# Projekt 4. [MNUM]

Bartosz Rajkowski, grupa 2AR, wtorek 16:00

4.12

# 1. Treść zadania:

Ruch punktu opisany jest równaniami: $x_1' = x_2 + x_1(0{,}9 - x_1^2 - x_2^2)$

$x_2' = -x_1 + x_2(0{,}9 - x_1^2 - x_2^2)$

Należy obliczyć przebieg trajektorii ruchu na przedziale [0, 20] dla następujących warunków początkowych:

a) $x_1(0)=10$ $x_2(0)=8$; b) $x_1(0)=0$ $x_2(0)=9$;
c) $x_1(0)=8$ $x_2(0)=0$; d) $x_1(0)=0{,}001$ $x_2(0)=0{,}001$.

Do rozwiązania zadania należy użyć zaimplementowanych przez siebie metod:

1. Rungego-Kutty czwartego rzędu (RK4) ze stałym krokiem. Wykonać tyle prób, ile będzie potrzebnych do znalezienia takiego kroku, którego zmniejszanie nie wpływa znacząco na rozwiązanie, podczas gdy zwiększanie – już wpływa;
2. Wielokrokowej predyktor-korektor Adamsa czwartego rzędu ze stałym krokiem, który należy dobrać w analogiczny sposób;
3. Rungego-Kutty czwartego rzędu (RK4) ze zmiennym krokiem. W każdym kroku należy szacować błąd aproksymacji.

# 2. Zastosowane algorytmy:

## 2.1. Metoda RK:

Metody Rungego-Kutty można zdefiniować wzorem:

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

Gdzie:

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

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

Przy czym:

$$\sum_{j=1}^{i-1} a_{ij} k_j = c_i, \qquad i = 2{,}3, \dots, m$$

Wartość $y_{n+1}$ zależy od $y_n$ oraz $h$.

Jeden krok metody wymaga obliczenia wartości prawych stron równań różniczkowych dokładnie m razy. Parametry $w_i, a_{ji}, c_i$ określa się tak, aby przy ustalonym m rząd metody był możliwie wysoki. Jeżeli przez $p(m)$ oznaczymy maksymalny możliwy do uzyskania rząd metody to:

$$p(m) = m\,, \qquad m = 1,2,3,4$$
$$p(m) = m - 1\,, \qquad m = 5,6,7$$
$$p(m) \leq m - 2\,, \qquad m \geq 8$$

Największe znaczenie mają metody z m=4 i rzędu 4. Jest to kompromis między dokładnością, a nakładem obliczeń na jedną iterację i związanym z tym wpływem błędów zaokrągleń.

### 2.2. Metoda RK4

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$
$$k_1 = f(x_n, y_n)$$
$$k_2 = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right)$$
$$k_3 = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2\right)$$
$$k_4 = f(x_n + h, y_n + hk_3)$$

### 2.3. Metoda predyktor-korektor Adamsa dla metody RK4

Metody typu *predyktor-korektor* są połączeniem metod jawnych i niejawnych w jeden algorytm. Dla metody k-krokowej realizacja w postaci struktury predyktor-korektor ma postać:

$$y_n^{[0]} = \sum_{j=1}^{k} \alpha_i y_{n-j} + h\sum_{j=1}^{k} \beta_i f_{n-j} \qquad \text{predykcja}$$
$$f_n^{[0]} = f\left(x_n, y_n^{[0]}\right) \qquad \text{ewaluacja}$$
$$y_n = \sum_{j=1}^{k} \alpha_j^* y_{n-j} + h\sum_{j=1}^{k} \beta_j^*\, f_{n-j} + h\beta_0^*\, f_n^{[0]} \qquad \text{korekcja}$$
$$f_n = f(x_n, y_n) \qquad \text{ewaluacja}$$

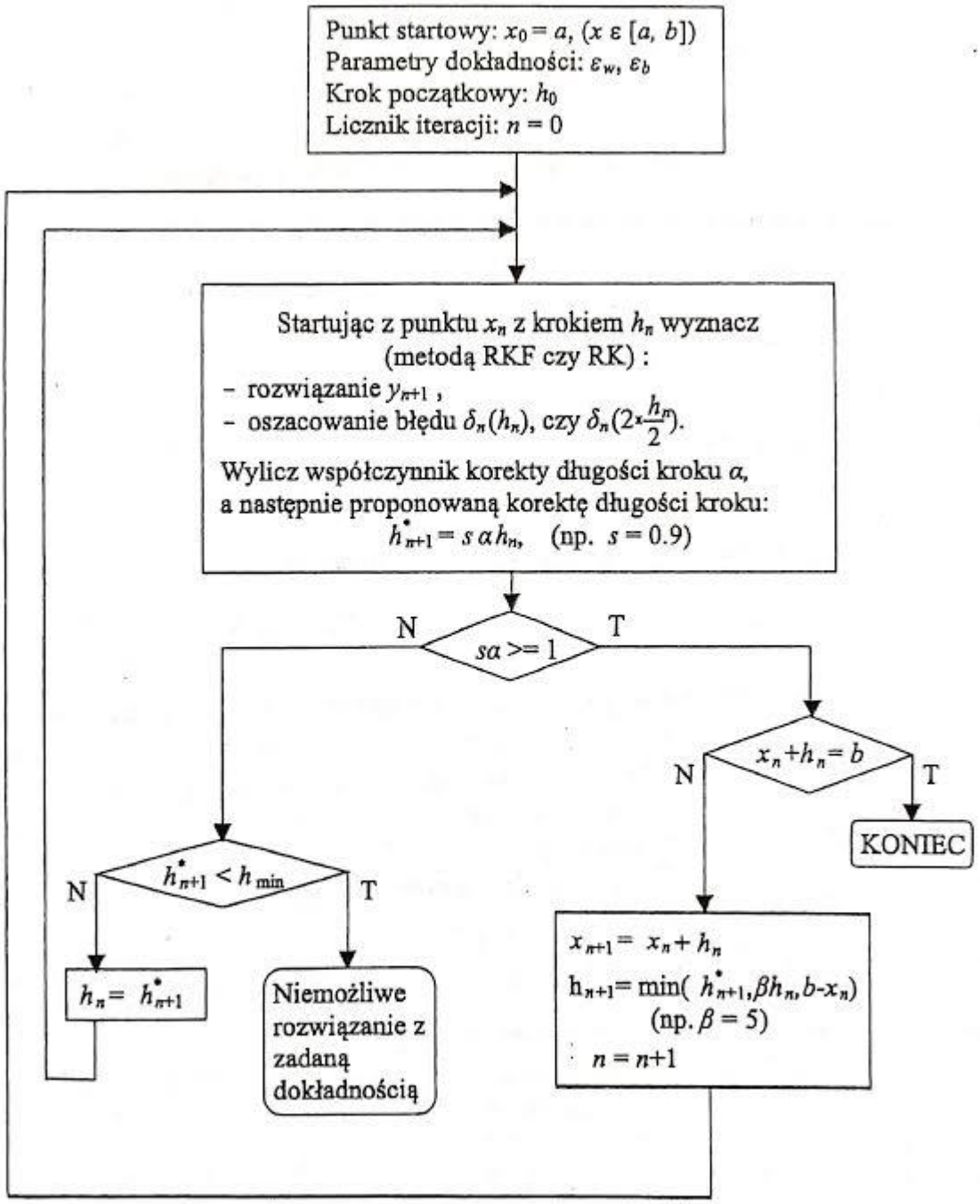
Dla metod Adamsa algorytm przyjmuje postać:

$$y_n^{[0]} = y_{n-1} + h\sum_{j=1}^{k} \beta_i f_{n-j}$$
$$f_n^{[0]} = f\left(x_n, y_n^{[0]}\right)$$

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

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

## 2.4. Metoda Rungego-Kutty czwartego rzędu ze zmiennym krokiem:




Rys. 7.6. Schemat blokowy podstawowej realizacji metod RK i RKF

Stosuję szacowanie błędu wg zasady podwójnego kroku. Zatem współczynniki przyjmują wartości:

$$\delta_n(h)_i = \frac{(y_i)_n^{(2)} - (y_i)_n^{(1)}}{2^p - 1}\ , i = 1{,}2$$

$$\varepsilon_i = \left|(y_i)_n^{(2)}\right| * \varepsilon_n + \varepsilon_b$$

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

Dla naszego przypadku $p = 4$

## 3. Kod programu:

### 3.1. RK4:

```
function [Y,time] = rk4( x, lenght, step )
%Metoda RK4
time = zeros(ceil(lenght/step),1); %wektor chwil czasu
Y = zeros(ceil(lenght/step),2);
 iteracje=0;
for i=1:ceil(lenght/step)
    time(i)=i*step; %zapisanie czasu po kojenym kroku
    k1 = newx(x); %pochodna w punkcie y(xn)
    k2 = newx(x+(step/2)*k1); %pochodna w punkcie y(xn+step/2*k1)
    k3 = newx(x+(step/2)*k2); %pochodna w punkcie y(xn+step/2*k2)
    k4 = newx(x+step*k3); %pochodna w punkcie y(xn+step*k3)
    x=x+(1/6)*step*(k1+2*k2+2*k3+k4); %obliczenie następnego punktu
    Y(i,:) = x;
%     iteracje=iteracje+1;
end
%  plot(time,Y);
plot3(time,Y(:,1),Y(:,2));
 iteracje
end
```

### 3.2. Predyktor-korektor:

```
function [Y,time, Err] = predyktor_korektor( x, lenght, step )
%predyktor-korektor

halfstep=step/2;
time=zeros(ceil(lenght/step),1);
Y=zeros(ceil(lenght/step),2);
Err=zeros(ceil(lenght/step),2);
iteracje=3;
    for i = 1:3
        time(i) = i*step; %zapisanie czasu próbki
        k1 = newx(x); %pochodna w punkcie y(xn)
        %display (k1);
        k2 = newx(x+halfstep*k1); %pochodna w punkcie y(xn+step/2*k1)
        k3 = newx(x+halfstep*k2); %pochodna w punkcie y(xn+step/2*k2)
        k4 = newx(x+step*k3); %pochodna w punkcie y(xn+step*k3)
        x=x+(1/6)*step*(k1+2*k2+2*k3+k4); %obliczenie następnego punktu
        Y(i,:) = x; %zapisanie punktu do wektora
    end
    for i = 4:ceil(lenght/step)
        time(i) = i*step; %zapisanie czasu próbki
        % predykcja (z ewaluacja)
        temp = x + step/24*(55*newx(x) - 59*newx(Y(i-1,:))+ 37*newx(Y(i-2,:)) -
9*newx(Y(i-3,:)));
        % korekcja (z ewaluacja)
        x = x + step/24*(9*newx(temp) + 19*newx(x) - 5*newx(Y(i-1,:)) + newx(Y(i-
2,:)));
        Y(i,:) = x; %zapis wyniku
        iteracje=iteracje+1;
    end
        iteracje
%      plot(time,Y);
%      plot3(time,Y(:,1),Y(:,2));
end
```

### 3.3. RK4 ze zmiennym krokiem:

```
function [ time,Y] = zmienny_krok(x, length, step )
%Metoda RK4 z szacowaniem błędu i zmianą kroku
    time=[0];
    Y=x;
    en=0.000001;
    eb=0.000001;
    s=0.9;
    stepmin=10^-10;
    beta=5;%maksymalny współczynnik zmiany kroku
    n=1;
    iteracje=0;
    tic;
    while time(end)<=length
        iteracje=iteracje+1;
        x1=Y(n,:);
        x2=Y(n,:);
        k1 = newx(x1); %pochodna w punkcie y(xn)
        k2 = newx(x1+(step/2)*k1); %pochodna w punkcie y(xn+step/2*k1)
        k3 = newx(x1+(step/2)*k2); %pochodna w punkcie y(xn+step/2*k2)
        k4 = newx(x1+step*k3); %pochodna w punkcie y(xn+step*k3)
        x1=x1+(1/6)*step*(k1+2*k2+2*k3+k4); %obliczenie następnego punktu
        step2=step/2; %pol kroku
        for i=1:2
            k1 = newx(x2); %pochodna w punkcie y(xn)
            k2 = newx(x2+(step2/2)*k1); %pochodna w punkcie y(xn+step/2*k1)
            k3 = newx(x2+(step2/2)*k2); %pochodna w punkcie y(xn+step/2*k2)
            k4 = newx(x2+step2*k3); %pochodna w punkcie y(xn+step*k3)
            x2=x2+(1/6)*step2*(k1+2*k2+2*k3+k4); %obliczenie następnego punktu
        end
       d1=(x2(1)-x1(1))/15;%obliczanie współczynników
       d2=(x2(2)-x1(2))/15;
       e1=abs(x2(1))*en+eb;
       e2=abs(x2(2))*en+eb;
       alfa=min((e1/abs(d1))^(1/5),(e2/abs(d2))^(1/5));%współczynnik zmiany kroku
       nextstep=s*alfa*step;%wyznaczenie kolejnego kroku zgodnie z algorytmem
       Y(n+1,1:2)=x1;%zapis kolejnego punktu
       if(s*alfa<1)
           if(nextstep<stepmin)
              error('Niemożliwe rozwiązanie z zadaną dokładnością');
           else
              step=nextstep;
           end
       else
           if(time(end)+step==length)%koniec petli
              break;
           else
               time=[time; time(end)+step]; %zapis kolejnego punktu w czasie
               step=min([nextstep,beta*step,length-time(end)]);%wybor kolejnego
kroku
               n=n+1;%przejscie do kolejnej iteracji
           end

       end
    end
    time=[time; length];
    iteracje
%     plot(time,Y);
end
```

### 3.4. Obliczanie pochodnych:

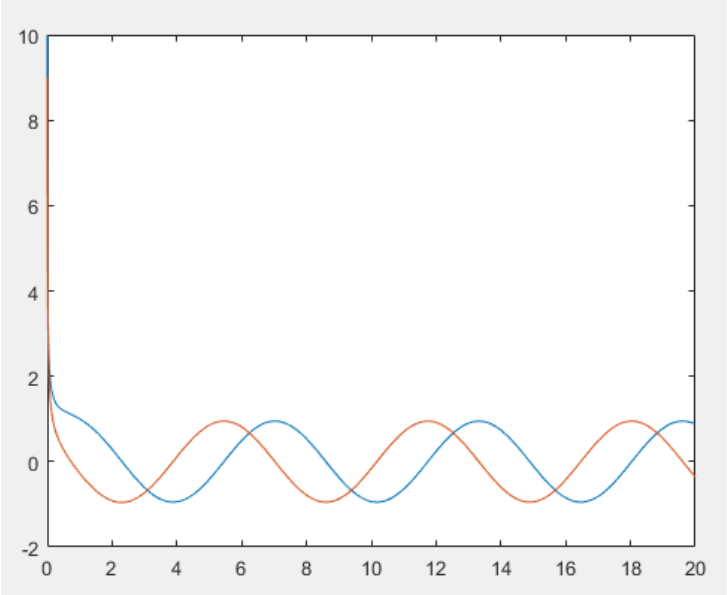
```
function [ x2 ] = newx(x)
%funkcja obliczająca pochodne wg wzoru
    x2(1)=x(2)+x(1)*(0.9-x(1)^2-x(2)^2);
    x2(2)=-x(1)+x(2)*(0.9-x(1)^2-x(2)^2);

end
```

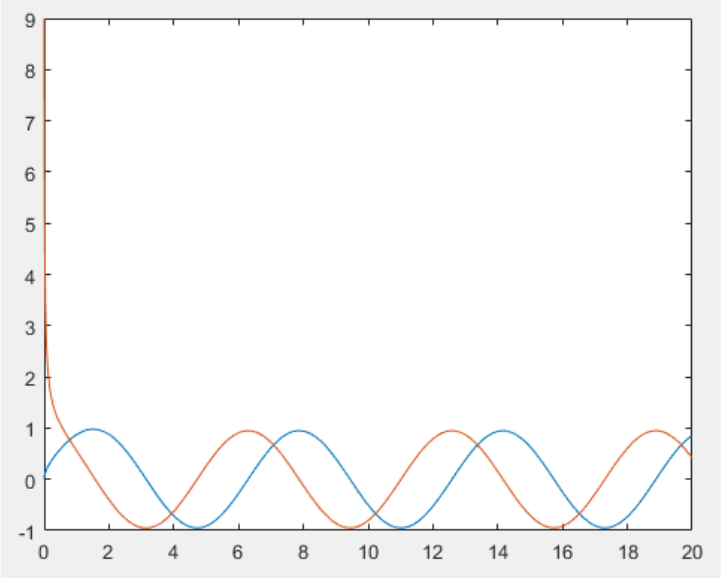
### 3.5. Funkcja do ode45

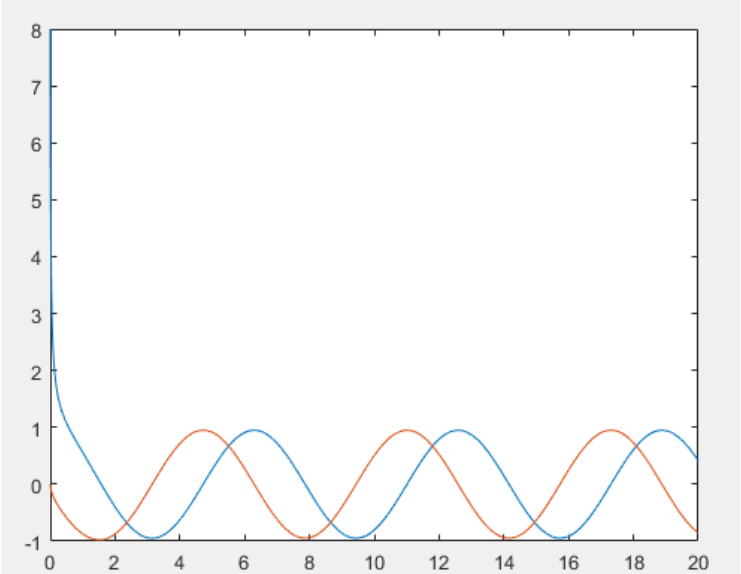
```
function [ Y ] = f( t,x )
A=newx(x);
Y = [A(1); A(2)];
end
```

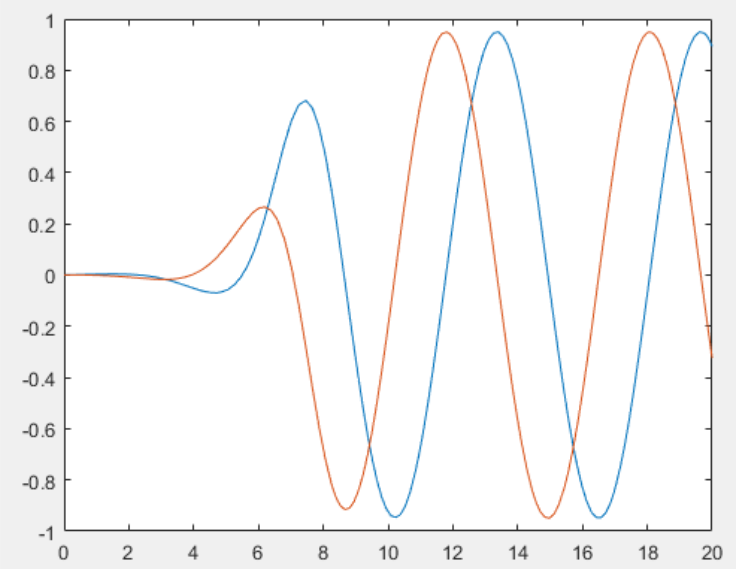
## 4. Wyniki:

### 4.1. Wyniki wyznaczone poleceniem ode45 dla kolejnych zestawów:



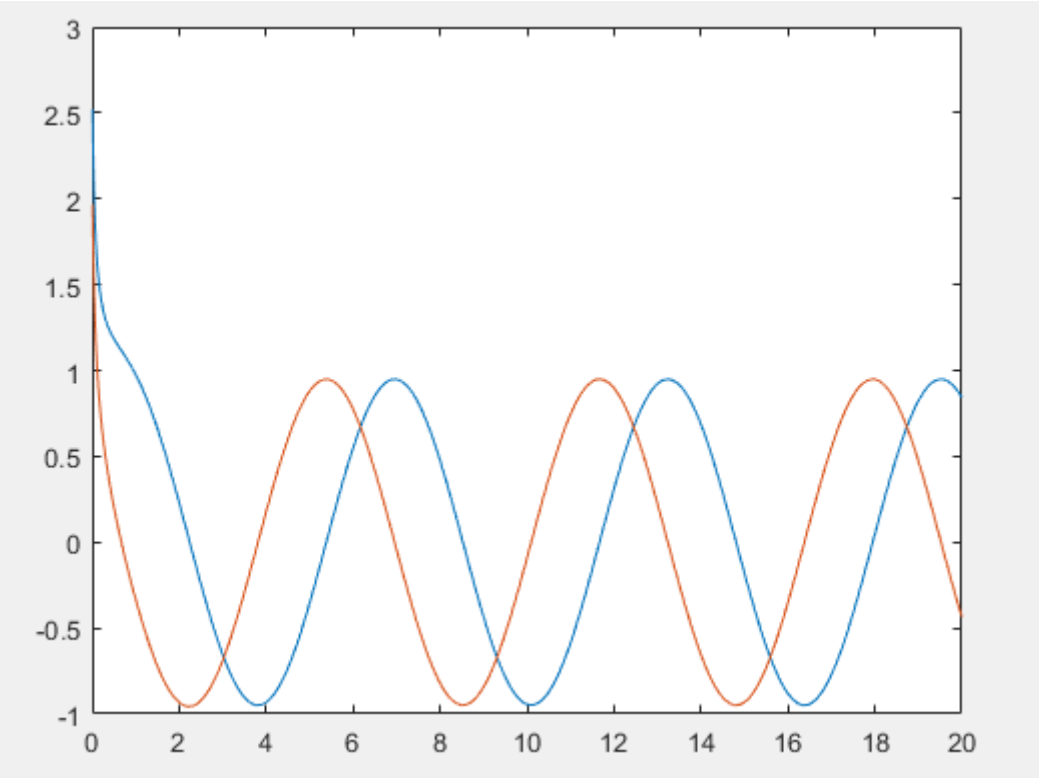



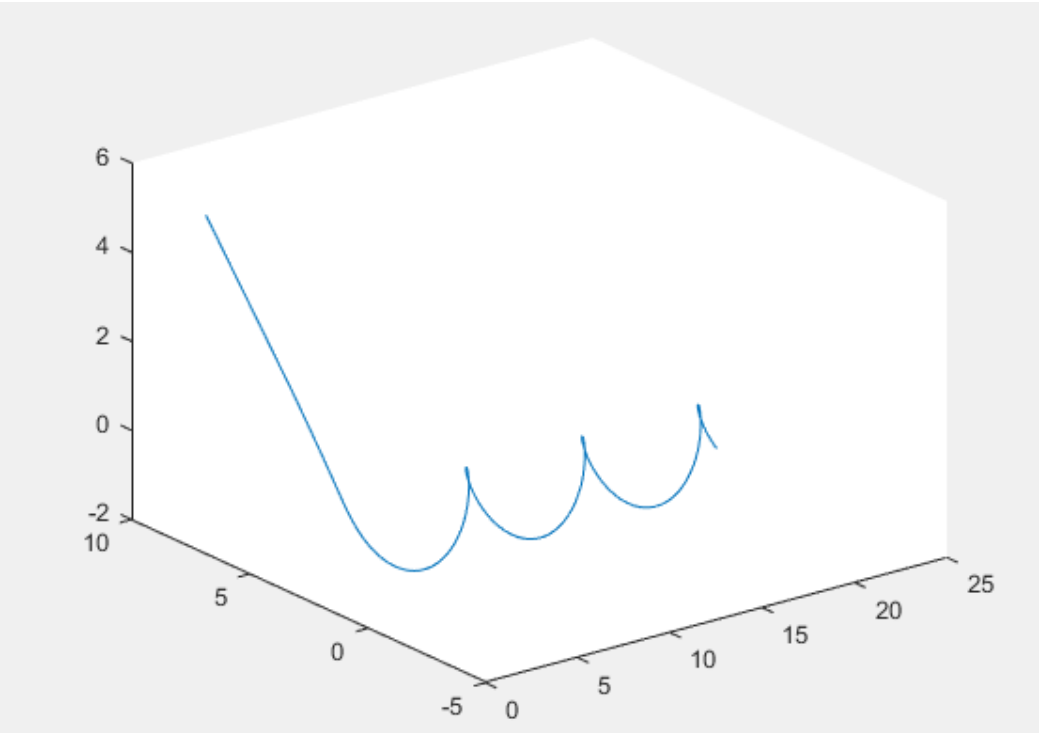



## 4.2. RK4 ze stałym krokiem

a) $x_1(0)=10$ $x_2(0)=8$

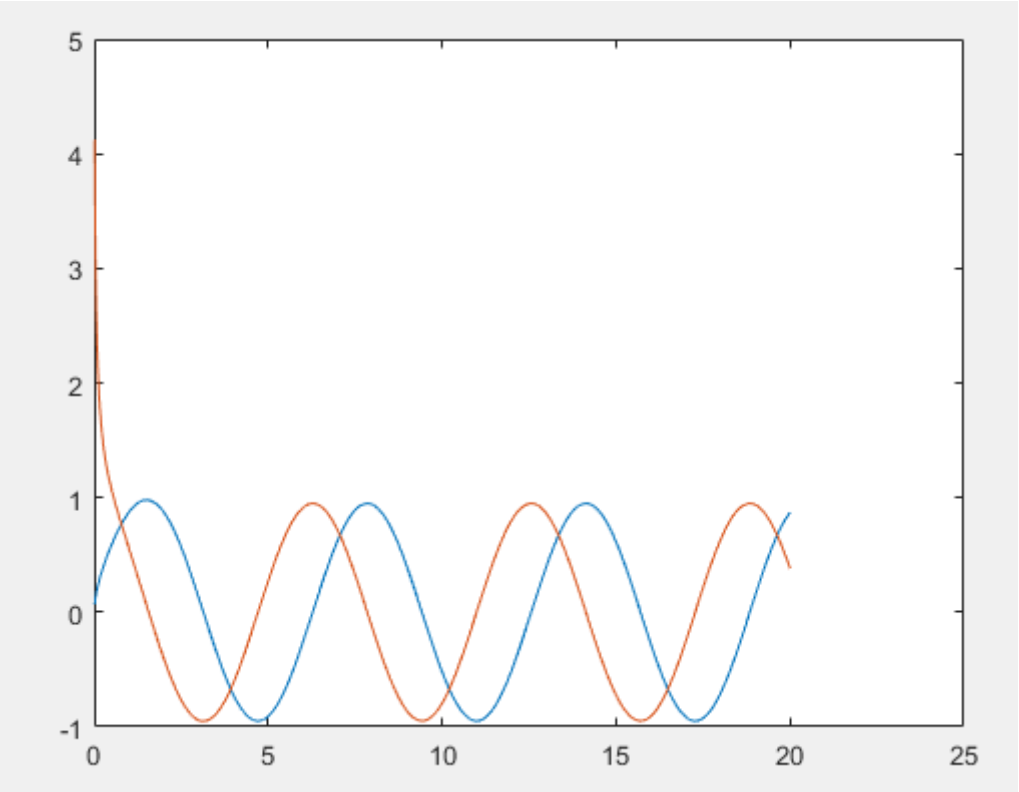
Metoda jest zbieżna dla kroku 0.014 lub mniejszego. Zmniejszanie kroku nie wpływa znacząco na przebieg funkcji.

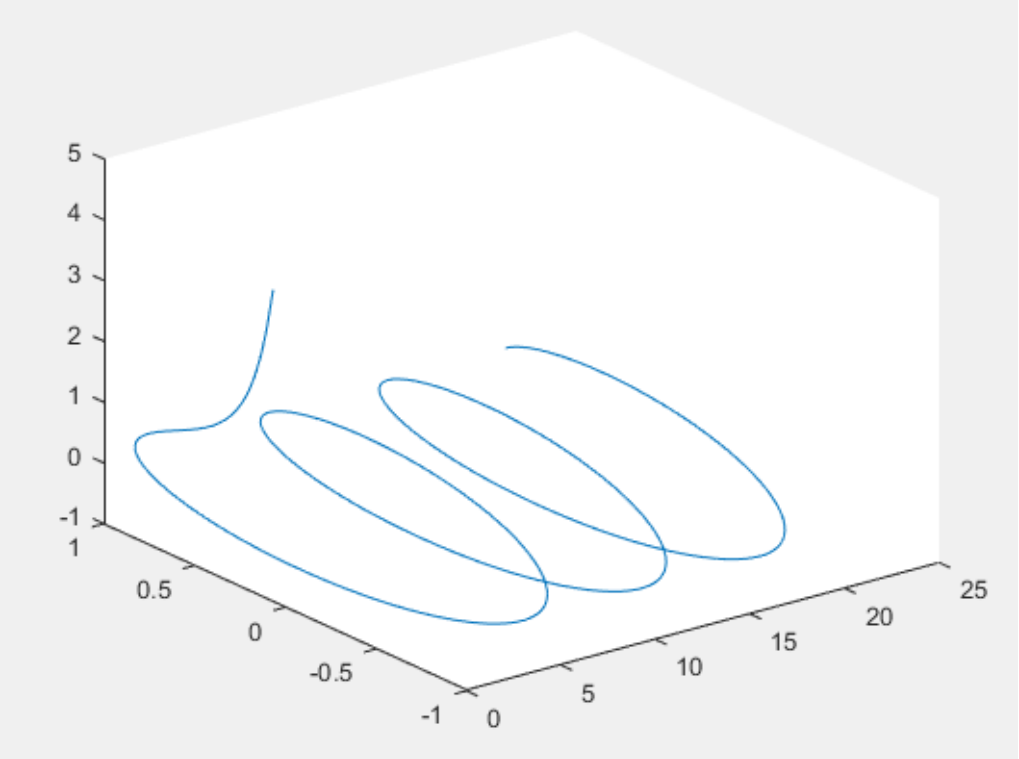


b) $x_1(0)=0$ $x_2(0)=9$

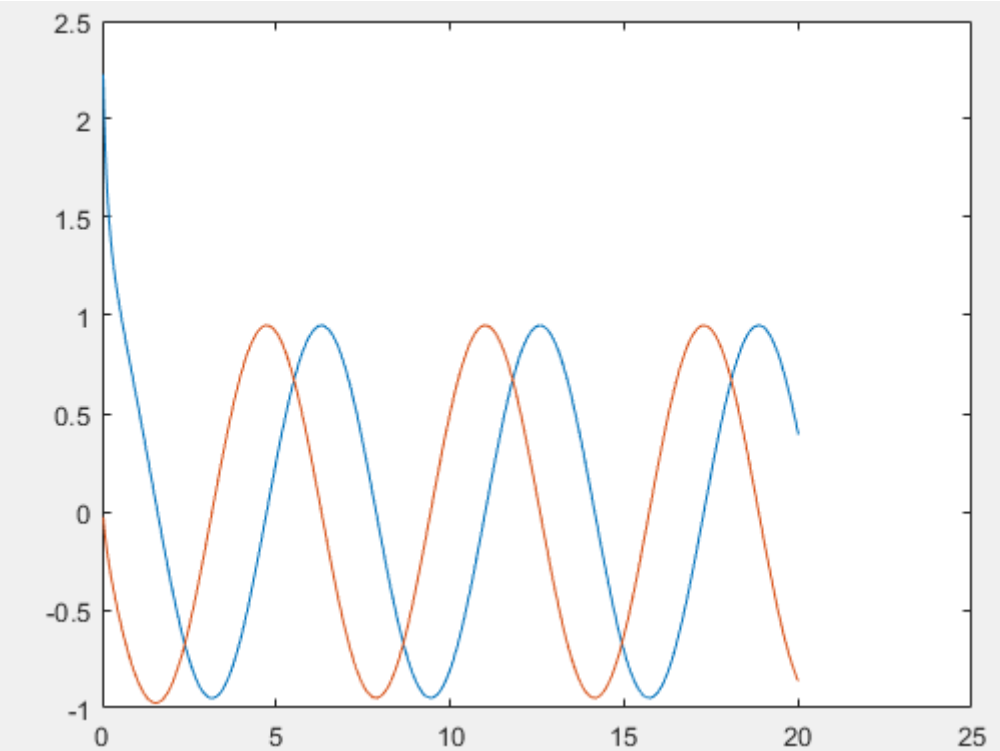
Metoda jest zbieżna dla kroku 0.03 lub mniejszego. Zmniejszanie kroku nie wpływa znacząco na przebieg funkcji.

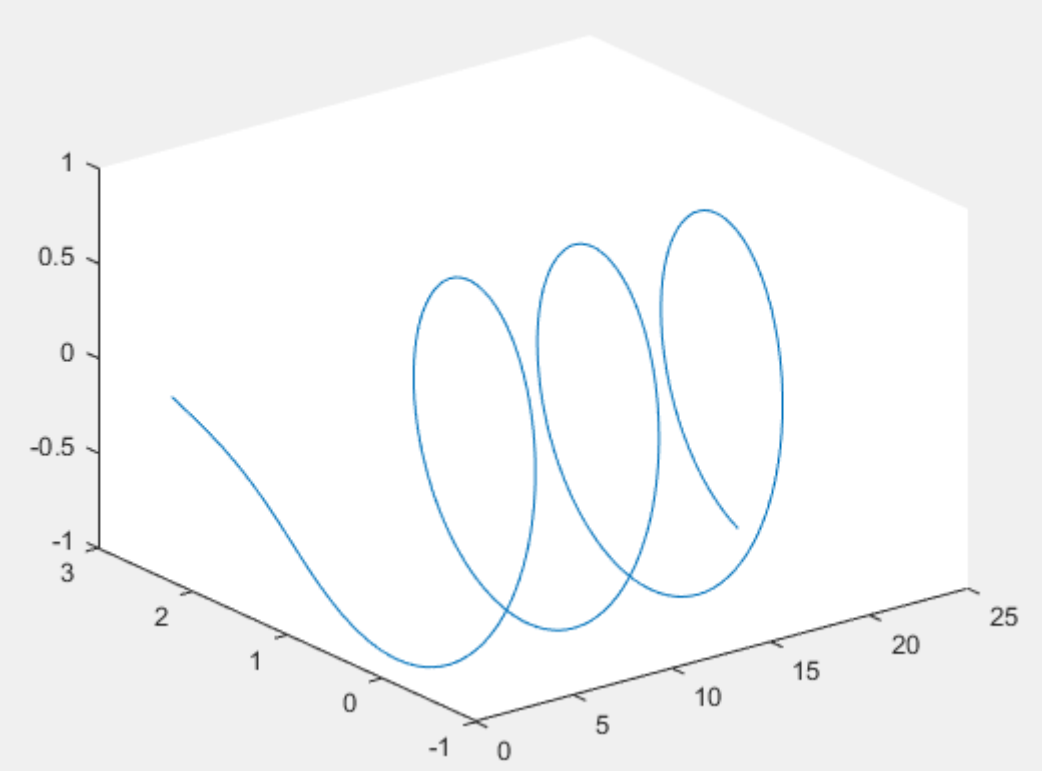


c) $x_1(0)=8$ $x_2(0)=0$

Metoda jest zbieżna dla kroku 0.038 lub mniejszego. Zmniejszanie kroku nie wpływa znacząco na przebieg funkcji – jedyne różnice widoczne gołym okiem na wykresie są w pobliżu 0 (początkowe wartości są większe).



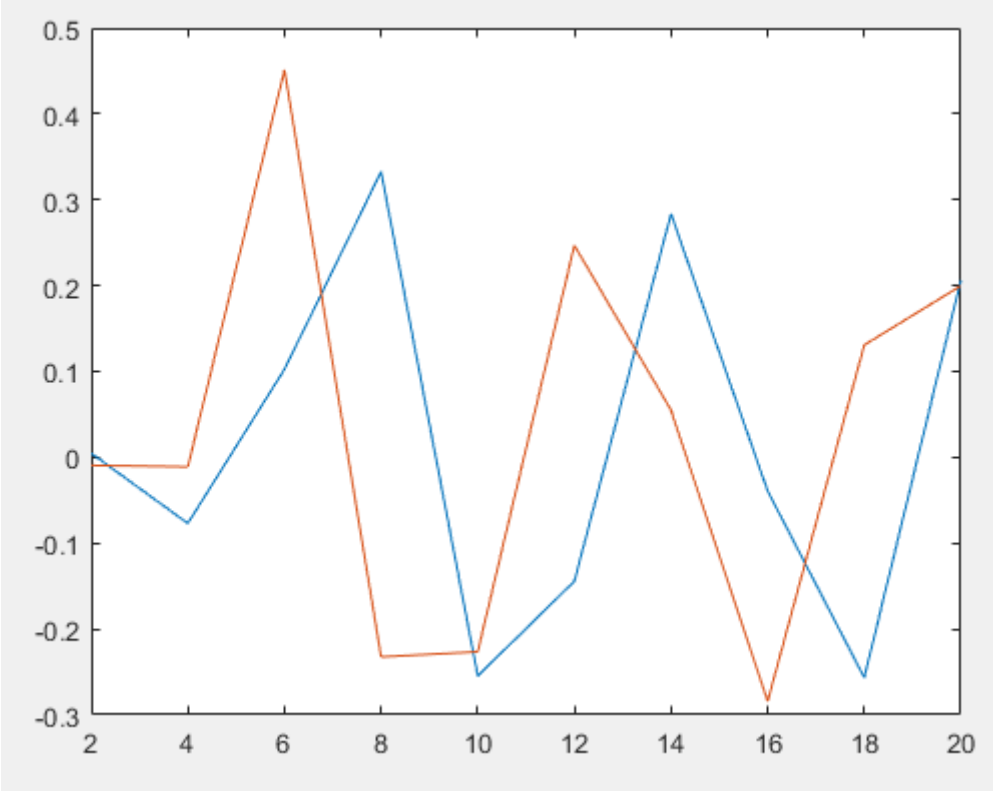



**d)** $x_1(0)=0{,}001$ $x_2(0)=0{,}001$

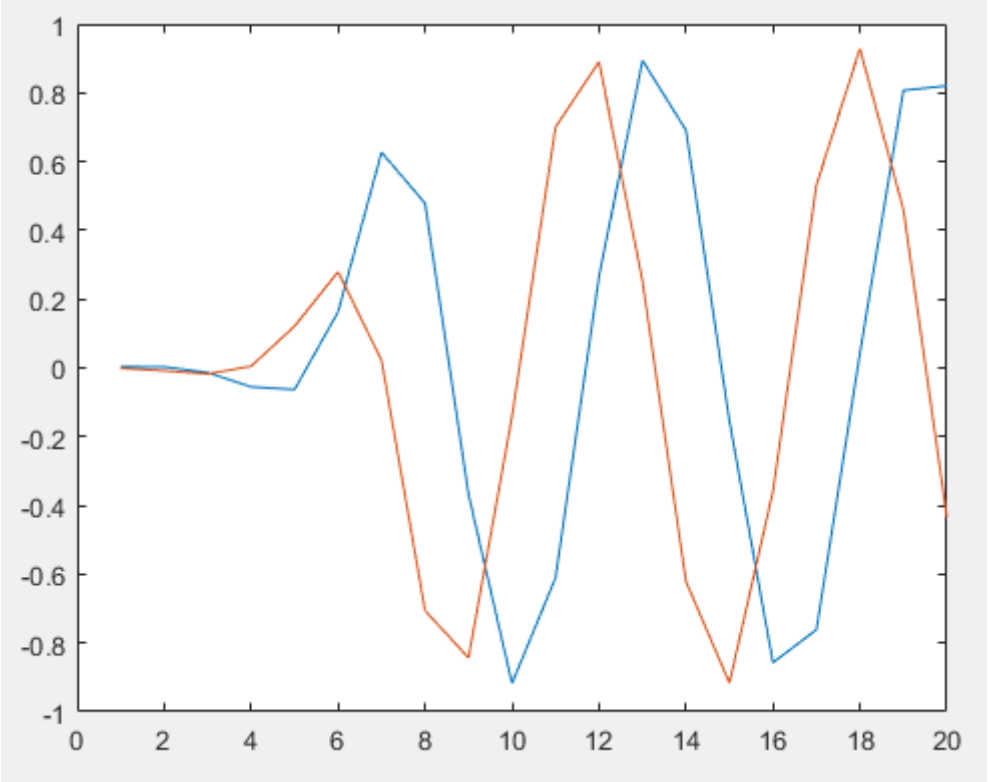
Metoda jest rozbieżna dla kroku równego 2.3 lub więcej.

Dla tych punktów startowych zmniejszanie kroku znacząco wpływa na wygląd wykresu:

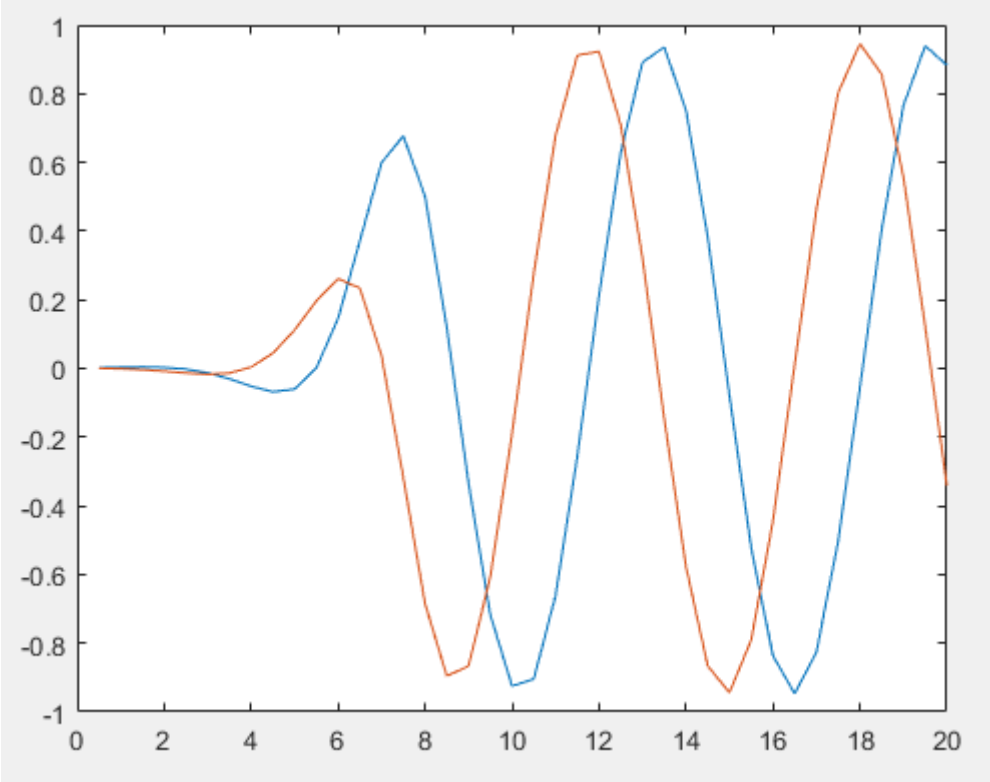
**a. dla kroku 2:**



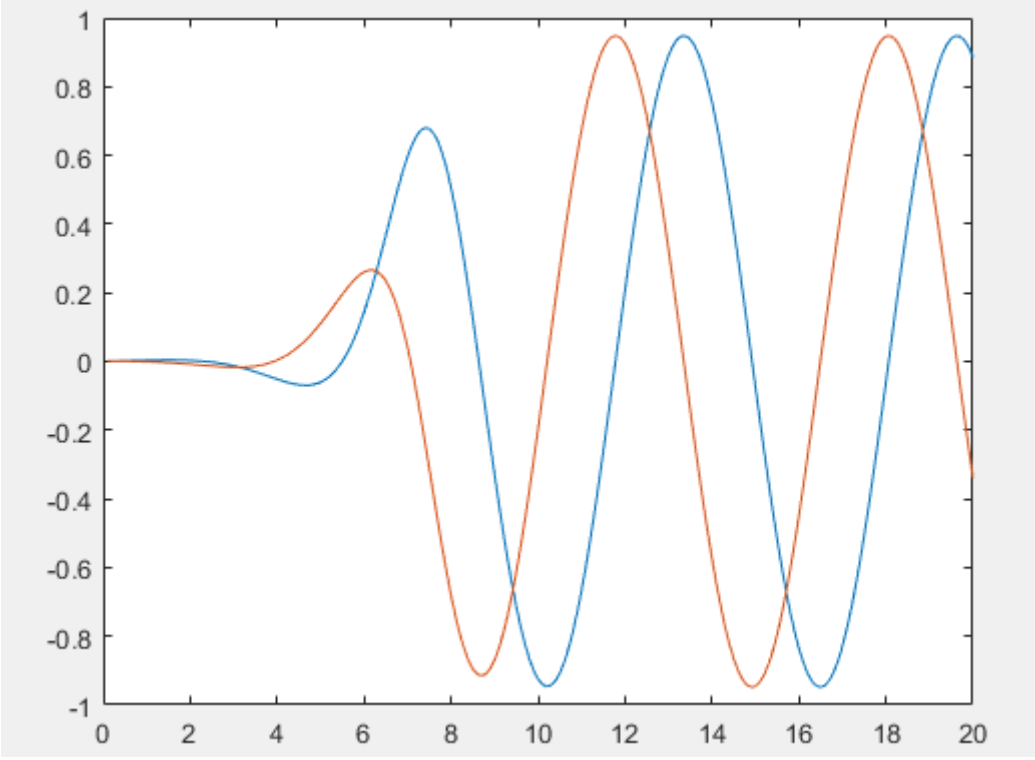

**b. dla kroku 1:**

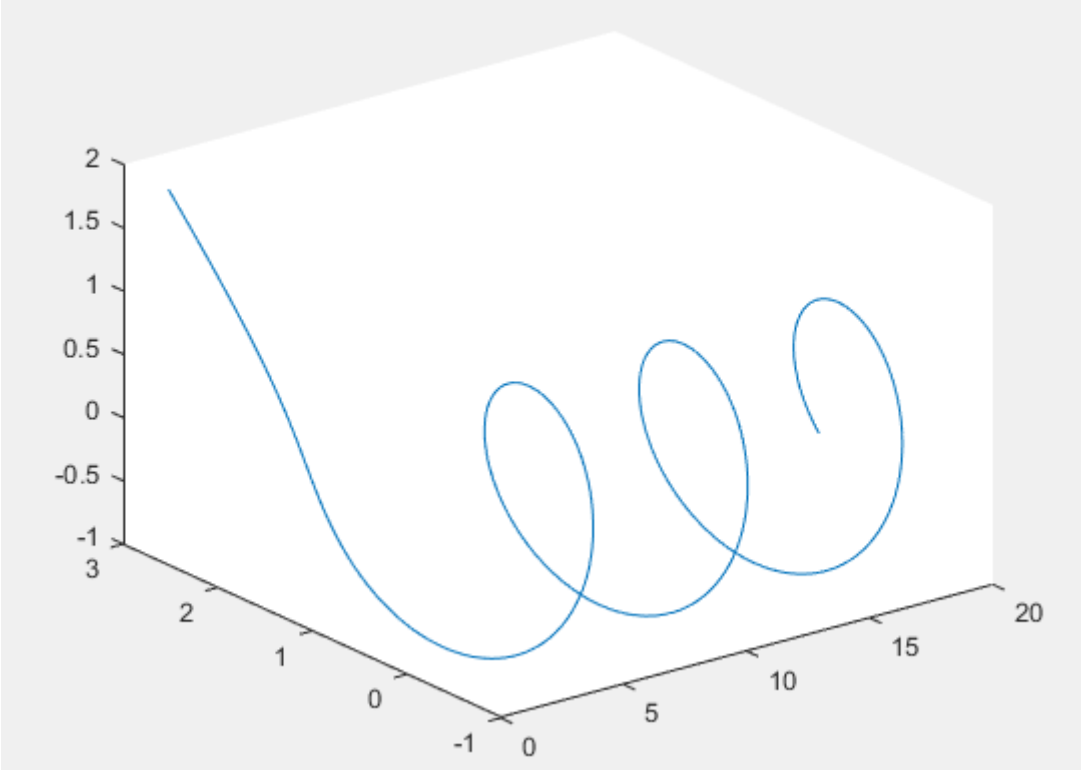


c. **dla kroku 0.5:**




d. **dla kroku 0.1:**





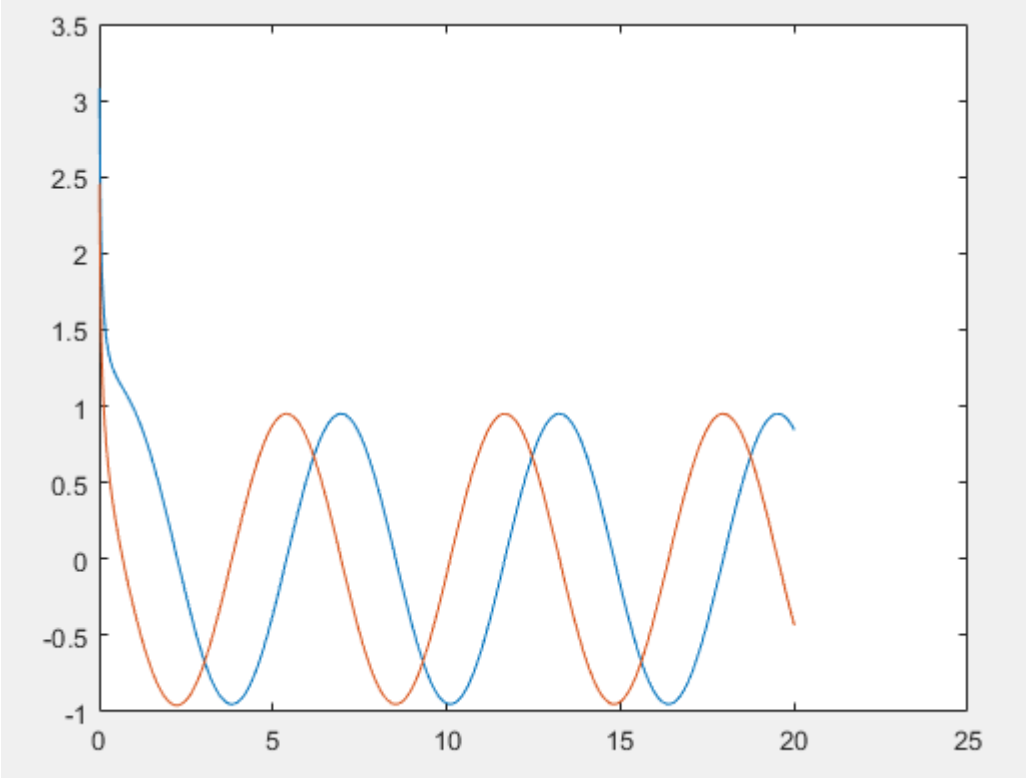

Dalsze zmniejszanie kroku nie wpływa w widoczny sposób na przebieg funkcji.

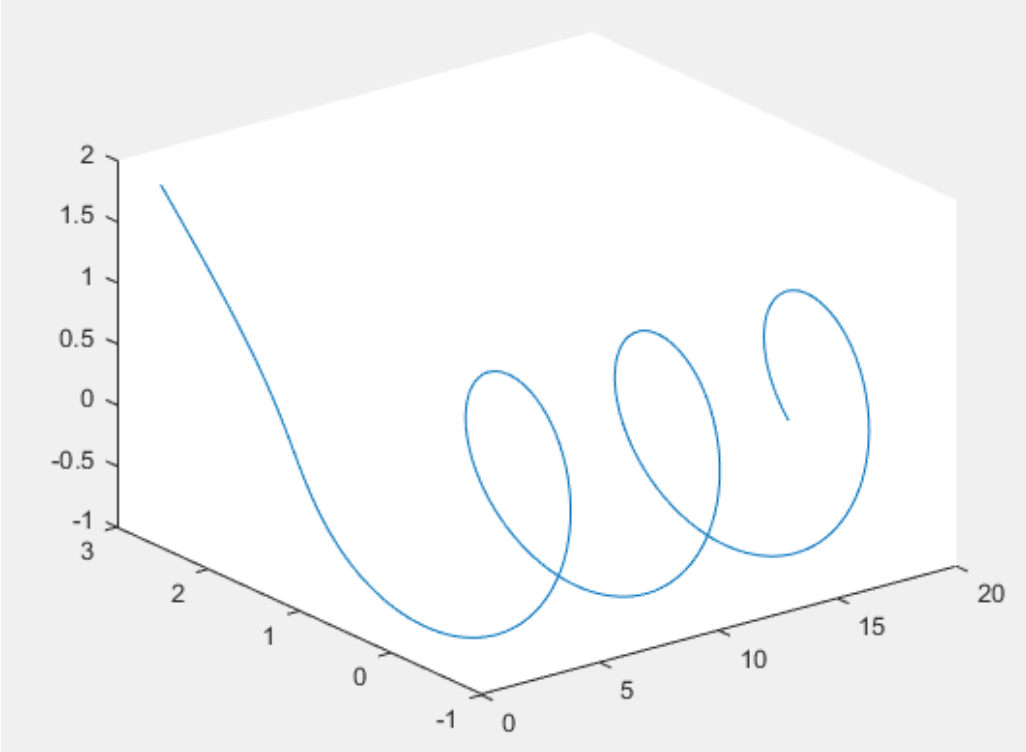
### 4.3. Metoda predyktor-korektor:

**a)** $x_1(0)=10$ $x_2(0)=8$

Metoda jest zbieżna dla kroku 0.011 lub mniejszego.

Zmniejszanie kroku nie wpływa znacząco na przebieg funkcji.

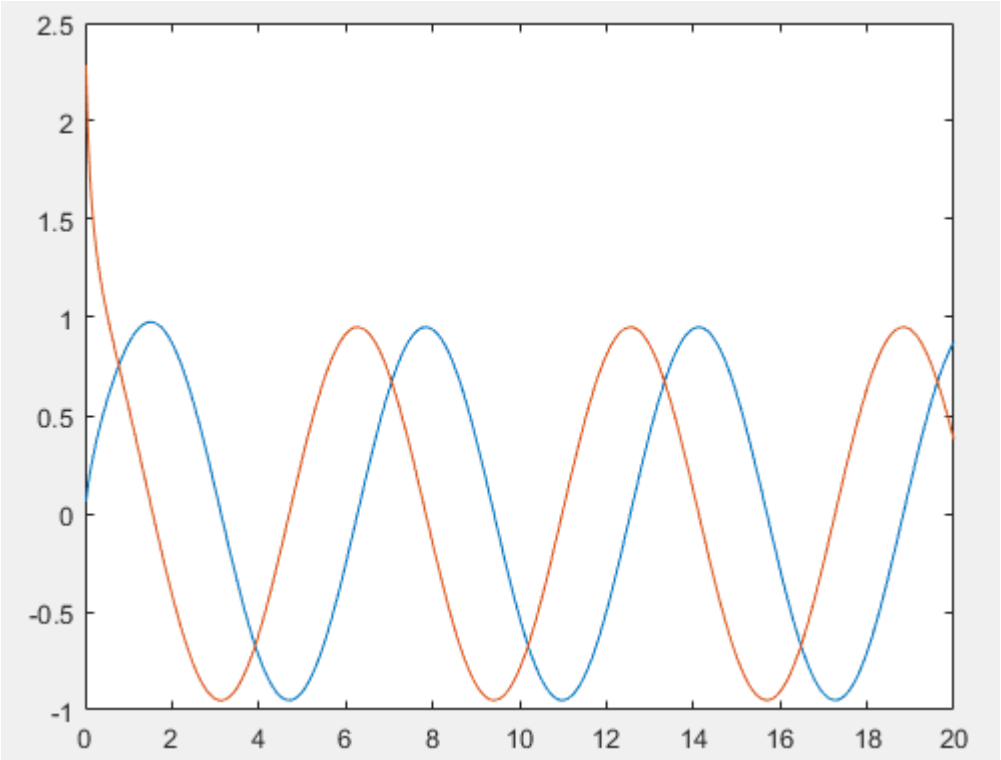


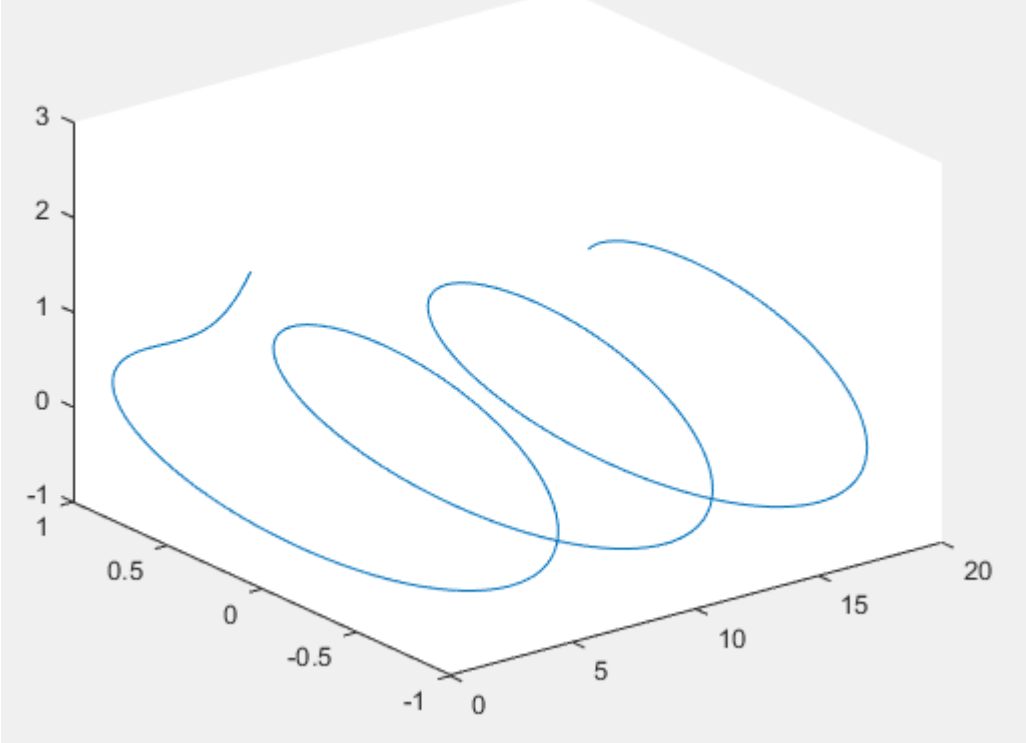



**b)** $x_1(0)=0$ $x_2(0)=9$

Metoda jest rozbieżna dla kroku 0.03 lub większego.

Zmniejszanie kroku nie wpływa znacząco na przebieg funkcji.

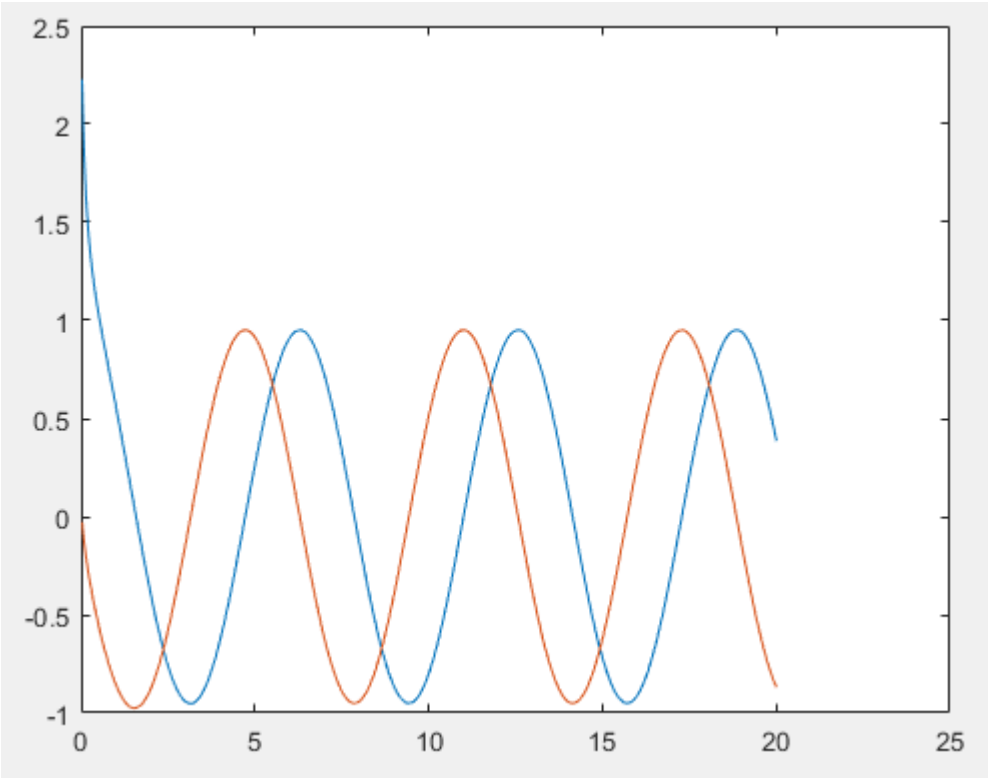


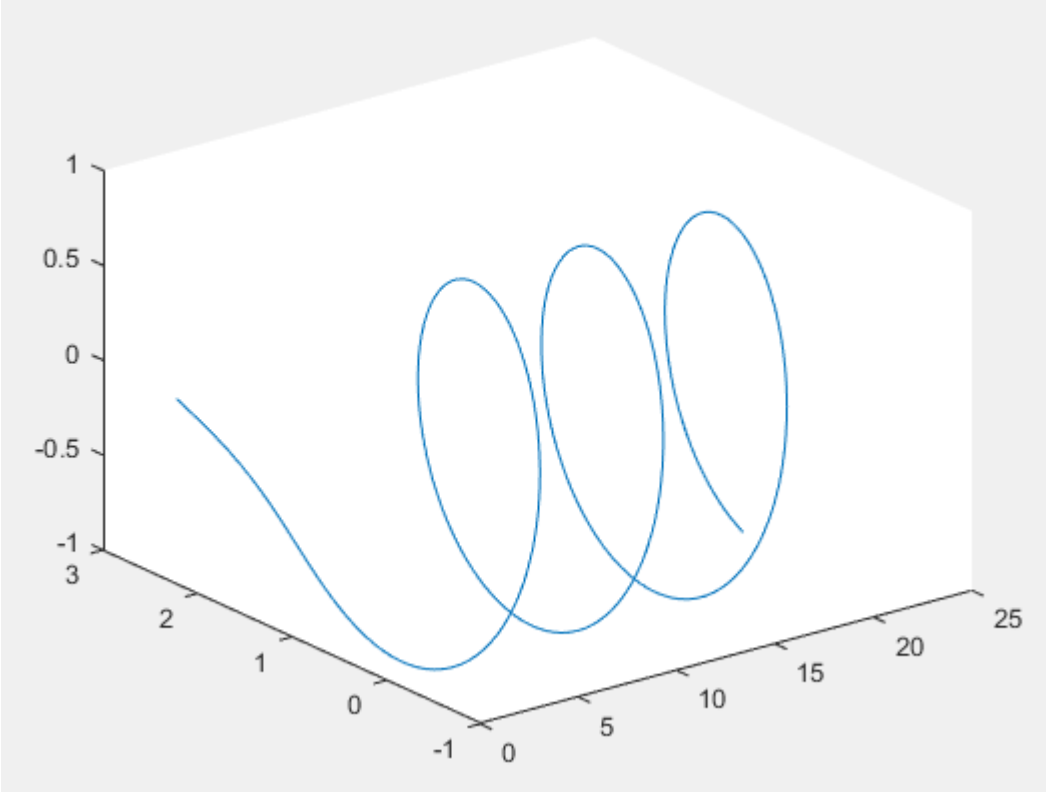



**c)** $x_1(0)=8$ $x_2(0)=0$

Metoda jest rozbieżna dla kroku 0.04 lub większego.

Zmniejszanie kroku nie wpływa znacząco na przebieg funkcji.



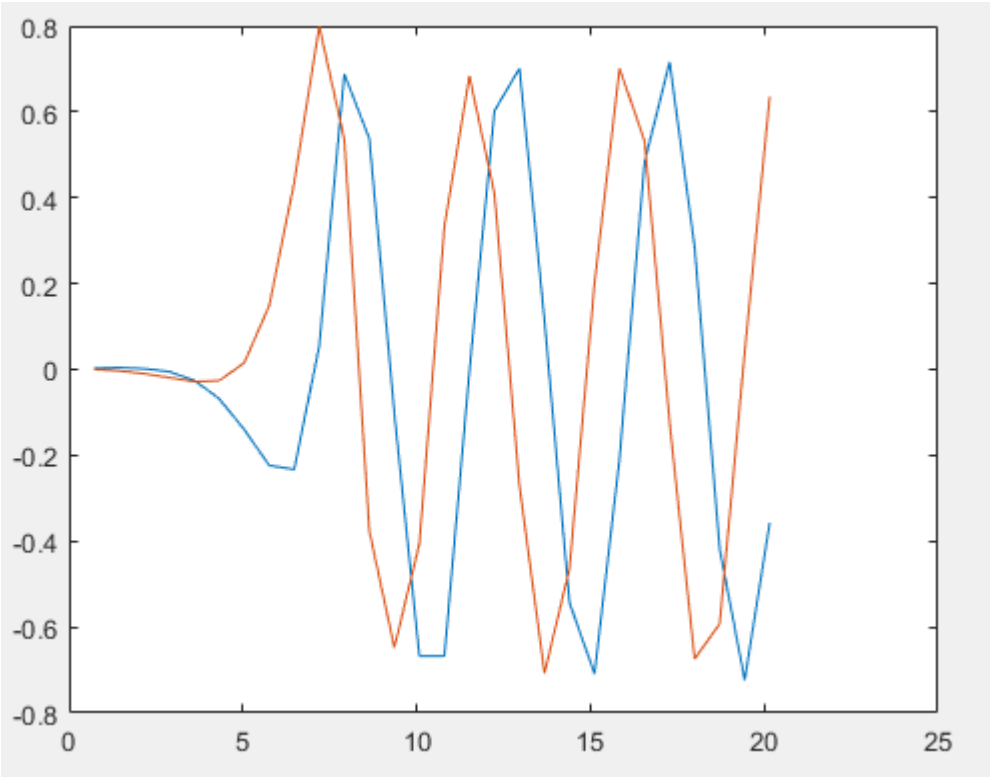



**d)** $x_1(0)=0{,}001$ $x_2(0)=0{,}001$

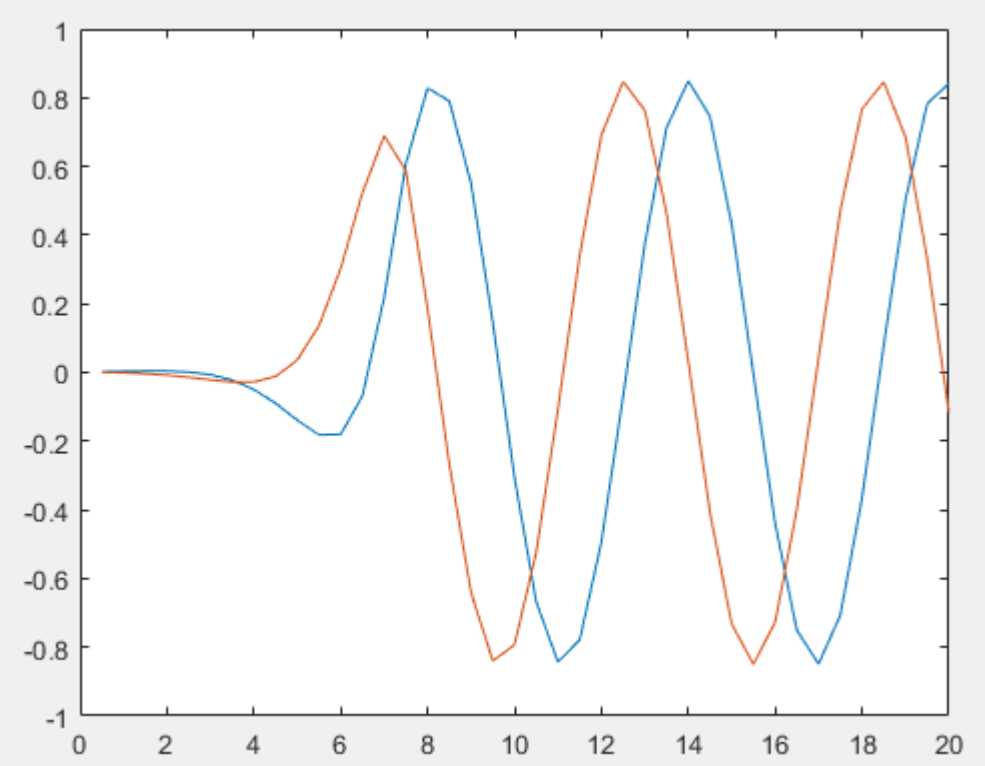
Metoda jest zbieżna dla kroku 0.72 lub mniejszego.

Zmniejszanie kroku wpływa na przebieg funkcji.

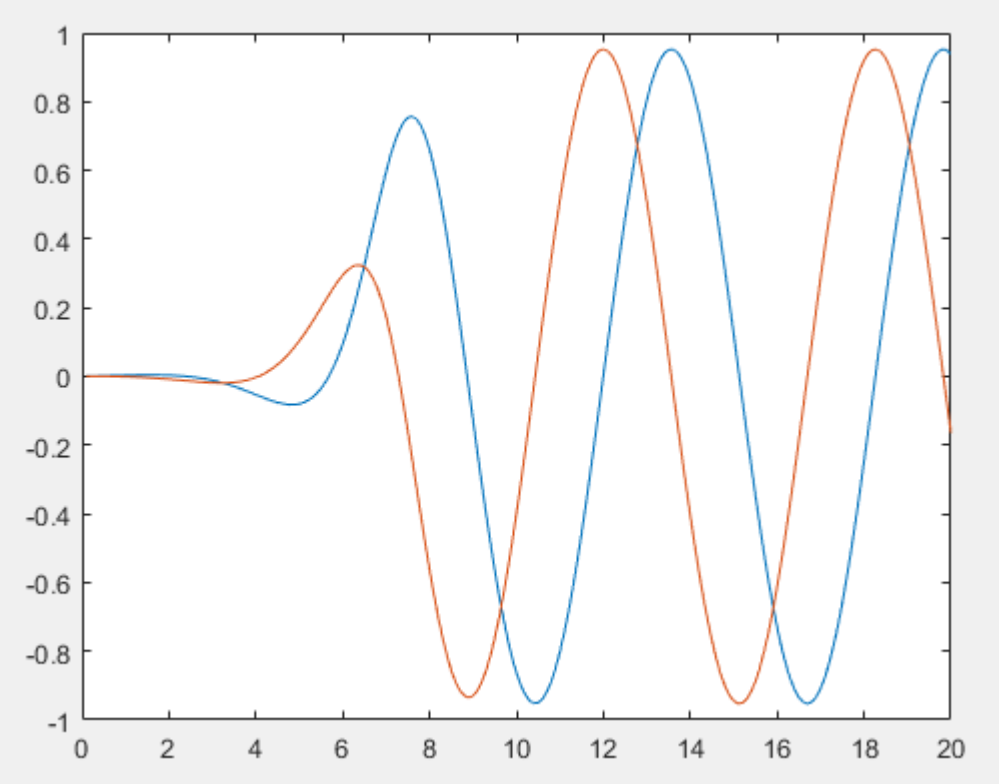
**a) dla 0.072:**



**b) dla 0.5**



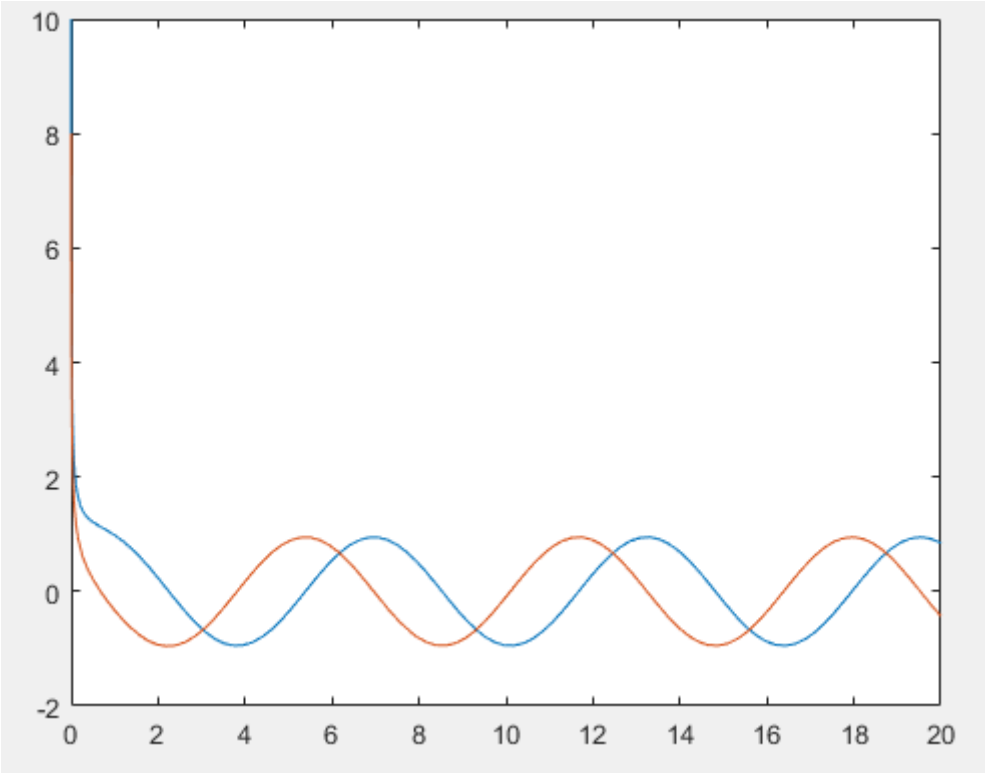

**c) dla 0.1**



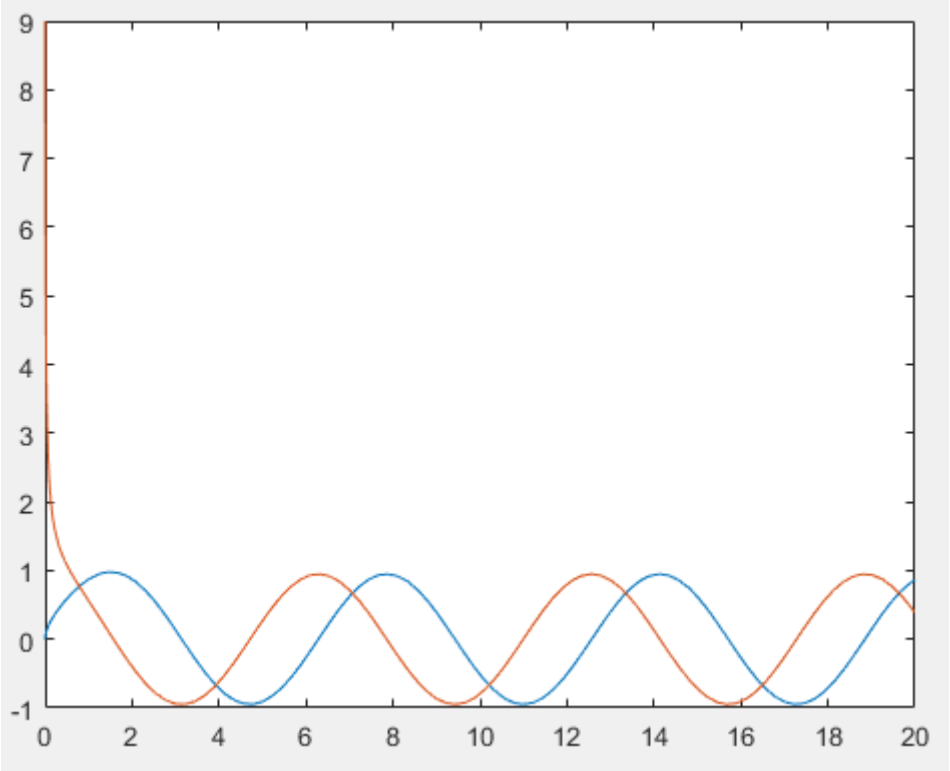

**Dalsze zmniejszanie kroku nie wpływa na wygląd wykresu.**

### 4.4. RK4 ze zmiennym krokiem:

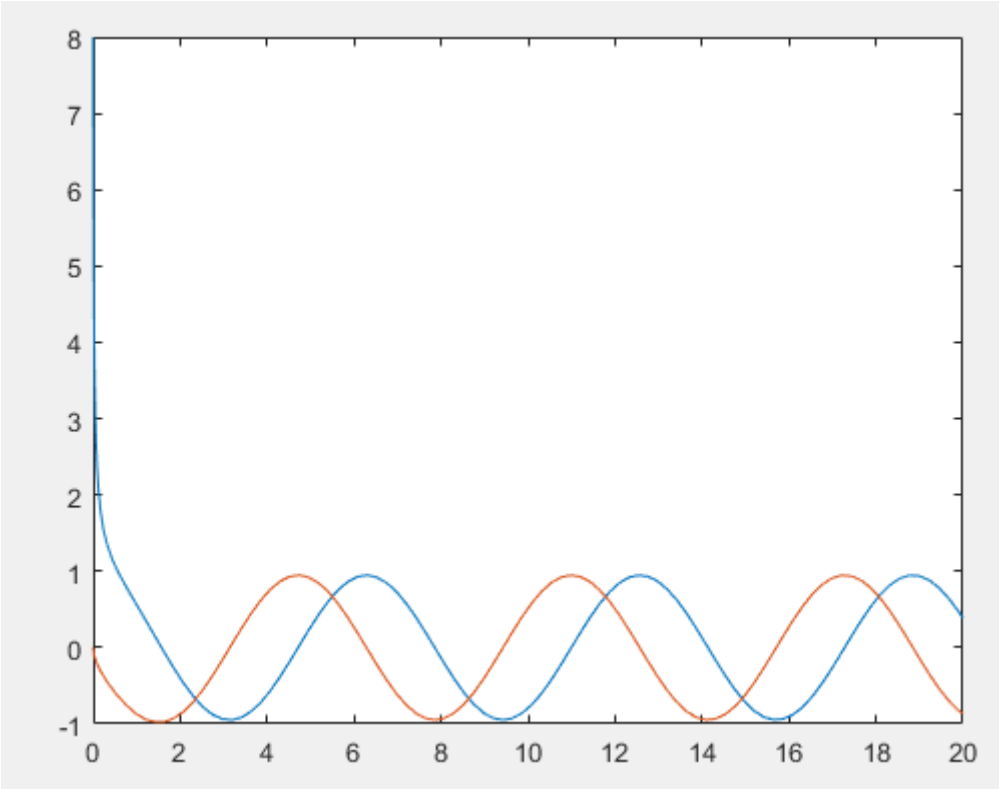
**a)** $x_1(0)=10$ $x_2(0)=8$



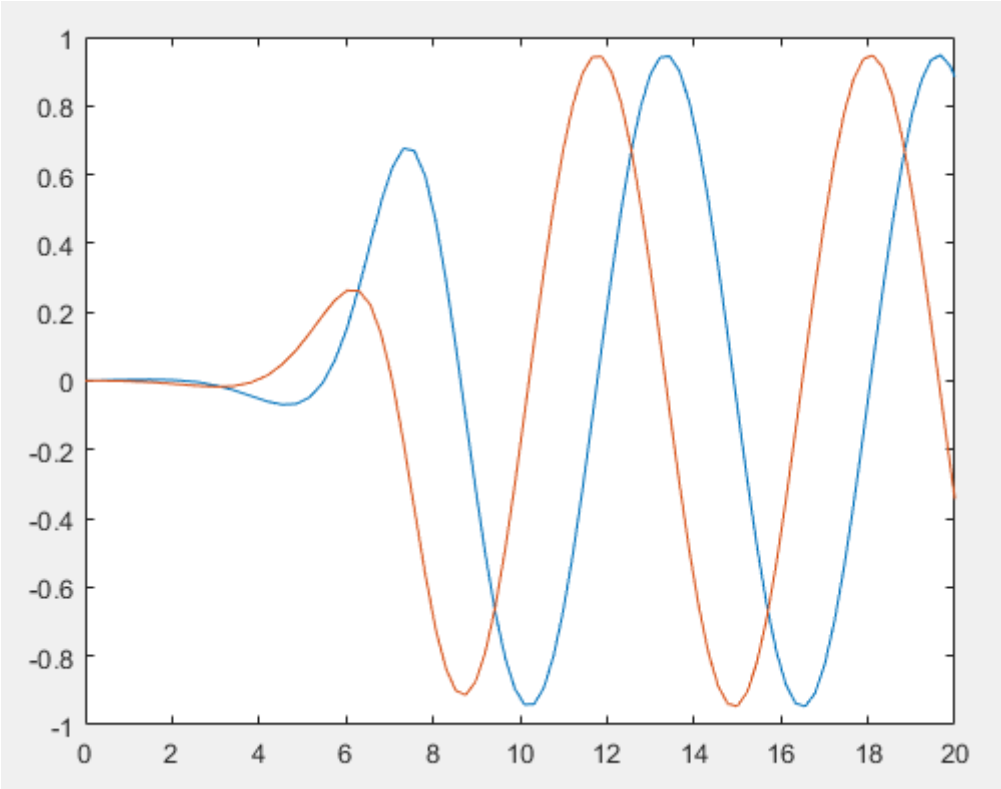

**b)** $x_1(0)=0$ $x_2(0)=9$



**c)** $x_1(0)=8$ $x_2(0)=0$




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



**Czasy wykonania i liczba iteracji:**

| Zestaw | | Rk4 ze stałym kr. | PKA4 ze stałym k. | Rk4 ze zmiennym k. |
|---|---|---|---|---|
| a | Czas | 0.018952 | 0. 018517 | 0. 012731 |
| | Liczba iteracji | 1819 | 1819 | 377 |
| | krok początkowy | 0.014 | 0.011 | 0.001 |
| b | Czas | 0. 007571 | 0. 012181 | 0. 013273 |
| | Liczba iteracji | 667 | 667 | 393 |
| | krok początkowy | 0.03 | 0.03 | 0.03 |
| c | Czas | 0. 008093 | 0.010524 | 0.013707 |
| | Liczba iteracji | 500 | 500 | 407 |
| | krok początkowy | 0.04 | 0.04 | 0.04 |
| d | Czas | 0. 002875 | 0.004404 | 0. 011940 |
| | Liczba iteracji | 200 | 200 | 343 |
| | krok początkowy | 0.1 | 0.1 | 0.2 |

**Wnioski:**

Metody RK4 oraz predyktor-korektor mają taką samą ilość iteracji i jest ona zależna jedynie od długości kroku. Metoda RK4 jest w 3 przypadkach najszybsza. Jednak jej dokładność zależy jedynie od długości kroku oraz nie bierze ona pod uwagę błędu rozwiązania. Metoda RK4 ze zmiennym krokiem wykonuje dużo mniej iteracji. Jest to wywołane tym, że krok jest zwiększany jeżeli nie sprawi to wzrostu błędu. Zmiana kroku początkowego w metodzie RK4 ze zmiennym krokiem zmienia ilość iteracji o kilkanaście-kilkadziesiąt. Jest to spowodowane faktem, że algorytm potrzebuje więcej obliczeń, aby prawidłowo ustawić krok.

Dla dobrze dobranych kroków wszystkie metody rozwiązują zadanie poprawie – ciężko zauważyć różnice na wykresach wykonanymi przy wykorzystaniu polecenia ode45. Czasy wykonania są dla wszystkich algorytmów podobne. Metoda ze zmiennym krokiem jest zazwyczaj najwolniejsza.

Jednak problemem jest fakt, że przy metodzie RK4 ze zmiennym krokiem nie wiemy z góry ile będziemy mieli punktów, dlatego nie możemy na początku zaalokować potrzebnej pamięci, a zmienianie rozmiaru zmiennych w trakcie wykonania znacznie wydłuża czas działania.