadku jest 0,021.

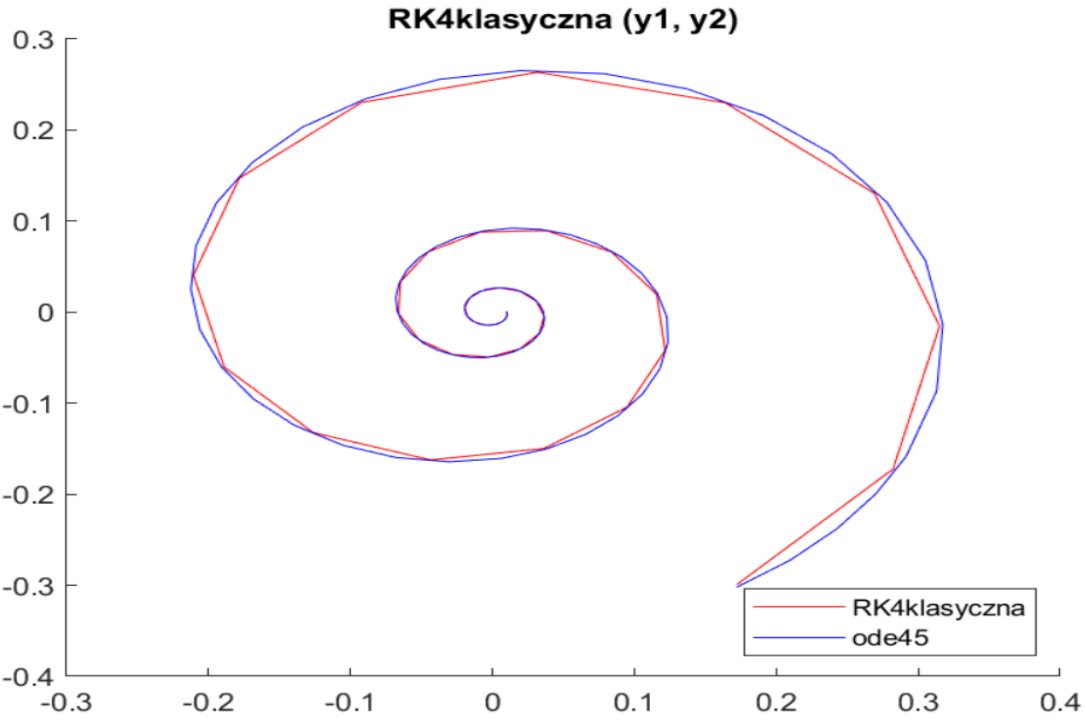
d) $x_1(0) = 0{,}01, x_2(0) = 0{,}001$

$$h = 1$$



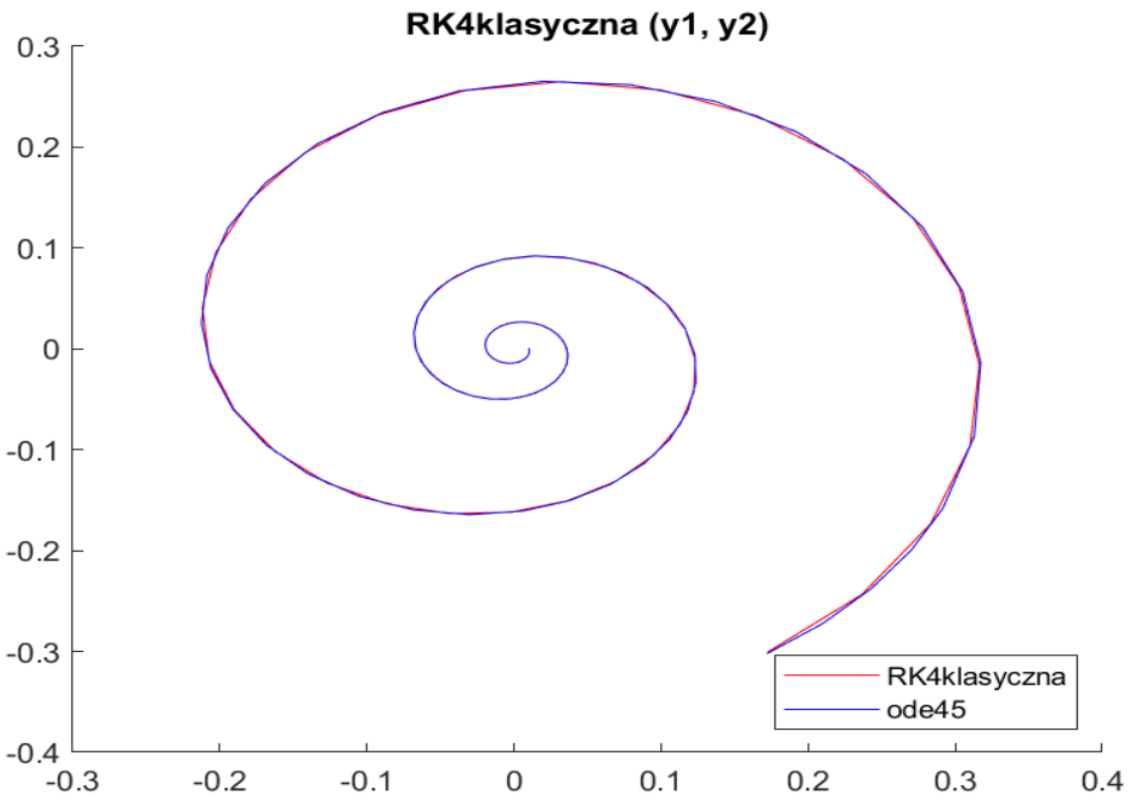
Widzimy tutaj bardzo niezadowalające rozwiązanie, jednak da się po nim określić ogólny kształt rozwiązania bardzo małym kosztem obliczeniowym

$$h = 0{,}5$$



Rozwiązanie jest nieco lepsze niż w przypadku poprzednim, jednak nadal odbiega od oczekiwanej dokładności.

$$h = 0{,}25$$



Dla takiego kroku rozwiązanie jest bardzo zbliżone do rozwiązania uzyskanego funkcją referencyjną ode45, w związku z tym uznajemy to za rozwiązanie optymalne

Metoda predyktor-korektor Adamsa czwartego rzędu ze stałym krokiem

Metody predyktor-korektor jest to rodzina metod wielokrokowych, które spełniają następujące założenia:

1) Wysoki rząd metody i niewielka stała błędu
2) Możliwie duży obszar absolutnej stabilności
3) Możliwie mała liczba obliczeń na iterację

Ogólny wzór algorytmu wygląda następująco

Predykcja: $y_n^{[0]} = \sum_{j=1}^{k} \alpha_i y_{n-j} + h \sum_{j=1}^{k} \beta_j f_{n-j}$

Ewaluacja: $f_n^{[0]} = f(x_n, y_n^{[0]})$

Korekcja: $y_n = \sum_{j=1}^{k} \alpha_j^* y_{n-j} + h \sum_{j=1}^{k} \beta_j^* f_{n-j} + h\beta_0^* f_n$

Ewaluacja: $f_n = f(x_n, y_n)$

Predyktor – metoda jawna (wartość w punkcie bieżącym zależy jedynie od wartości w punktach poprzednich) odpowiada za wyznaczenie dobrego punktu początkowego dla iteracji korektora.

Korektor – metoda niejawna (wartość w punkcie bieżącym zależy od wartości w punktach poprzednich oraz w punkcie bieżącym) rozwiązująca nieliniowe równanie algebraiczne.

W naszym przypadku rolę predyktora pełni jawna metoda Adamsa, a korektora niejawna metoda Adamsa. Ogólna postać metod Adamsa wynika z faktu, że równanie różniczkowe równoważne jest równaniu całkowemu, skąd rozważając je na przedziale odpowiadającym pojedynczej iteracji mamy:

$$y(x_n) = y(x_{n-1}) + \int_{x_{n-1}}^{x_n} f\big(t, y(t)\big)dt$$

Dla metody jawnej po scałkowaniu otrzymujemy

$$y_n = y_{n-1} + h\sum_{j=1}^{k} \beta_j f(x_{n-j}, y_{n-j})$$

Dla metody niejawnej:

$$y_n = y_{n-1} + h\beta_0^* \cdot f(x_n, y_n) + h\sum_{j=1}^{k} \beta_j^* \cdot f(x_{n-j}, y_{n-j})$$

Współczynniki β_j, β_j^* są stabelaryzowane.

Wobec tego, dla metod Adamsa algorytm predyktor-korektor wygląda następująco

Predykcja: $y_n^{[0]} = y_{n-1} + h\sum_{j=1}^{k} \beta_j f_{n-j}$

Ewaluacja: $f_n^{[0]} = f(x_n, y_n^{[0]})$

Korekcja: $y_n = y_{n-1} + h\sum_{j=1}^{k} \beta_j^* f_{n-j} + h\beta_0^* \cdot f_n^{[0]}$

Ewaluacja: $f_n = f(x_n, y_n)$

Dla metody Adamsa rzędu p należy wyznaczyć pierwsze p wartości rozwiązania metodą jednokrokową (w naszym przypadku wykorzystano tutaj metodę RK4)

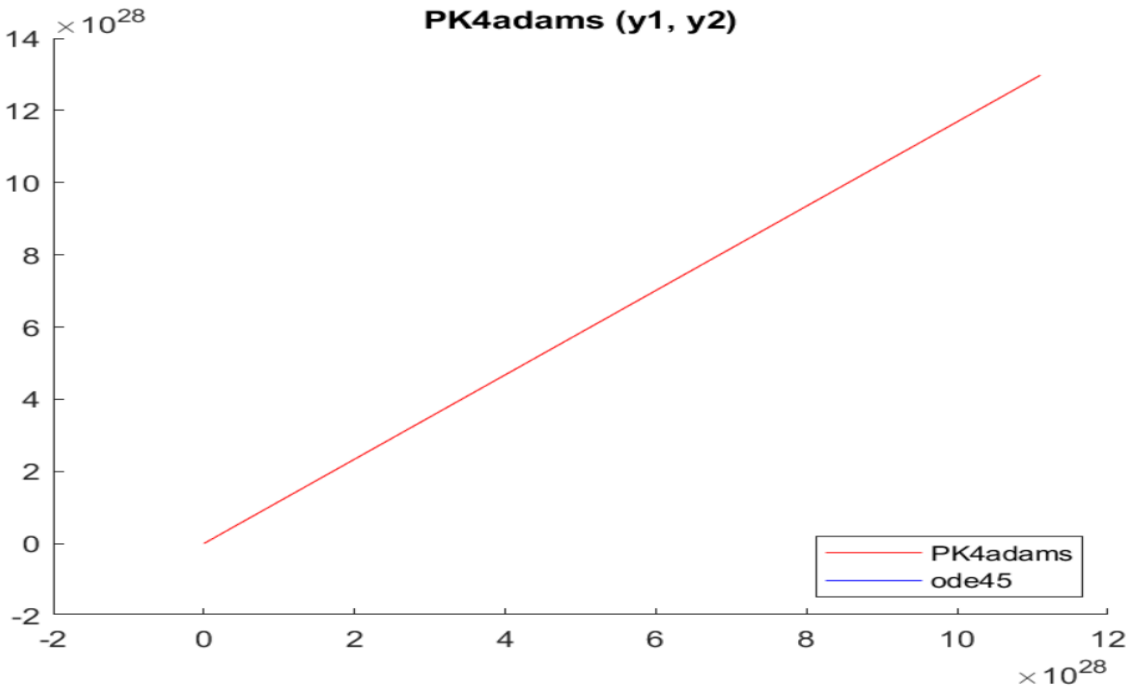
Implementacja w języku MATLAB

```
function [x, y] = PK4adams(f, y0, a, h, ~)

    % wartości współczynników beta dla metody Adamsa 4 rzędu
    betap = [55/24, -59/24, 37/24, -9/24];                      % dla predyktora (metoda Adamsa jawna)
    betak = [251/720, 646/720, -264/720, 106/720, -19/720];     % dla korektora (metoda Adamsa niejawna)

    n = length(y0);                         % wymiarowość układu
    x = (a(1):h:a(2))';                     % chwile czasu w których wyznaczane będą wartości
    y = zeros(length(x), n);                % macierz na rozwiązania y(i,j) - wartość j-tej funkcji dla xi

    % wyznaczenie pierwszych czterech wartości algorytmem RK4klasyczna
    if (a(2) < a(1) + 3 * h)
        error("Zbyt duży krok algorytmu, metoda nieskuteczna");
    end
    % [~, y(1:4, :)] = RK4klasyczna(f, y0, [a(1), a(1) + 3*h], h, eps);
    y(1, :) = y0;
    for i = 1:3
        k(1, :) = f(x(i), y(i, :));
        k(2, :) = f(x(i) + 0.5 * h, y(i, :) + 0.5 * h * k(1, :));
        k(3, :) = f(x(i) + 0.5 * h, y(i, :) + 0.5 * h * k(2, :));
        k(4, :) = f(x(i) + h, y(i, :) + h * k(3, :));

        y(i + 1, :) = y(i, :) + (1 / 6) * h * (k(1, :) + 2 * k(2, :) + 2 * k(3, :) + k(4, :));
    end

    % Predyktor - Korektor
    for i = 5:length(x)
        % predykcja i ewaluacja
        tmp = zeros(1, n);
        for j = 1:4
            tmp = tmp + betap(j) * f(x(i - j), y(i - j, :))';
        end
        yp = y(i - 1, :) + h * tmp;
        % korekcja i ewaluacja
        tmp = zeros(1, n);
        for j = 1:4
            tmp = tmp + betak(j + 1) * f(x(i - j), y(i - j, :))';
        end
        y(i, :) = y(i - 1, :) + h * (tmp + betak(1) * f(x(i), yp)');
    end
end
```

Wyniki

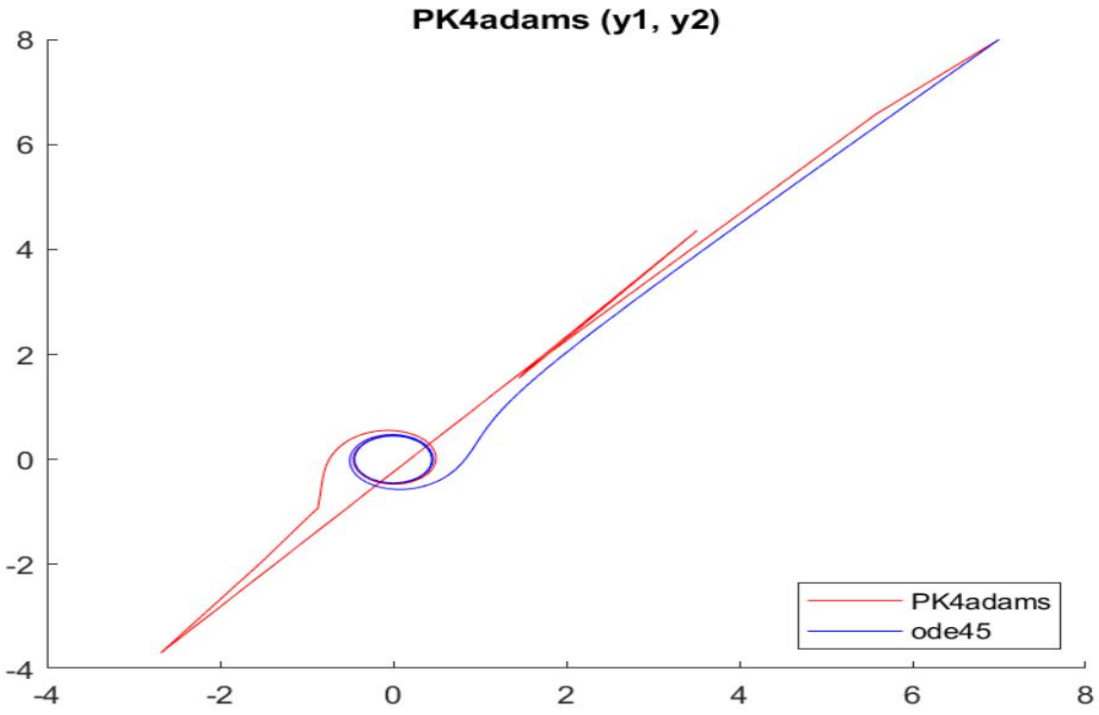
a) $x_1(0) = 7, x_2(0) = 8$

$$h = 0{,}1$$



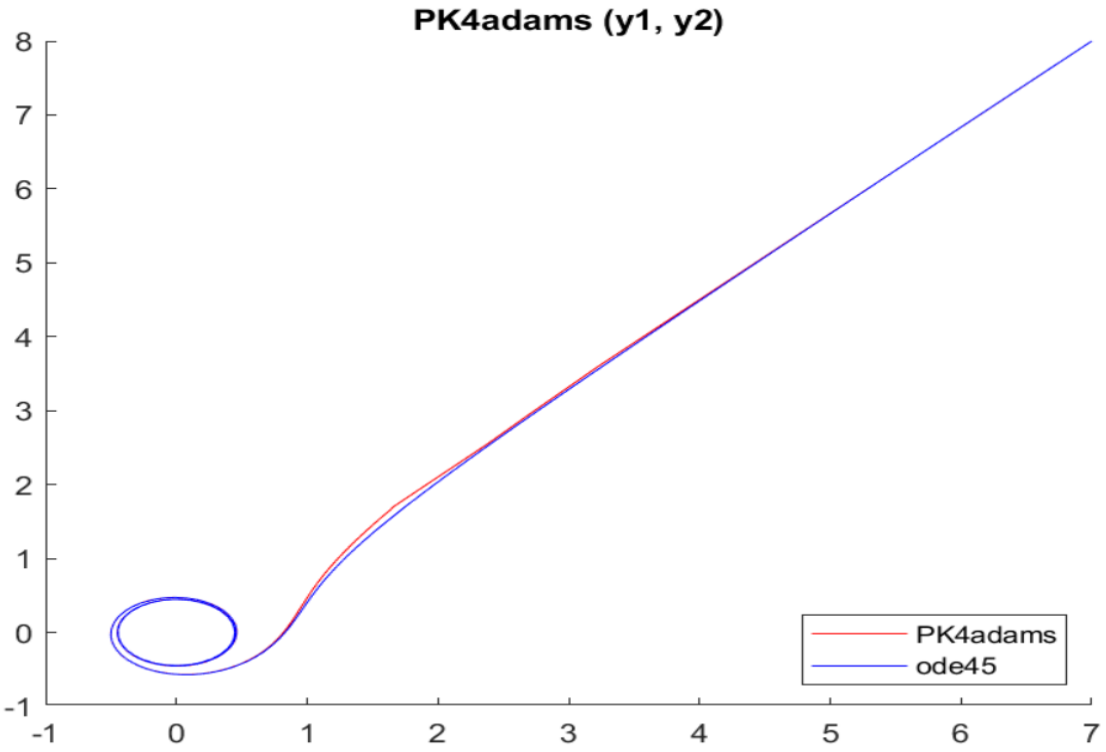
Dla tak dużego kroku metoda jest rozbieżna

$$h = 0{,}02065$$



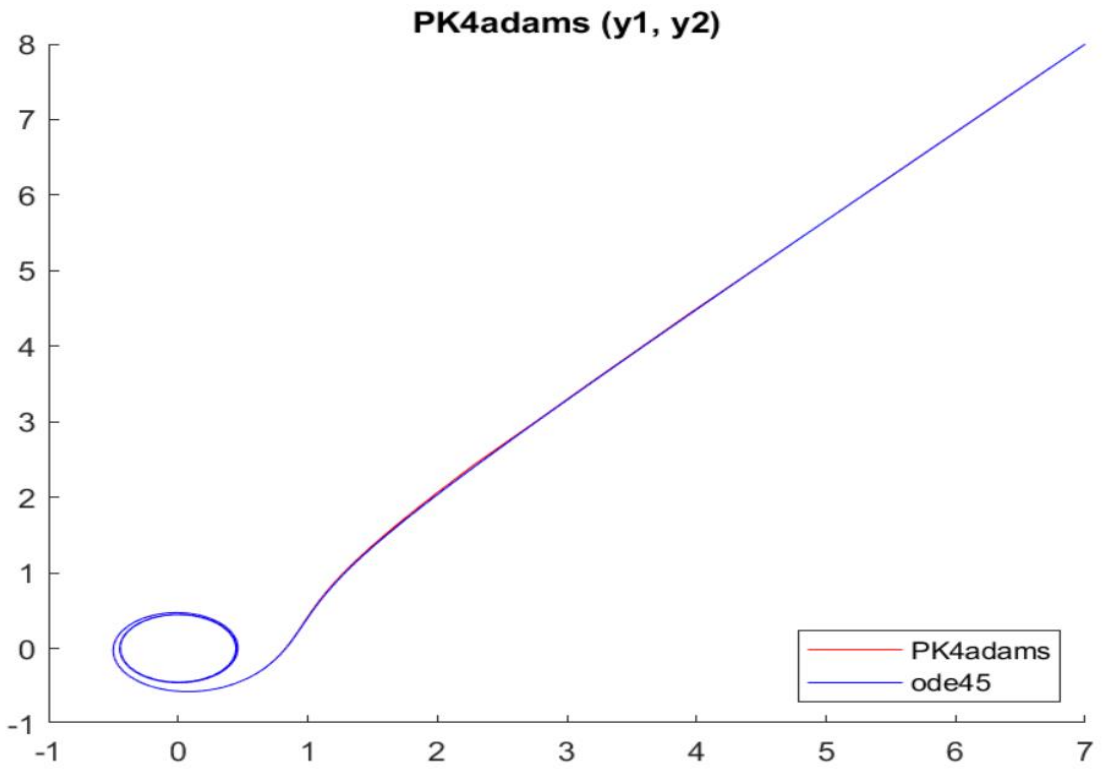
Jest to największy krok, dla którego metoda nie jest już rozbieżna, jednak dokładność rozwiązania jest bardzo słaba.

$h = 0{,}01$



Ogólne rozwiązanie jest już całkiem dokładne, pojawiają się jednak lokalnie duże błędy

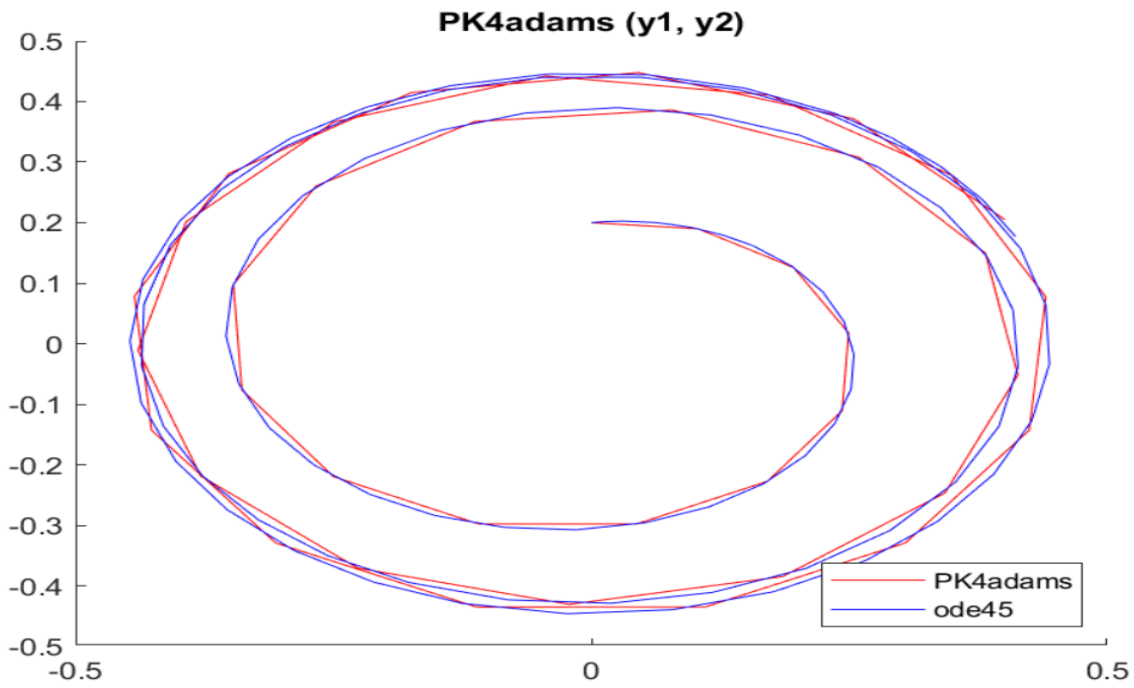
$h = 0{,}008$



To rozwiązanie jest już całkiem dokładne, bardzo zbliżone do rozwiązania uzyskanego referencyjną metodą ode45. W związku z tym dla tego przypadku optymalnym krokiem jest 0,008

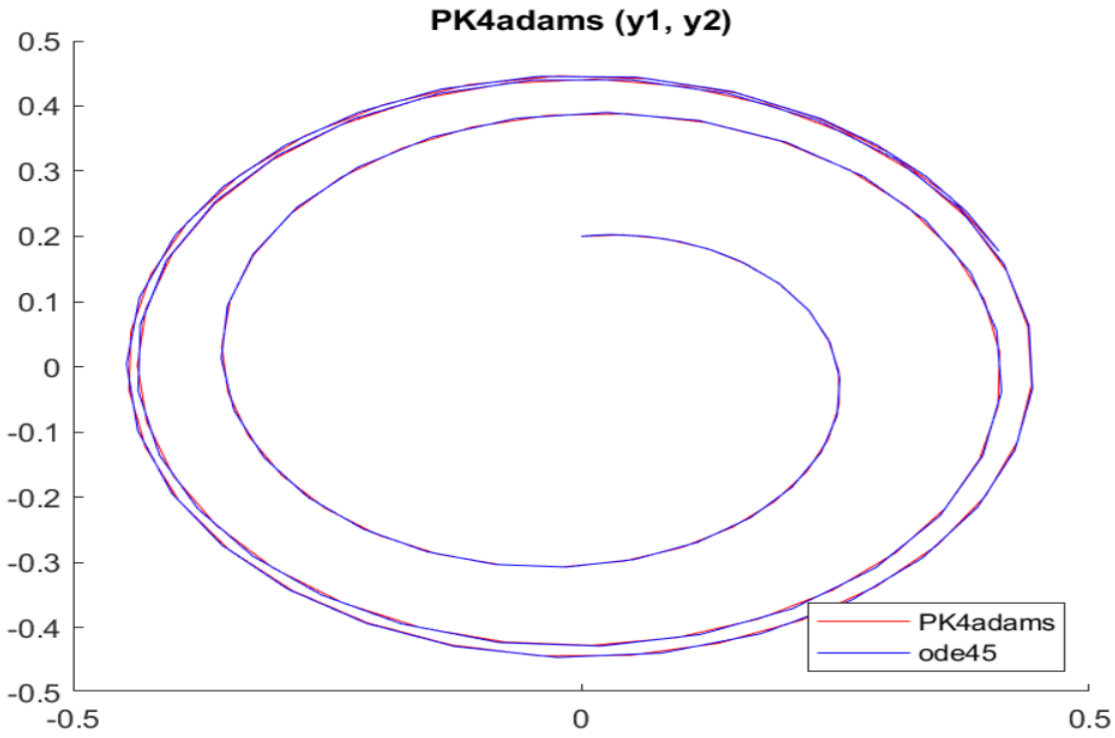
b) $x_1(0) = 0, x_2(0) = 0{,}2$

$$h = 0{,}5$$



Rozwiązanie jest obarczone dużą niedokładnością, jednak da się po nim określić ogólny przebieg rozwiązania niewielkim kosztem obliczeniowym.

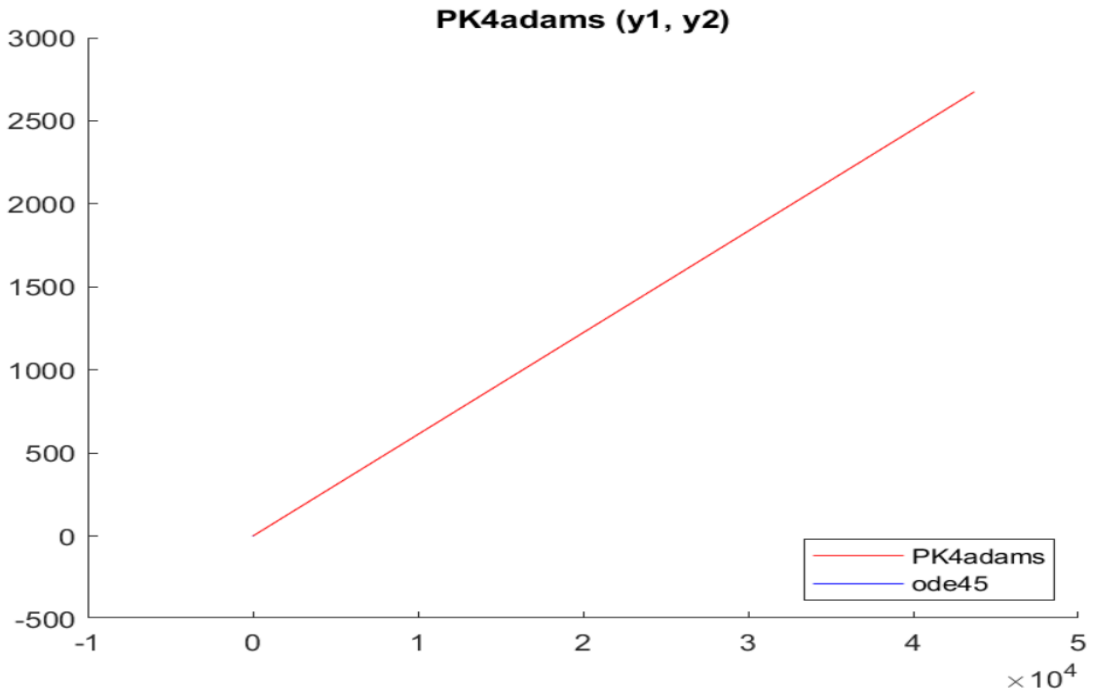
$$h = 0{,}2$$



Rozwiązanie to jest bardzo zbliżone do rozwiązania uzyskanego przy wykorzystaniu funkcji referencyjnej ode45, wobec tego za wartość optymalną kroku uznajemy 0,2

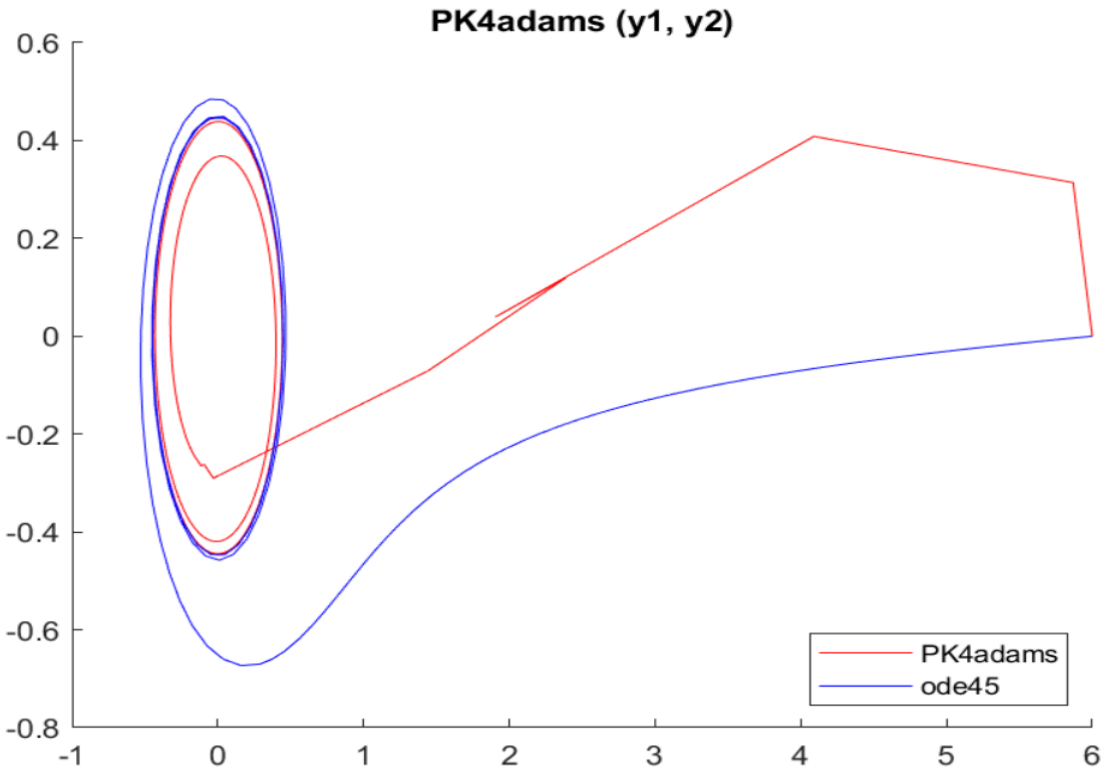
c) $x_1(0) = 6, x_2(0) = 0$

$$h = 0{,}1$$



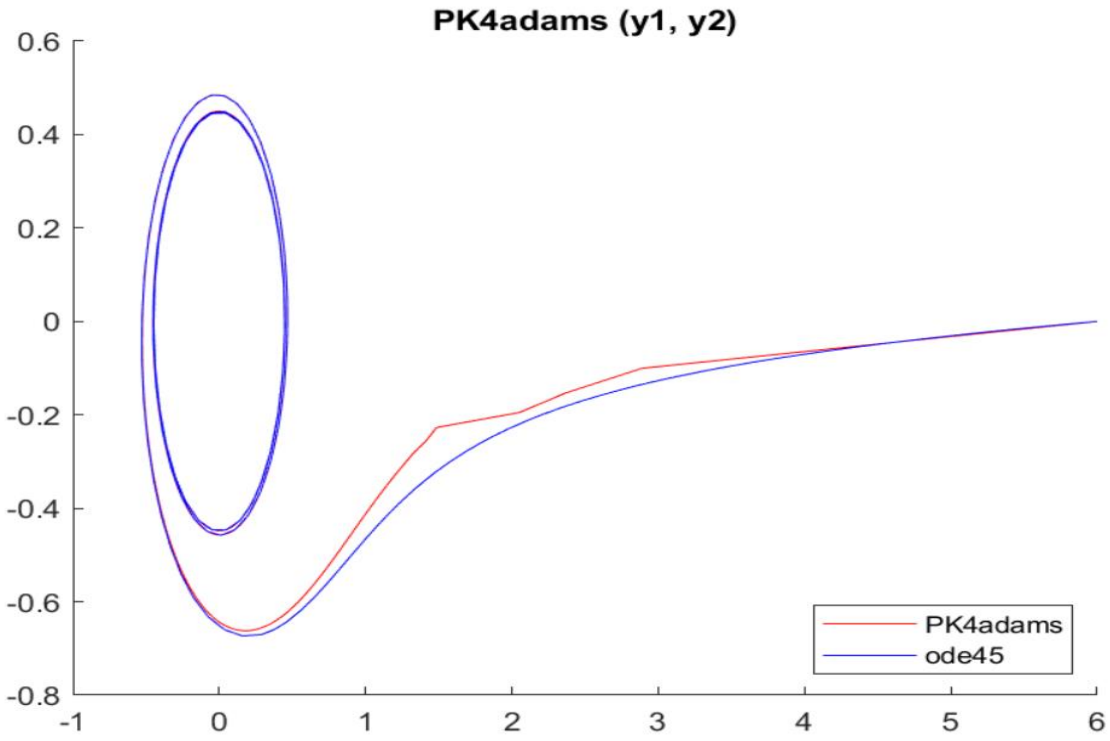
Dla takiej wartości kroku metoda jest rozbieżna

$$h = 0{,}0667$$



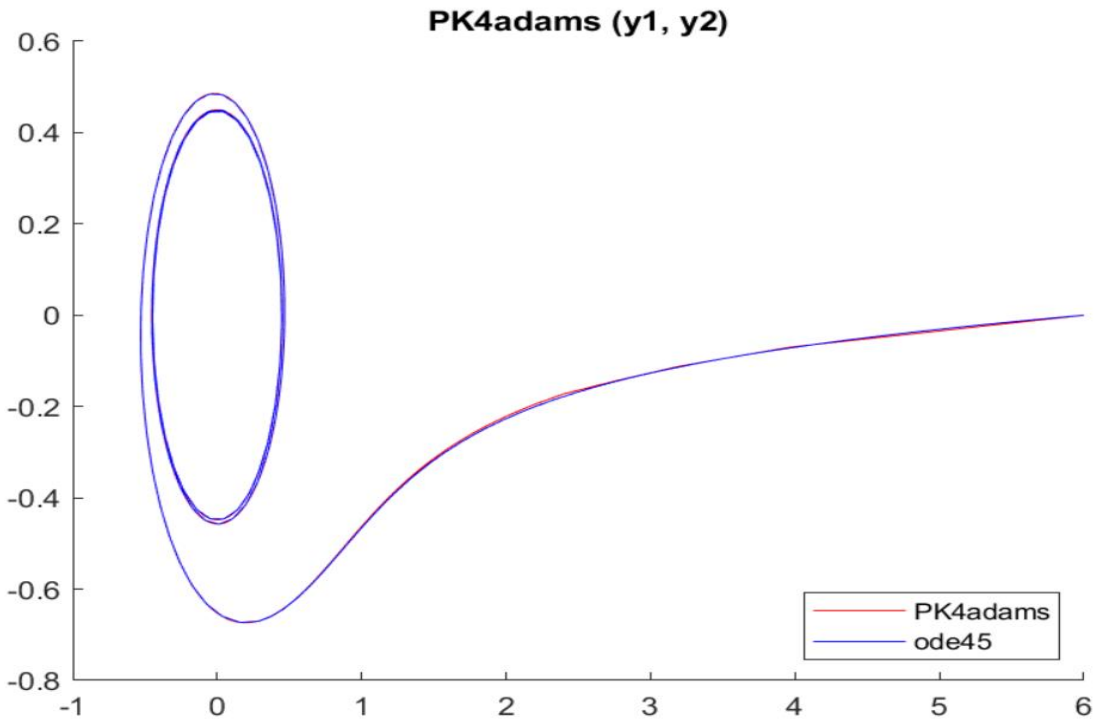
Jest to pierwsza wartość kroku, dla której metoda nie jest rozbieżna. Błąd rozwiązania jest znaczący, jednak pozwala na określenie kształtu funkcji niewielkim kosztem obliczeniowym

$$h = 0{,}03$$



Rozwiązanie jest już dużo lepsze niż w przypadku poprzednich wartości kroku, jednak nadal widzimy lokalnie występujące duże błędy.

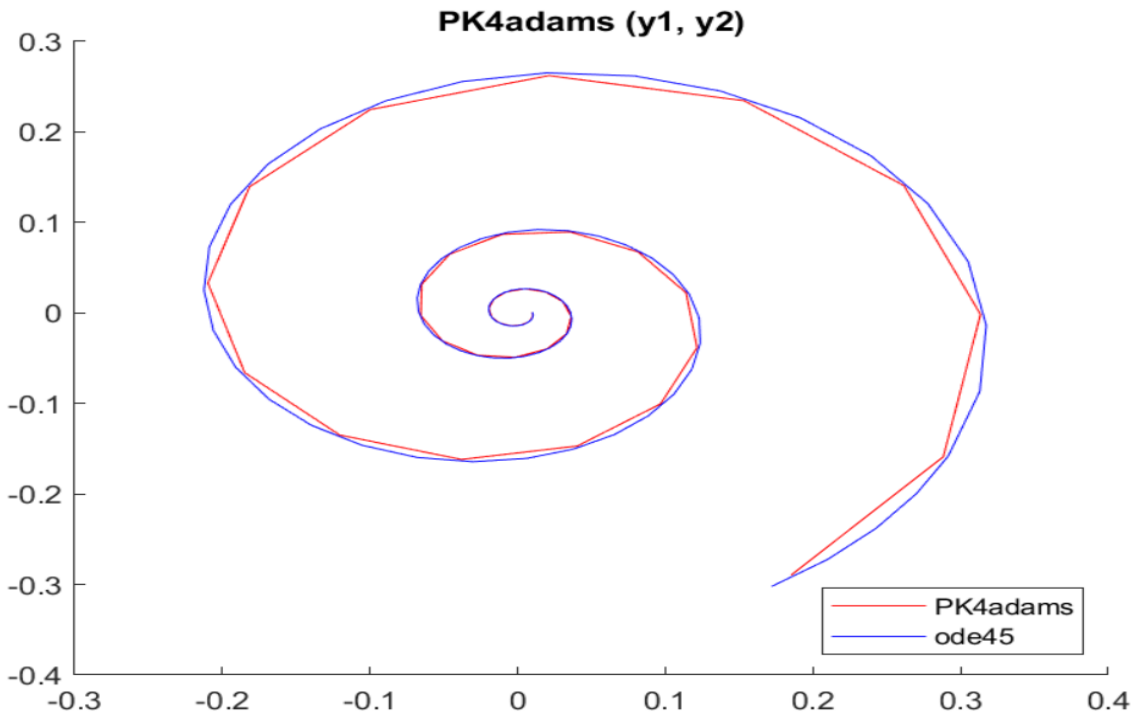
$$h = 0{,}0175$$



Rozwiązanie jest już bardzo zbliżone do tego uzyskanego referencyjną metodą ode45, w związku z tym uznajemy wartość kroku 0,0175 za optymalną.

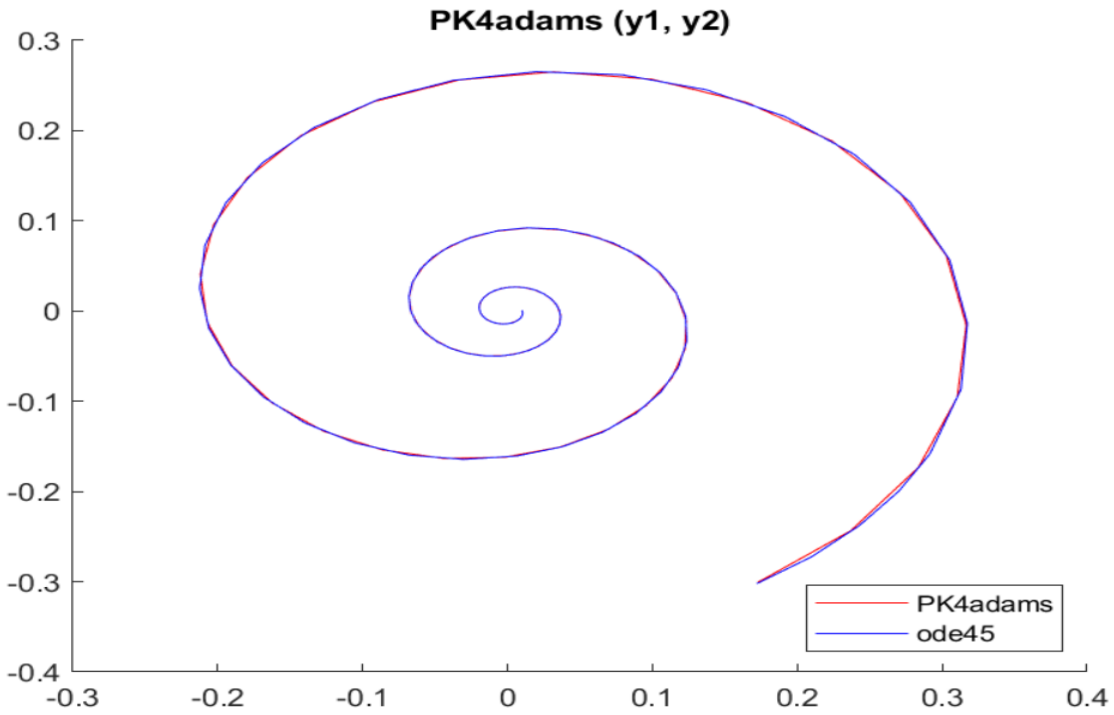
d) $x_1(0) = 0{,}01, x_2(0) = 0{,}001$

$$h = 0{,}5$$



Dla takiego kroku rozwiązanie jest całkiem dobre. Lokalnie pojawiają się większe błędy, ale ogólny kształt i przebieg funkcji są dobrze odwzorowane.

$$h = 0{,}25$$



Dla takiego kroku widzimy że uzyskane rozwiązanie jest bardzo zbliżone do rozwiązania referencyjnego, w związku z tym uznajemy wartość kroku 0,25 za optymalną.

Metoda Rungego-Kutty czwartego rzędu ze zmiennym krokiem

Głównym problemem metod ze stałym krokiem jest fakt, że krok musi być dobrany w taki sposób, aby w przedziałach najszybszej zmienności rozwiązania było ono wystarczająco dokładne. Jednakże w przedziałach o mniejszej zmienności powoduje to wykonywanie niepotrzebnych iteracji. Rozwiązaniem tego problemu są metody ze zmiennym krokiem, który „dostosowuje" się do lokalnej zmienności funkcji. Wykorzystuje się do tego oszacowanie lokalnej wartości błędu metody.

W przypadku metody Rungego-Kutty oszacowanie to uzyskiwane jest poprzez wykorzystanie zasady podwójnego kroku. W tym celu, oprócz kroku o długości h wykonywane są dodatkowo dwa kroki o długości $0{,}5h$. Oszacowanie tego błędu ma wówczas wartość

Dla pojedynczego kroku o długości h:

$$\delta(h) = \frac{2^p}{2^p - 1}(y_n^{(2)} - y_n^{(1)})$$

Dla dwóch kroków o długości $0{,}5h$:

$$\delta\left(2 \times \frac{h}{2}\right) = \frac{y_n^{(2)} - y_n^{(1)}}{2^p - 1}$$

Gdzie p – rząd metody, $y_n^{(1)}$ – punkt uzyskany w kroku o długości h, $y_n^{(2)}$- punkt uzyskany w kroku o długości $0{,}5h$.

Widać tutaj, że $\frac{\delta(h)}{\delta\left(2\times\frac{h}{2}\right)} = 2^p$, a więc rozwiązanie $y_n^{(2)}$ jest rozwiązaniem dokładniejszym. Wobec tego jako rozwiązanie danej iteracji przyjmuje się wartość $y_{n+1} = y_n^{(2)}$.

Zmiana kroku polega na wyznaczeniu współczynnika modyfikacji kroku w każdej iteracji. Współczynnik ten przyjmuje wartość

$$\alpha = \min_{1 \le i \le k} \left(\frac{\varepsilon_i}{|\delta_n(h)_i|}\right)^{\frac{1}{p+1}}$$

Gdzie $\delta_n(h) = \delta_n\left(2 \times \frac{h}{2}\right) = \frac{y_n^{(2)} - y_n^{(1)}}{2^p - 1}$, ponieważ tak jak zostało zapisane wcześniej, za rozwiązanie w danej iteracji przyjmujemy wartość uzyskaną w dwóch krokach o długości $0{,}5h$.

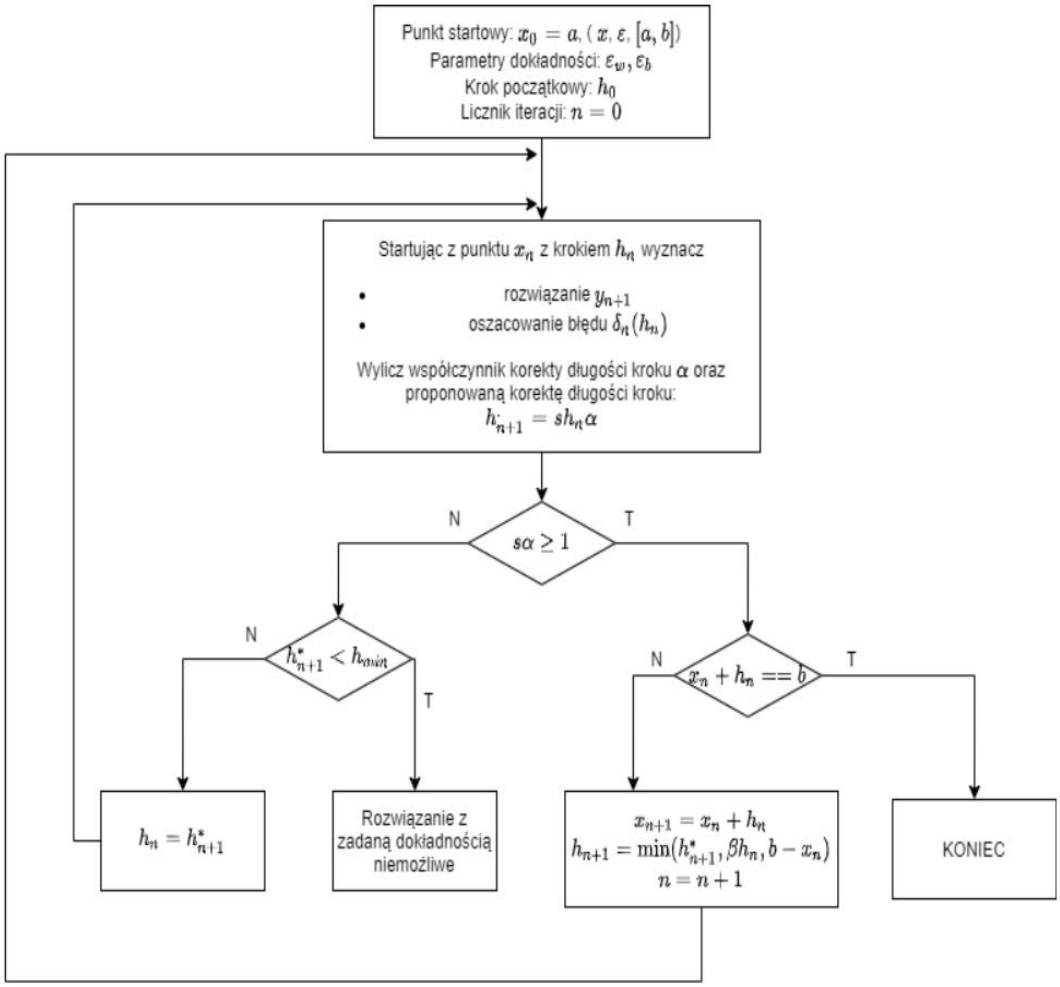
$$\varepsilon = |y_n|\varepsilon_w + \varepsilon_b$$

Oraz ε_w – dokładność względna, ε_b – dokładność bezwzględna są parametrami użytkownika

Nowy krok wyznaczany jest z zależności:

$$h_{n+1} = s \cdot \alpha \cdot h_n$$

s jest współczynnikiem bezpieczeństwa uwzględniającym niedokładność oszacowania błędu ($s \approx 0{,}9$). W praktyce stosuje się jeszcze heurystyczny współczynnik $\beta > 1$, który ogranicza od góry maksymalny wzrost długości kroku w pojedynczej iteracji. Ogólny schemat algorytmu


Punkt startowy: $x_0 = a$, (x, ε, $[a, b]$)
Parametry dokładności: $\varepsilon_w, \varepsilon_b$
Krok początkowy: h_0
Licznik iteracji: $n = 0$
Startując z punktu x_n z krokiem h_n wyznacz
rozwiązanie y_{n+1}
oszacowanie błędu $\delta_n(h_n)$
Wylicz współczynnik korekty długości kroku α oraz proponowaną korektę długości kroku:
$h^*_{n+1} = sh_n\alpha$
N
$s\alpha \geq 1$
T
N
$h^*_{n+1} < h_{min}$
T
N
$x_n + h_n == b$
T
$h_n = h^*_{n+1}$
Rozwiązanie z zadaną dokładnością niemożliwe
$x_{n+1} = x_n + h_n$
$h_{n+1} = \min(h^*_{n+1}, \beta h_n, b - x_n)$
$n = n + 1$
KONIEC

Implementacja w języku MATLAB

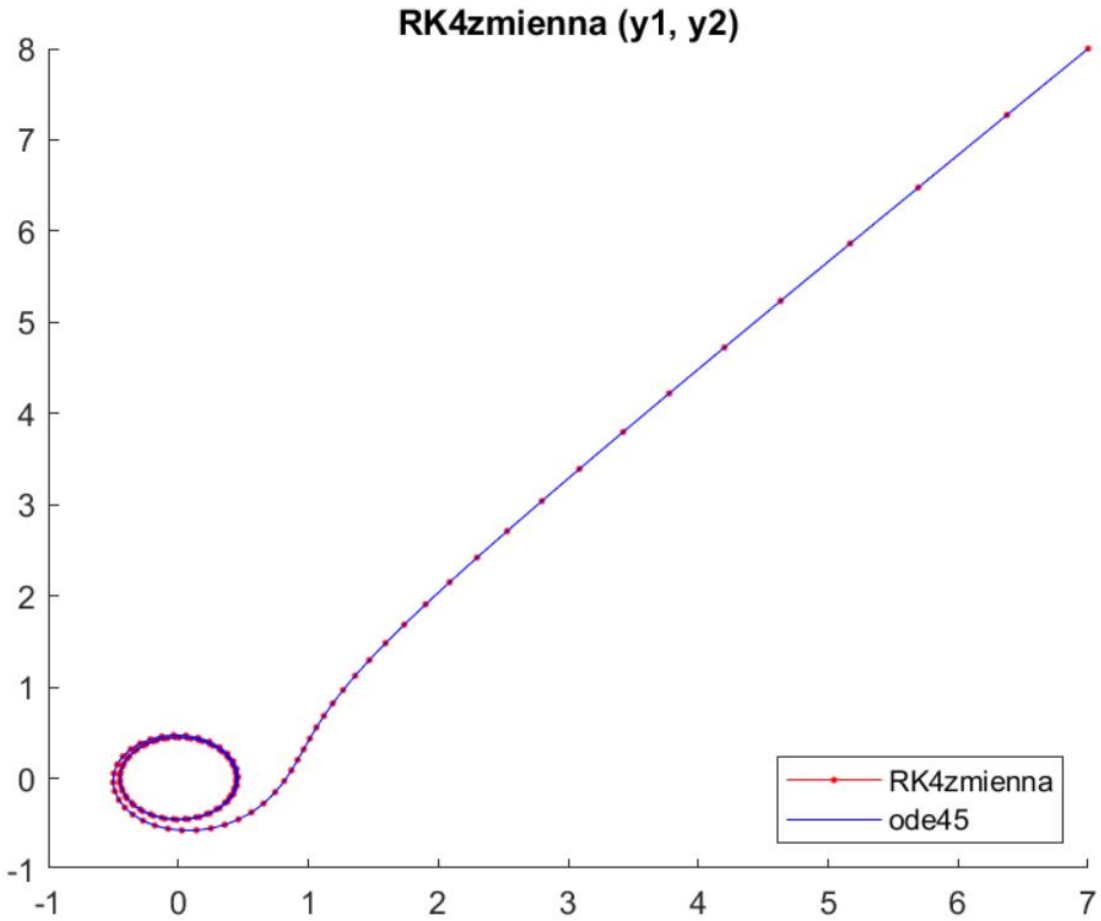
```
function [x, y] = RK4zmienna(f, y0, a, h, eps)

    x = zeros(2, 1);
    x(1) = a(1);
    y(1, :) = y0;
    s = 0.9;
    i = 1;
    beta = 1.5;
    hmin = 10e-12;

    %iteracja
    while (x(i) <= a(2))
        % wyznaczenie rozwiązania y(i+1) metodą RK
%         [~, odp] = RK4klasyczna(f, y(i, :), [x(i), x(i) + h], h, eps);
%         y1 = odp(2, :);
        k(1, :) = f(x(i), y(i, :));
        k(2, :) = f(x(i) + 0.5 * h, y(i, :) + 0.5 * h * k(1, :));
        k(3, :) = f(x(i) + 0.5 * h, y(i, :) + 0.5 * h * k(2, :));
        k(4, :) = f(x(i) + h, y(i, :) + h * k(3, :));

        y1 = y(i, :) + (1 / 6) * h * (k(1, :) + 2 * k(2, :) + 2 * k(3, :) + k(4, :));

        % wyznaczenie rozwiązania wg zasady podwójnego kroku
%         [~, odp] = RK4klasyczna(f, y(i, :), [x(i), x(i) + h], h / 2, eps);
%         y(i + 1, :) = odp(3, :);
        % 1 krok
        k(1, :) = f(x(i), y(i, :));
        k(2, :) = f(x(i) + 0.5 * (h / 2), y(i, :) + 0.5 * (h / 2) * k(1, :));
        k(3, :) = f(x(i) + 0.5 * (h / 2), y(i, :) + 0.5 * (h / 2) * k(2, :));
        k(4, :) = f(x(i) + h / 2, y(i, :) + (h / 2) * k(3, :));

        tmp_y = y(i, :) + (1 / 6) * (h / 2) * (k(1, :) + 2 * k(2, :) + 2 * k(3, :) + k(4, :));

        % 2 krok
        tmp_x = x(i) + h / 2;
        k(1, :) = f(tmp_x, tmp_y);
        k(2, :) = f(tmp_x + 0.5 * (h / 2), tmp_y + 0.5 * (h / 2) * k(1, :));
        k(3, :) = f(tmp_x + 0.5 * (h / 2), tmp_y + 0.5 * (h / 2) * k(2, :));
        k(4, :) = f(tmp_x + h / 2, tmp_y + (h / 2) * k(3, :));

        y(i + 1, :) = tmp_y + (1 / 6) * (h / 2) * (k(1, :) + 2 * k(2, :) + 2 * k(3, :) + k(4, :));

        % oszacowanie błędu
        delta = (y(i + 1, :) - y1) / 15;        % 15 == 2^p - 1, p - rząd metody
        epse = abs(y(i + 1, :)) * eps(1) + eps(2);

        % współczynnik korekty długości kroku alfa
        alfa = min(epse / abs(delta))^(1 / 5);       % 1 / 5 == 1 / (p + 1), p - rząd metody

        %korekta długości kroku
        hprop = s * alfa * h;

        if (s * alfa < 1) % poprawka
            if (hprop < hmin)
                error("Nie da się znaleźć rozwiązania z podaną dokładnością, wybierz mniejsze wartości eps");
            else
                h = hprop;
            end
        else %kolejny punkt
            x(i + 1) = x(i) + h;
            h = min([hprop, beta * h, a(2) - x(i)]);
            i = i + 1;
        end
    end
end
```

Wyniki

W przypadku tej metody na wykresach zaznaczono także punkty, w których wyznaczane było rozwiązanie, aby pokazać zmienność kroku w zależności od lokalnej zmienności rozwiązania. Jako krok startowy wybierano taki krok, dla którego metoda zwracała rozwiązanie. Wartości błędów ustalono na poziomie $\varepsilon_w = \varepsilon_b = 10^{-8}$ tak, aby liczba iteracji odpowiadała mniej więcej liczbie iteracji wykonywanych przez metodę ode45 dla danego problemu.

a) $x_1(0) = 8, \qquad x_2(0) = 7$

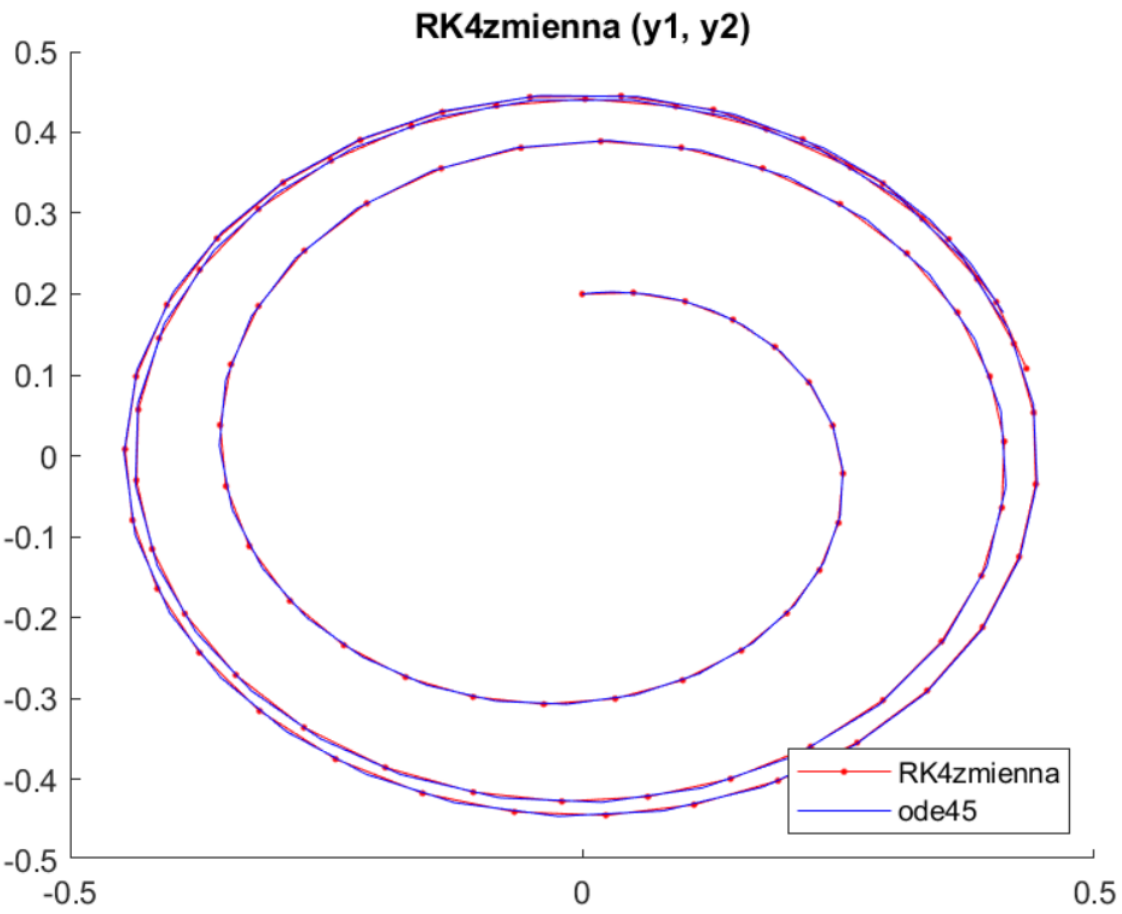
$$h_0 = 0{,}06$$



Rozwiązanie jest bardzo dobre, widać też że krok jest mały w otoczeniu szybkiej zmienności funkcji oraz zwiększa się, gdy funkcja ma liniowy wzrost

b) $x_1(0) = 0, \qquad x_2(0) = 0{,}2$

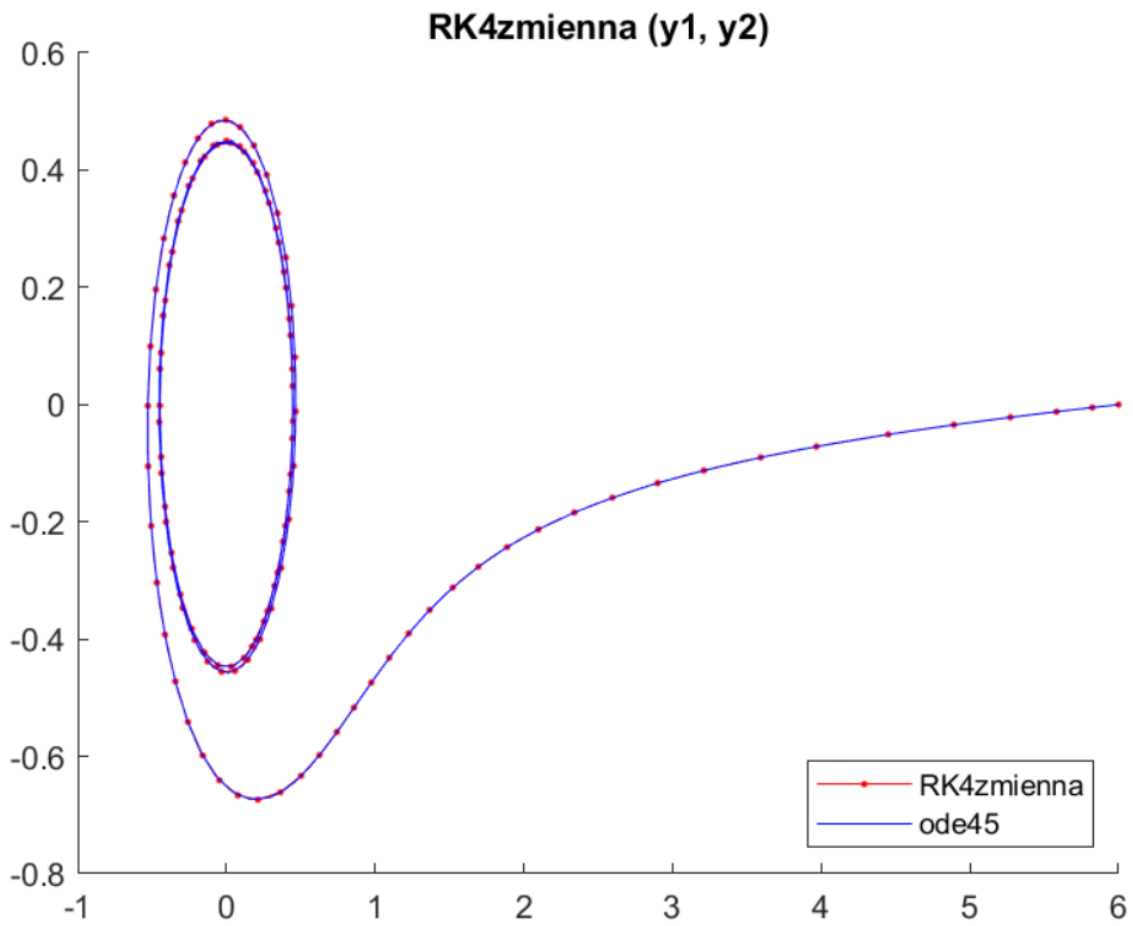
$$h = 1$$



Uzyskane rozwiązanie również jest bardzo dobre, widać że zmienność rozwiązania jest globalnie w miarę stabilna, dlatego też kroki całkowania są rozłożone równomiernie na całej krzywej.

c) $x_1(0) = 6, \qquad x_2(0) = 0$

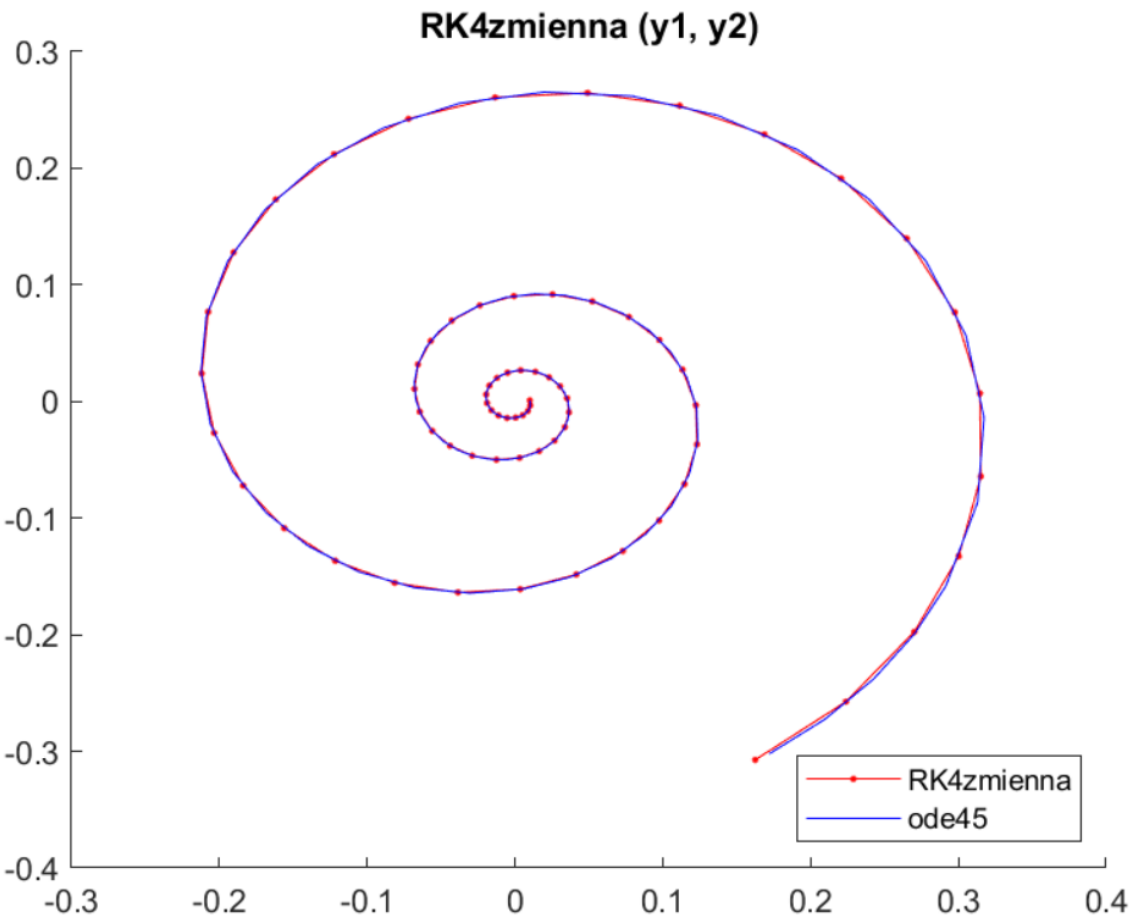
$$h = 0{,}1$$



Rozwiązanie jest bardzo zbliżone do referencyjnej metody ode45, widać również że mniejsza wartość kroku jest w obszarach szybkiej zmienności rozwiązania, a większa w obszarach gdzie wartość drugiej pochodnej jest niewielka.

d) $x_1(0) = 0{,}01, \qquad x_2(0) = 0{,}001$

$$h = 1$$



Rozwiązanie jest bardzo dobre, widać że krok jest zwiększany od środka spirali w stronę zewnętrza, ponieważ szybkość zmienności rozwiązania maleje.

Porównanie algorytmów

Przypadek	Parametr	RK4 ze stałym krokiem	PK4 Adams ze stałym krokiem	RK4 ze zmiennym krokiem	ode45
a) $x_1(0) = 7$ $x_2(0) = 8$	Krok optymalny (startowy)	0,007	0,008	0,06	-
	Liczba iteracji	2858	2501	123	149
	Czas obliczeń [ms]	34,92	47,16	6,95	0,95
b) $x_1(0) = 0$ $x_2(0) = 0{,}2$	Krok optymalny (startowy)	0,1	0,2	1	-
	Liczba iteracji	201	101	96	109
	Czas obliczeń [ms]	2,76	2,41	15,93	1,13
c) $x_1(0) = 6$ $x_2(0) = 0$	Krok optymalny (startowy)	0,021	0,0175	0,1	-
	Liczba iteracji	953	1143	119	149
	Czas obliczeń [ms]	12,91	24,72	7,39	1,05
d) $x_1(0) = 0{,}01$ $x_2(0) = 0{,}001$	Krok optymalny (startowy)	0,25	0,25	1	-
	Liczba iteracji	81	81	71	105
	Czas obliczeń [ms]	1,08	1,67	5,85	0,87

Wnioski

Wszystkie metody zostały porównane z referencyjną metodą ode45 wbudowaną w MATLABie. Pierwszym nasuwającym się wnioskiem jest to, że dla każdego z przypadków metoda ta jest lepsza niż pozostałe trzy.

Główną wadą metod ze stałym krokiem jest po pierwsze konieczność ręcznego doboru kroku, co implikuje konieczność uruchomienia metody przynajmniej kilkukrotnie w celu znalezienia optymalnego kroku. Jest to niewygodne z punktu widzenia użytkownika. Kolejną wadą metod ze stałym krokiem jest fakt, że krok ten jest dobierany globalnie do zwykle niewielkiego obszaru, który ma najszybszą zmienność rozwiązania i wymaga dużej liczby iteracji. Natomiast w pozostałych obszarach często wykonujemy zbędne iteracje, dla których optymalny krok mógłby być mniejszy. Metody te działają jednak dobrze dla przypadków takich jak b) czy d), gdzie zmienność funkcji jest globalnie podobna. W takich przypadkach metody te działają lepiej, ponieważ mają dużo mniejszą złożoność obliczeniową (szczególnie warta uwagi jest metoda RK4 ze stałym krokiem, która jest dużo prostsza implementacyjnie od pozostałych badanych metod oraz osiąga najlepsze czasy obliczeń w przeliczeniu na iterację). Metoda Predyktor-korektor wypada nieco gorzej pod względem implementacyjnym od metody RK4, ponieważ jest bardziej skomplikowana, a ponadto wymaga dostarczenia metody startowej, która będzie wyznaczać pierwsze kilka punktów (w zależności od rzędu metody), np. metody takiej jak RK4, dlatego też jest pod tym względem oceniona niżej.

Metoda RK4 ze zmiennym krokiem jest dużo wygodniejsza dla użytkownika. Dobór kroku początkowego nie jest tak istotny jak w przypadku pozostałych metod, po wybraniu dowolnego kroku mniejszego niż pewien krok graniczny, przy którym metoda nie znajdzie rozwiązania, w zasadzie nie ma wpływu na sposób działania metody, ponieważ krok zostanie dostosowany w pierwszych

iteracjach do zmienności rozwiązania. Podanie oczekiwanej dokładność rozwiązania w postaci dwóch parametrów $\varepsilon_w, \varepsilon_b$ jest dużo wygodniejsze z punktu widzenia użytkownika. Ponadto, ze względu na fakt, że wartość kroku nie jest dobierana globalnie, w obszarach małej szybkości zmienności rozwiązania metoda zwiększy krok co pozwoli na niewykonywanie niepotrzebnych iteracji. Metoda ta ma jednak dużo większą złożoność obliczeniową, w szczególności bardzo duży koszt czasowy pojedynczej iteracji, ponieważ dla nieodpowiedniej wartości kroku jest on zmieniany, a iteracja jest powtarzana do momentu uzyskania oczekiwanej dokładności w danym punkcie. W związku z tym mimo niewielkiej liczby iteracji, metoda ta działa najwolniej.

Jako wniosek ogólny można podać, że wybór najlepszej metody jest uwarunkowany kontekstem. Dla funkcji w miarę regularnych lepszym wyborem będzie metoda RK4 ze stałym krokiem, pod warunkiem że użytkownikowi nie przeszkadza uciążliwość doboru kroku, lub wykonuje wielokrotnie obliczenia dla jednego zagadnienia Cauchy'ego, wówczas ustalenie wartości kroku wystarczy wykonać raz. Natomiast dla funkcji bardziej nieregularnych, z obszarami o zróżnicowanych szybkościach wzrostu lepszym wyborem będzie metoda RK4 ze zmiennym krokiem.