

Dokumentacja projektu laboratoryjnego numer 4 przedmiot MNUM

Marcin Dziedzic

10 czerwca 2018

Spis treści

| | | |
|----------|---|-----------|
| 1 | Zadanie 1 | 1 |
| 1.1 | Polecenie | 1 |
| 1.2 | Ogólny opis zagadnienia rozwiązywania układu równań różniczkowych | 2 |
| 2 | Metoda RK4 ze stałym krokiem | 2 |
| 2.1 | Opis algorytmu | 2 |
| 2.2 | Kod funkcji zwracającej punkt określony równaniami | 2 |
| 2.3 | Kod funkcji zwracającej rozwiązanie metodą RK4 | 2 |
| 2.4 | Kod programu generujący dane wynikowe dla podpunktu 1 | 3 |
| 2.5 | Dobieranie długości kroku | 3 |
| 2.6 | Wyniki dla podpunktu 1 | 4 |
| 2.7 | Wnioski | 8 |
| 3 | Metody wielokrokowe | 8 |
| 3.1 | Metoda predyktor-korektor Adamsa | 8 |
| 3.2 | Realizacja funkcji RK4 w programie Matlab | 8 |
| 3.3 | Skrypt generujący wykresy do zadania | 9 |
| 3.4 | Dobieranie długości kroku | 9 |
| 3.5 | Wyniki | 10 |
| 3.6 | Wnioski | 14 |
| 4 | Metoda RK4 ze zmiennym krokiem | 14 |
| 4.1 | Opis algorytmu | 14 |
| 4.2 | Realizacja funkcji w programie Matlab | 14 |
| 4.3 | Funkcja generująca rozwiązania do zadania | 15 |
| 4.4 | Wnioski | 15 |
| 5 | Porównanie z wynikami funkcji ode45 | 16 |
| 6 | Wnioski końcowe | 16 |

1 Zadanie 1

1.1 Polecenie

Ruch punktu jest opisany równaniami:

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

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

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

$$\begin{aligned}
a) x_1(0) &= 8, x_2(0) = 7 \\
b) x_1(0) &= 0, x_2(0) = 0,4 \\
c) x_1(0) &= 5, x_2(0) = 0 \\
d) x_1(0) &= 0,01, x_2(0) = 0,001
\end{aligned}$$

1.2 Ogólny opis zagadnienia rozwiązywania układu równań różniczkowych

Rozważane jest zagadnienie układu równań (pogrubione wartości to wektory) różniczkowych zwyczajnych pierwszego rzędu (ale mogą być one nieliniowe). Dane jest równanie (układ równań):

$\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x})$, przy czym \mathbf{x} to szukana funkcja. Znany jest przedział, na którym szukamy \mathbf{x} : $t \in [a, b]$ oraz warunki początkowe: $\mathbf{x}(a)$. Wyróżnia się metody jednokrokowe, bazujące tylko na punkcie otrzymanym w poprzedniej iteracji, oraz metody wielokrokowe, które opierają się na większej liczbie punktów.

2 Metoda RK4 ze stałym krokiem

2.1 Opis algorytmu

Metody Rungego - Kuty to grupa metod jednokrokowych. Na przedziale $[t_n, t_n + h]$ obliczane są pochodne \mathbf{x} (poprzez podstawienie do funkcji \mathbf{f} danej równaniu), w różnych punktach w badanym przedziale, a następnie badana funkcja jest przybliżana przez pewną liniową kombinację tych pochodnych. Kompromisem pomiędzy dokładnością (rzędem metody) a nakładem obliczeń na jedną iterację jest metoda RK4 (obliczenie pochodnej w 4 punktach). Wzory opisujące jedną iterację:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{1}{6}h(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

$$\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{x}_n)$$

$$\mathbf{k}_2 = \mathbf{f}(t_n + \frac{1}{2}h, \mathbf{x}_n + \frac{1}{2}h\mathbf{k}_1)$$

$$\mathbf{k}_3 = \mathbf{f}(t_n + \frac{1}{2}h, \mathbf{x}_n + \frac{1}{2}h\mathbf{k}_2)$$

$$\mathbf{k}_4 = \mathbf{f}(t_n + h, \mathbf{x}_n + h\mathbf{k}_3)$$

2.2 Kod funkcji zwracającej punkt określony równaniami

```

function [D] = md_fx(x)                                1
% Funkcja zwracająca pochodne                          2
D(1) = x(2) + x(1)*(0.2 -x(1)^2 -x(2)^2);             3
D(2) = -x(1) + x(2)*(0.2 -x(1)^2 -x(2)^2);           4
end                                                    5

```

2.3 Kod funkcji zwracającej rozwiązanie metodą RK4

```

function [Y] = md_rk4s(x,timelimit,stp)           1
% Rozwiązanie układu metoda Rungego–Kutty czwartego rzędu 2
% x – stan początkowy                               3
% timelimit – zakres czasu                           4
% step – rozmiar kroku                               5
                                                    6
Y = zeros(ceil(timelimit/stp),3); %macierz stanów x1, x2 i czasu 7
hstp=stp/2; %połowa kroku                             8
for i = 1:(ceil(timelimit/stp))                     9
    Y(i,3) = i*stp;                                  10
    k1 = md_fx(x);                                  11
    k2 = md_fx(x+hstp*k1);                          12
    k3 = md_fx(x+hstp*k2);                          13
    k4 = md_fx(x+stp*k3);                          14
    x=x+(1/6)*stp*(k1+2*k2+2*k3+k4);                15
    Y(i,1:2) = x;                                   16
    %zapisanie punktu do wektora                     17
end                                                  18
end                                                  19

```

2.4 Kod programu generujący dane wynikowe dla podpunktu 1

```

% Realizacja zadania 1                               1
% metoda RK4 ze stałym krokiem                       2
clear;                                                3
zero=[8 7; 0 0.4; 5 0; 0.01 0.001]; %wektor stanów początkowych 4
step = 0.0004; %krok                                5
                                                    6
for k = 3:3                                           7
    data = md_rk4s(zero(k,:),20,step);                8
    h = figure;                                       9
    plot(data(:,1),data(:,2),'-o');                  10
    l = size(data,1);                                11
    hold on;                                          12
    grid on;                                          13
    name = [ 'metoda RK4 krok:' num2str(step) ' podpunkt:' num2str(k) ]; 14
    title(name);                                     15
    saveas(h,name,'jpg');                             16
end                                                  17

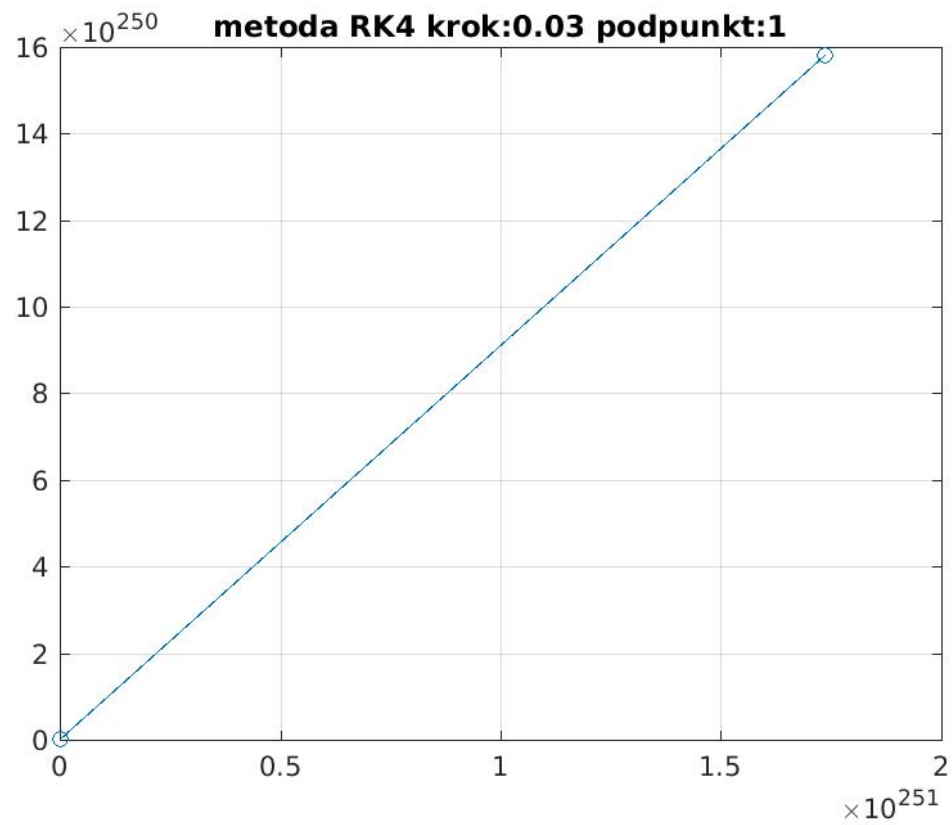
```

2.5 Dobieranie długości kroku

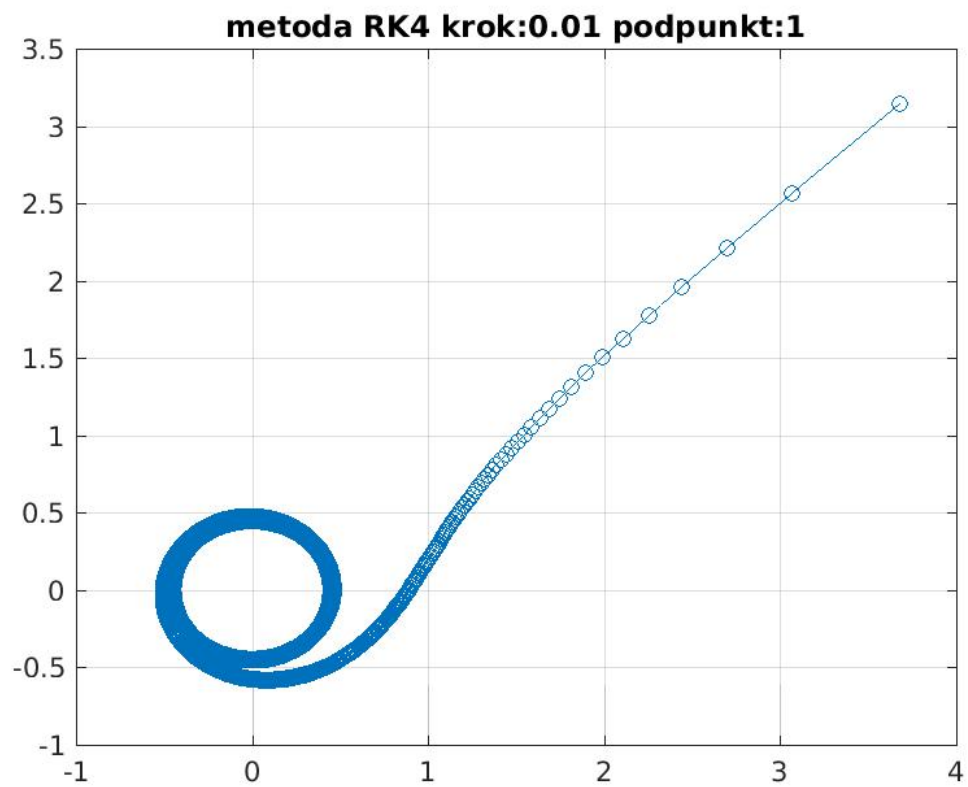
Największym wyzwaniem w powyższej metodzie jest dobranie odpowiedniej długości kroku. Moim zadaniem było dobranie go w taki sposób, aby był wystarczający do uzyskania założonej dokładności, ale nie powinien być znacznie mniejszy od wartości, przy której wymagana dokładność jest osiągalna. Postanowiłem dokonać tego interaktywnie, na zasadzie prób i błędów. Posługiwałem się metodą przeszukiwania binarnego. Zainicjalizowałem zakres przeszukiwań od kroku bardzo małego do bardzo dużego, następnie wybierałem wartość środkową. W zależności, czy chciałem osiągnąć lepszą lub gorszą aproksymację wybierałem lewą lub prawą połowę kolejnego przedziału.

2.6 Wyniki dla podpunktu 1

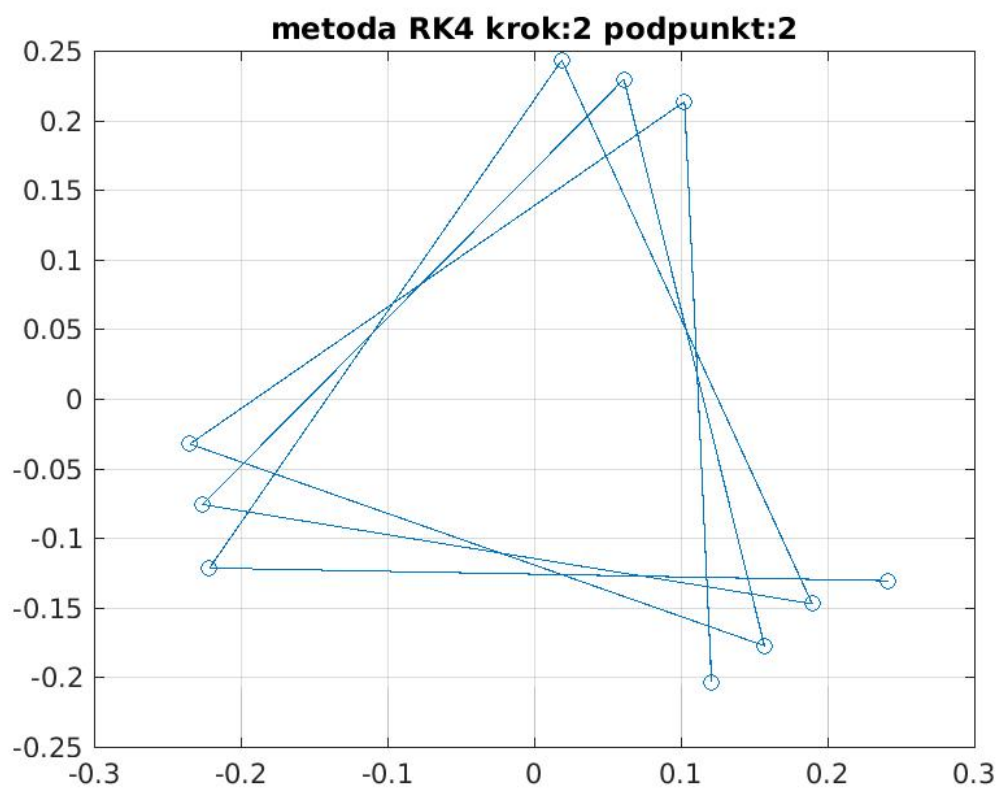
RK4 krok:0,03 podpunkt:1.jpg



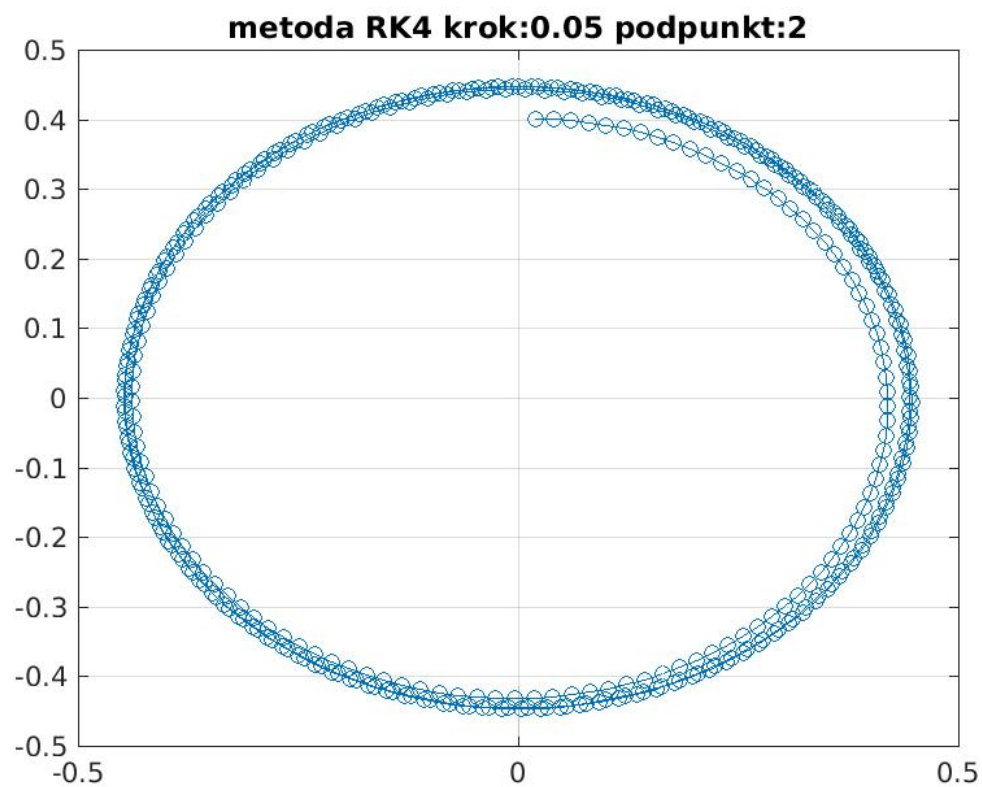
RK4 krok:0,01 podpunkt:1.jpg



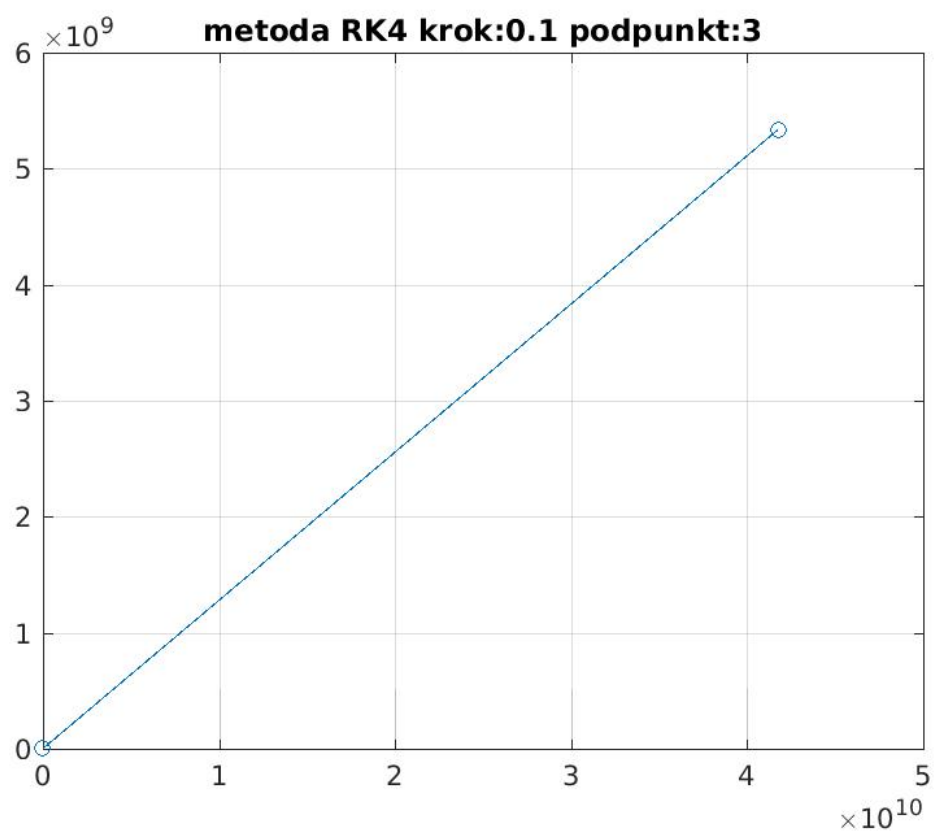
RK4 krok:2 podpunkt:2.jpg



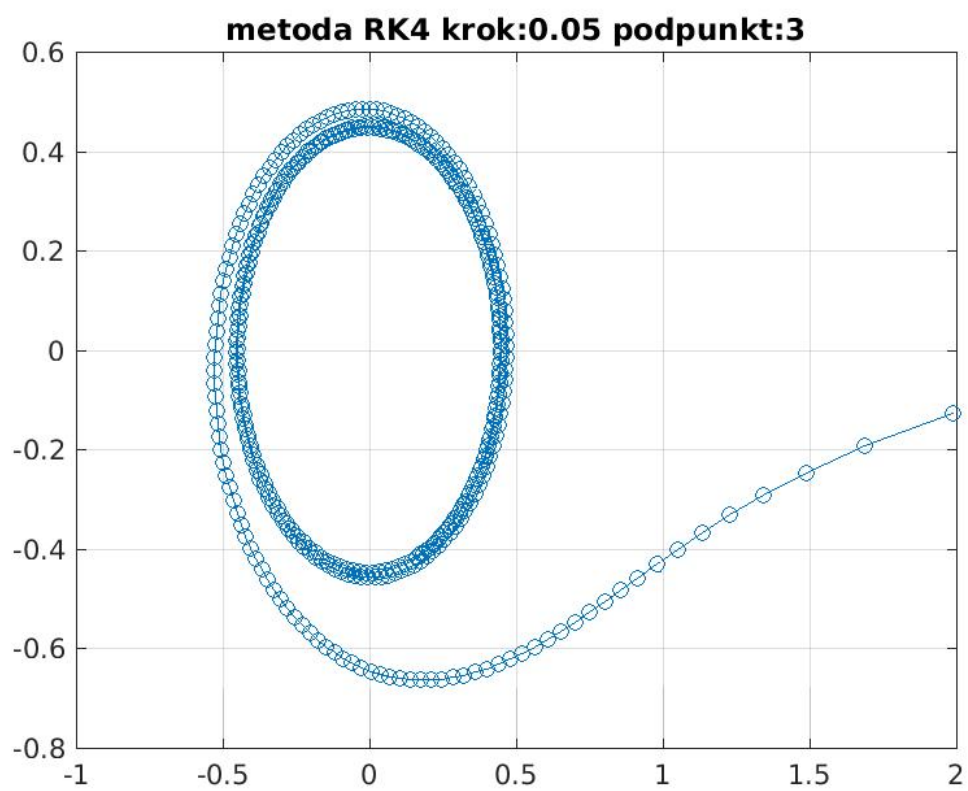
RK4 krok:0,05 podpunkt:2.jpg



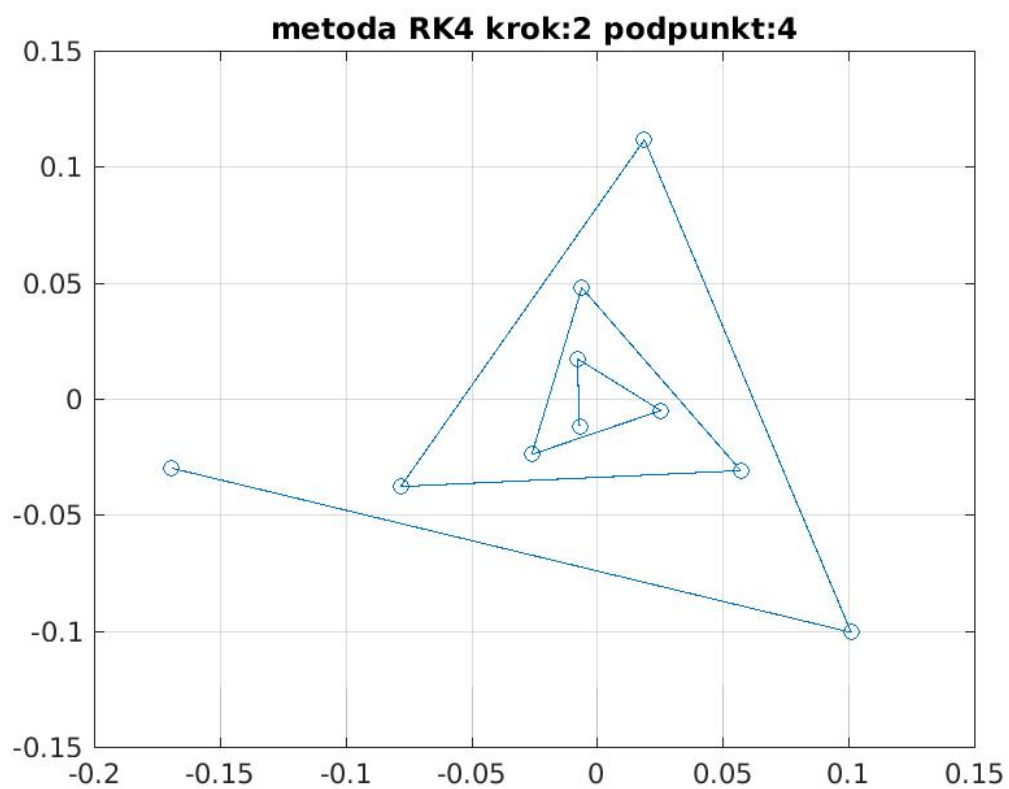
RK4 krok:0,1 podpunkt:3.jpg



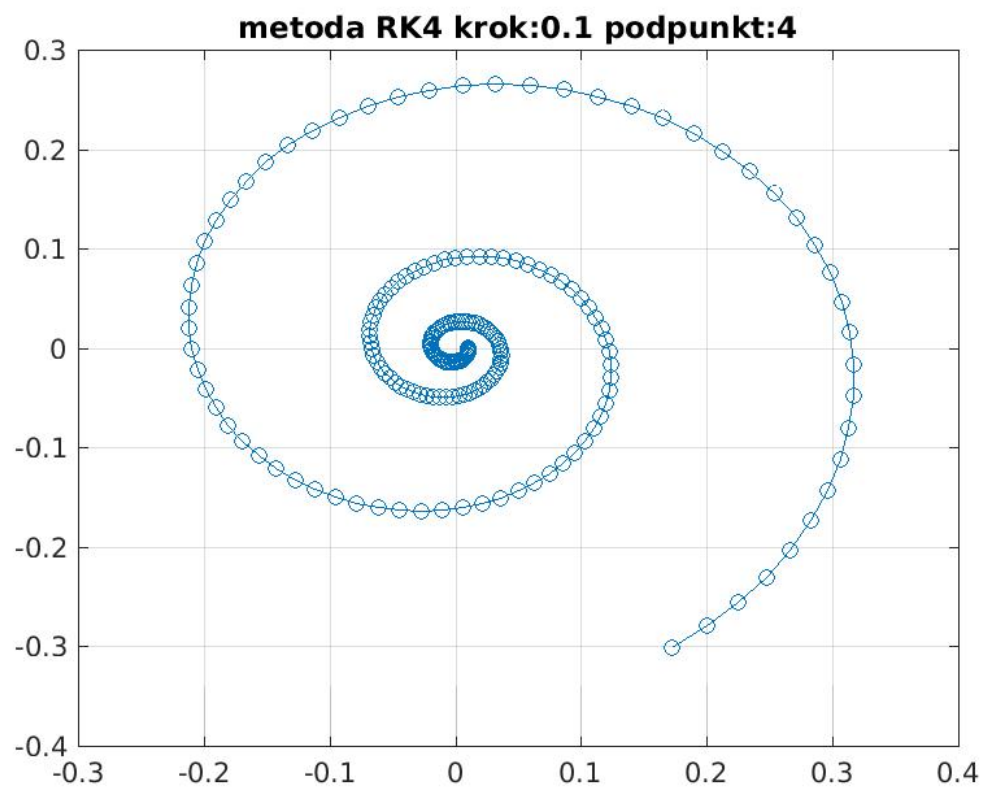
RK4 krok:0,05 podpunkt:3.jpg



RK4 krok:2 podpunkt:4.jpg



RK4 krok:0,1 podpunkt:4.jpg



2.7 Wnioski

Metoda Rungego-Kutty dla podanych równań jest szybka przy odpowiednim kroku. Zmniejszając krok można coraz dokładniej wyznaczyć tor ruchu. Dobór odpowiedniej długości kroku w sposób interaktywny jest problematyczny ze względu na różnorodność kroków w zależności od punktów startowych. Metoda również ma dość wysoki nakład obliczeniowy ponieważ jest on zależny tylko od przedziału i długości kroku, dlatego nie ma znaczenia czy równania są skomplikowane czy bardzo proste.

3 Metody wielokrokowe

Metody wielokrokowe w odróżnieniu od iteracyjnych używają do wyznaczenia punktu wartości obliczonych w poprzednich krokach. Wynika z tego że przy rozpoczynaniu rozwiązywania zagadnienia metodą wielokrokową musimy wyznaczyć punkty początkowe (zazwyczaj jedną z metod iteracyjnych) a następnie na podstawie wartości początkowych obliczane są kolejne punkty. Metod tych używa się w przypadku gdy wyznaczanie wartości bezpośrednio ze wzoru jest czasochłonne, ponieważ metody wielokrokowe rzadziej wykorzystują bezpośrednie obliczanie wartości funkcji. Wyróżniamy metody jawne które wykorzystują jedynie wartości funkcji w poprzednio obliczonych punktach oraz niejawne, które dodatkowo korzystają z wartości w punkcie bieżącym.

3.1 Metoda predyktor-korektor Adamsa

Metoda predykcyjno-korekcyjna łączy zalety metod jawnych i niejawnych. Dzięki temu zachowuje wysoki rząd i małą stałą błędów na szerokim obszarze przy zachowaniu małej liczby iteracji. Praktyczna realizacja metody polega na obliczeniu czterech członów: P - predykcja, E - ewaluacja, K - korekcja, E - kolejna ewaluacja. Dla metody predyktor - korektor Adamsa rozwiązanie możemy przybliżyć następującymi równaniami:

$$\begin{aligned}P : y_n^{[0]} &= y_{n-1} + h \sum_{j=1}^k \beta_j f_{n-j} \\E : f_n^{[0]} &= f(x_n, y_n^{[0]}) \\K : y_n &= y_{n-1} + h \sum_{j=1}^k \beta_j^* f_{n-j} + h \beta_0^* f_n^{[0]} \\E : f_n &= f(x_n, y_n)\end{aligned}$$

3.2 Realizacja funkcji RK4 w programie Matlab

```
function [Y] = md_pkadams(x, timelimit, stp)           1
%funkcja zwracająca wektor wyznaczony metoda PK adamsa 2
%dane wejściowe                                         3
%x – wektor danych                                    4
%timelimit – zakres czasu                             5
%stp – długość kroku                                  6
                                                        7
Y = zeros(timelimit/stp, 3); %wektor stanów x1, x2, czasu, i błędów 8
hstp = stp/2;                                          9
for i = 1:3 %generowanie punktów początkowych        10
    Y(i, 3) = i*stp; %generowanie czasu              11
    k1 = md_fx(x); %pochodna w punkcie y(xn);        12
    k2 = md_fx(x+hstp*k1); %pochodna w punkcie y(xn+stp/2*k1) 13
    k3 = md_fx(x+hstp*k2); %pochodna w punkcie y(xn+stp/2*k2) 14
    k4 = md_fx(x+stp*k3); %pochodna w punkcie y(xn+stp*k3)   15
    x = x + (1/6)*stp*(k1+2*k2+2*k3+k4); %obliczenie następnego punktu 16
    Y(i, 1:2) = x; %zapisanie punktu do wektora      17
```



```

end 18
for i = 4:(timelimit/stp) 19
    Y(i,3) = i*stp; %generowanie czasu 20
    tmp = x + stp/24*(55*md_fx(x) - 59*md_fx(Y(i-1,1:2)) + 37*md_fx(Y 21
        (i-2,1:2)) - 9*md_fx(Y(i-3,1:2))); % predykcja i ewaluacja
    x = x + stp/24*(9*md_fx(tmp) + 19*md_fx(x) - 5*md_fx(Y(i-1,1:2)) 22
        + md_fx(Y(i-2,1:2))); % korekcja i ewaluacja
    Y(i,1:2) = x; %zapis wyniku 23
end 24
end 25

```

3.3 Skrypt generujący wykresy do zadania

```

% Realizacja zadania 2 1
% metoda metoda predyktor-korektor Adamsa 4-rzedu 2
clear; 3
zero=[8 7; 0 0.4; 5 0; 0.01 0.001]; %wektor stanów początkowych 4
step = 0.01; %krok 5
6
for k = 1:1 7
8
    data = md_pkadams(zero(k,:),20,step); 9
10
    h = figure;%('visible','off'); 11
    plot(data(:,1),data(:,2),'-o'); 12
    l = size(data,1); 13
    hold on; 14
    xl = get(gca,'xlim'); 15
    yl = get(gca,'ylim'); 16
    zl = get(gca,'zlim'); 17
    grid on; 18
    name = [ 'metoda Adamsa krok:' num2str(step) ' podpunkt:' num2str(k) 19
        ];
    title(name); 20
    saveas(h,name,'jpg'); 21
end 22

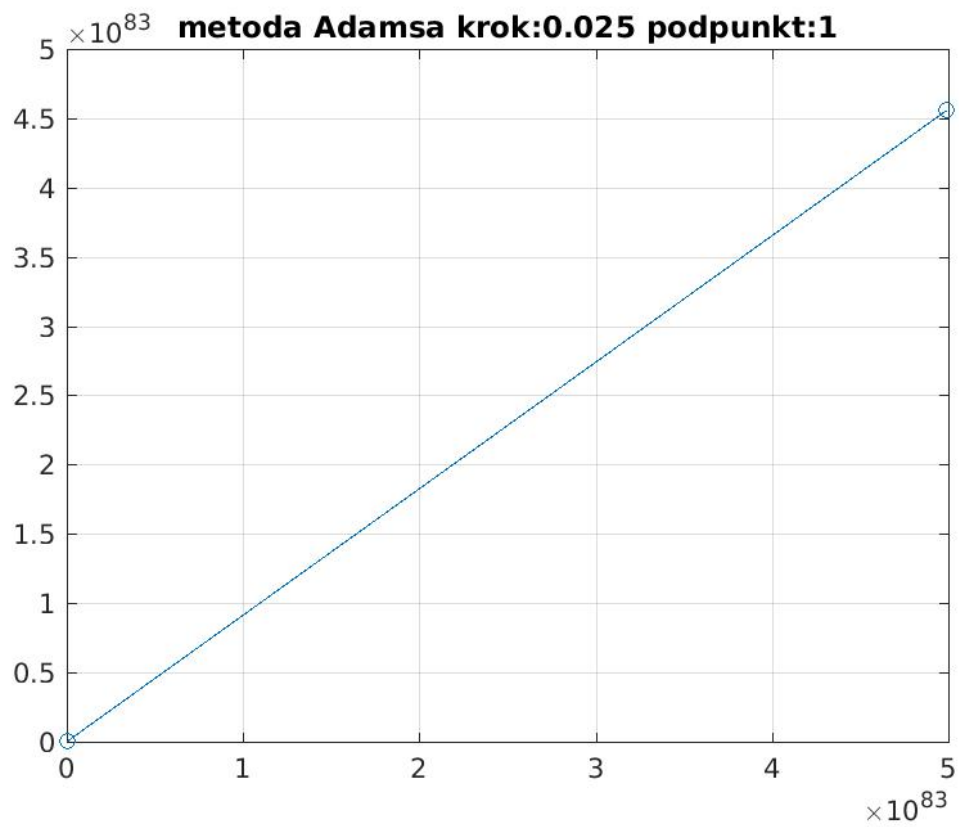
```

3.4 Dobieranie długości kroku

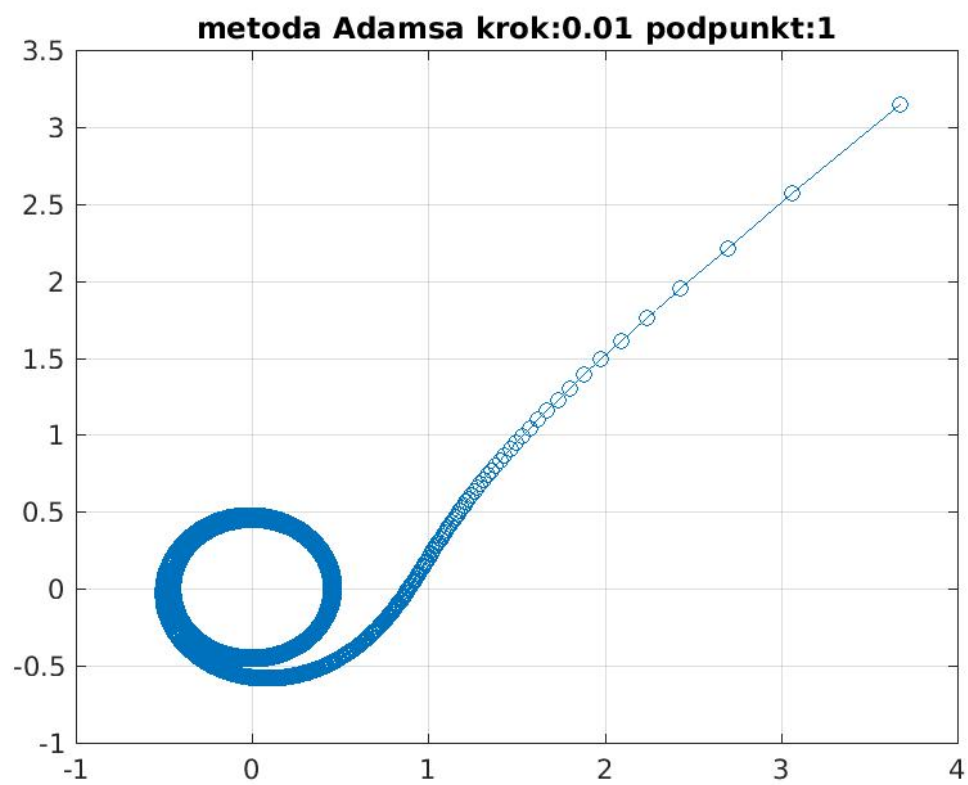
Podobnie jak w poprzednim podpunkcie, długość kroku była wprowadzana przez użytkownika w trybie interaktywnym.

3.5 Wyniki

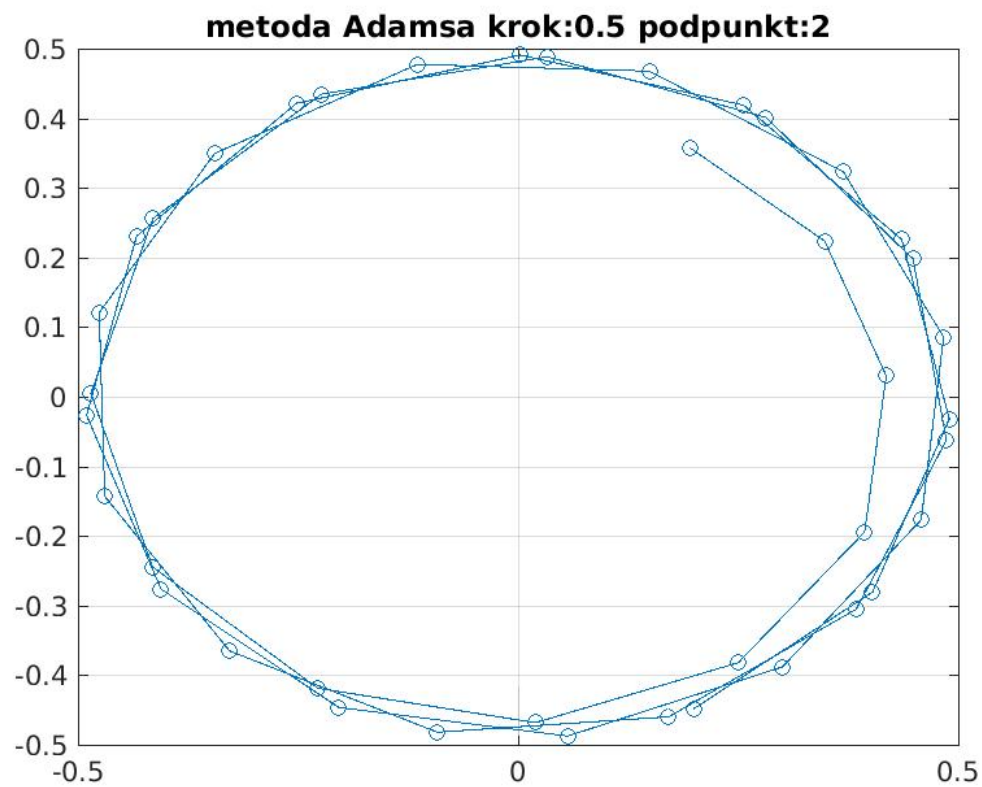
Adamsa krok:0,025 podpunkt:1.jpg



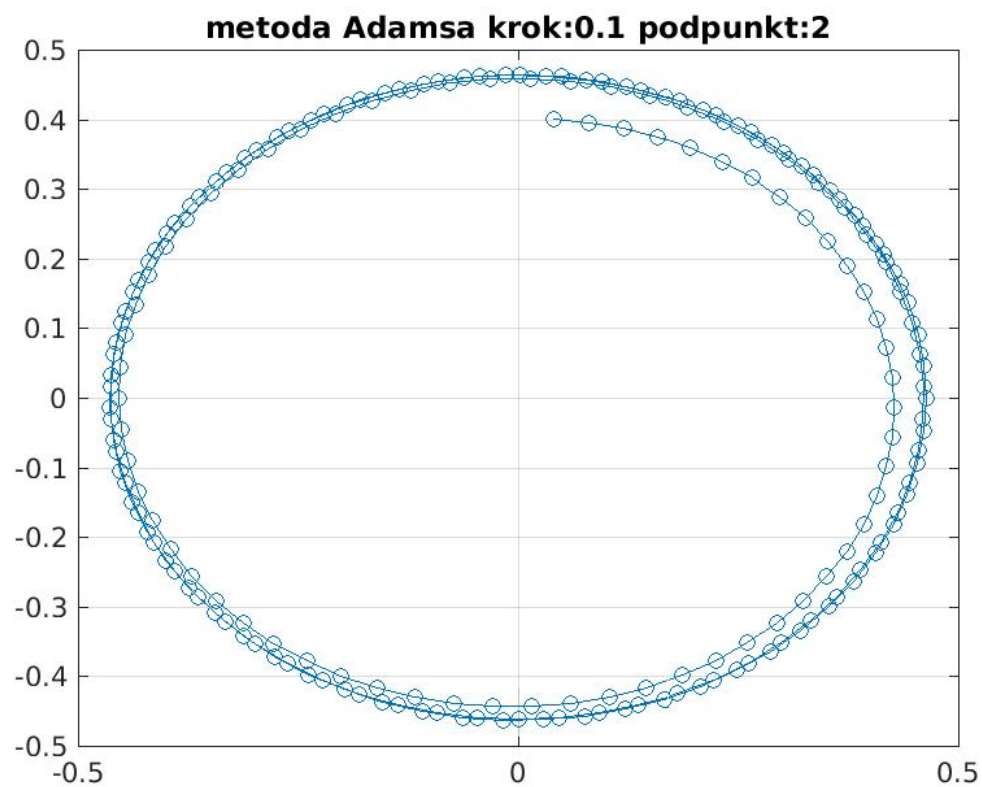
Adamsa krok:0,01 podpunkt:1.jpg



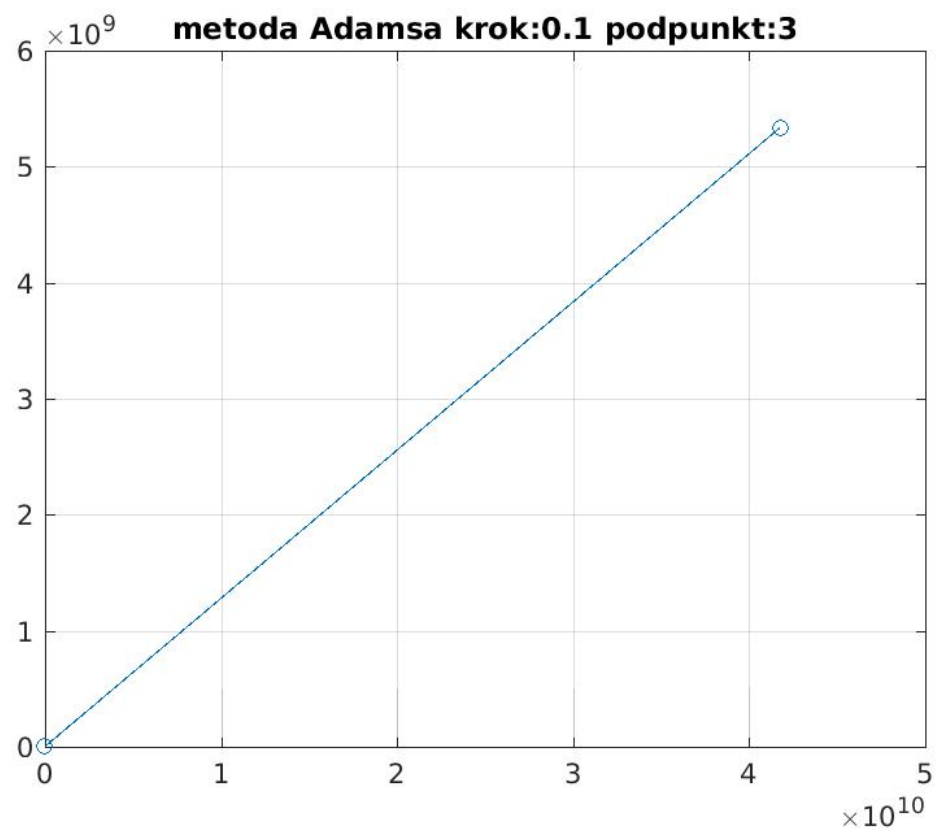
Adamsa krok:0,5 podpunkt:2.jpg



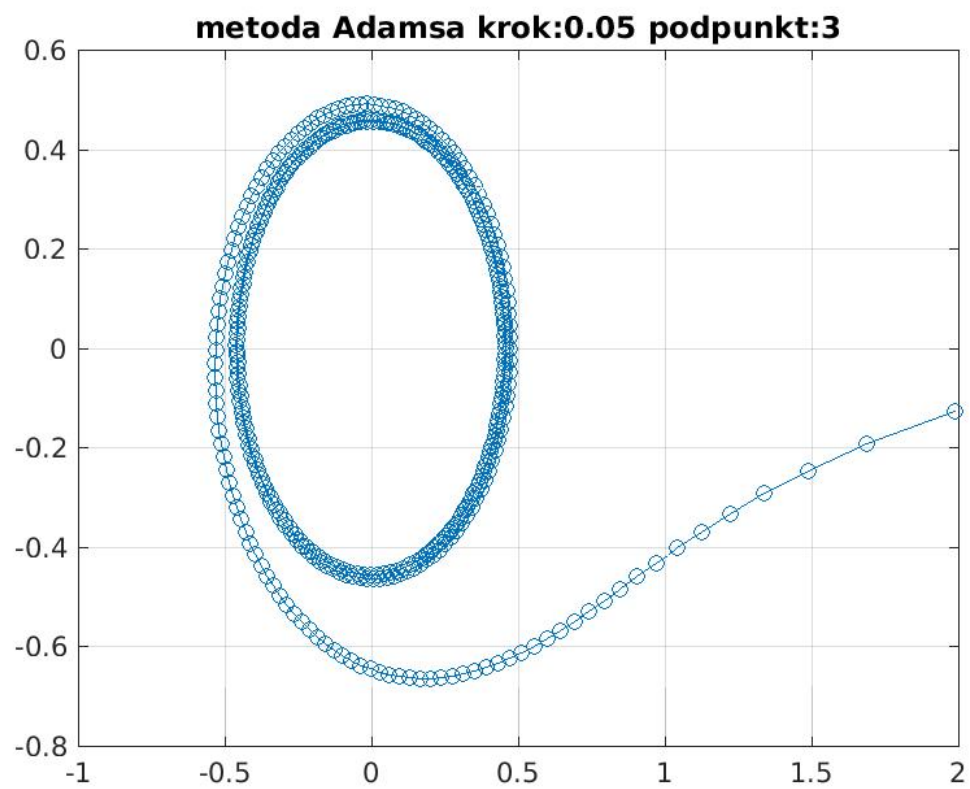
Adamsa krok:0,1 podpunkt:2.jpg



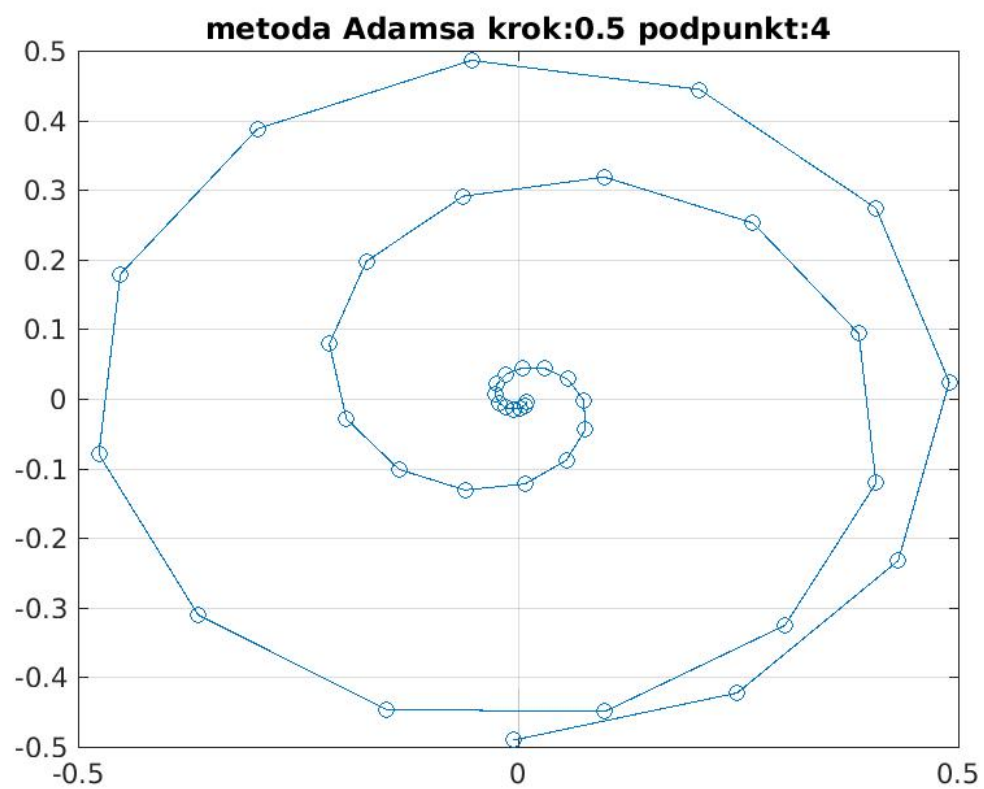
Adamsa krok:0,1 podpunkt:3.jpg



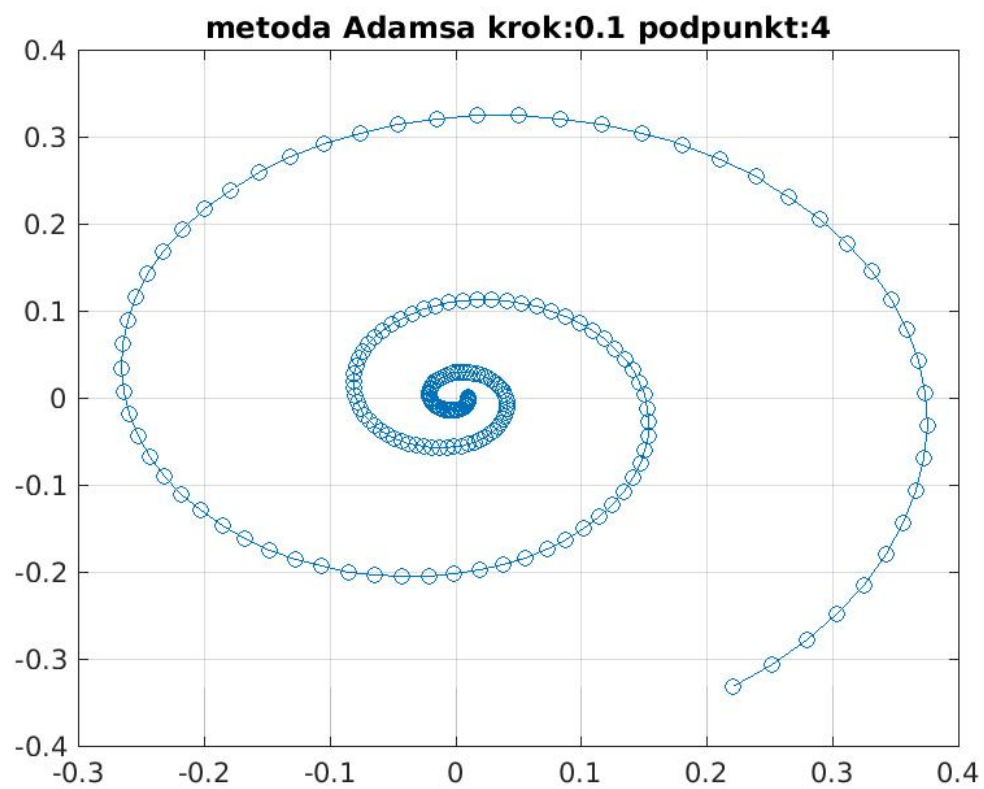
Adamsa krok:0,05 podpunkt:3.jpg



Adamsa krok:0,5 podpunkt:4.jpg



Adamsa krok:0,1 podpunkt:4.jpg



3.6 Wnioski

Metoda predyktor-korektor Adamsa zdecydowanie dokładniej określa trajektorium niż metoda RK4 dla tego samego kroku, jednak potrzebuje znacznie mniejszego kroku do "ustabilizowania się". Różnica pomiędzy krokiem dla którego metoda jest zbieżna a krokiem dla którego jest dokładna jest niewielka w przeciwieństwie do metody RK4. Dla dużych punktów początkowych metoda podobnie jak RK4 potrzebuje dosyć małego kroku aby uzyskać zbierczość na danym przedziale.

4 Metoda RK4 ze zmiennym krokiem

4.1 Opis algorytmu

Jeżeli chodzi o metodę obliczeń to jest ona identyczna jak w tej ze stałym krokiem. Na szczególną uwagę zasługuje sposób oceny błędu i na tej podstawie regulacji długości kroku. Określanie błędu jest realizowane metodą połowienia kroku. W pojedynczej iteracji obliczne są wartości funkcji dla całego kroku i jego połowy, następnie jeżeli różnica między otrzymanymi wartościami jest mniejsza niż założony współczynnik dokładności, to krok jest zwiększany, a gdy większa to zmniejszany. Jako wartość funkcji brana pod uwagę jest zawsze ta wyznaczona za pomocą dwóch mniejszych kroków. Długość kroku jest dodatkowo mnożona przez współczynnik bezpieczeństwa tak aby wzrost kroku nie był zbyt drastyczny, i aby również drastycznie nie zmalał, co doprowadziło by do znacznego wydłużenia czasu obliczeń.

4.2 Realizacja funkcji w programie Matlab

```
function [Y] = rk4dynamic(x,timelimit,stp)           1
%RK4 Rozwiązanie układu metoda Rungego-Kutty czwartego rzędu ze zmiennym      2
%krokiem                                           3
%x – stan początkowy                             4
%timelimit – zakres czasu                         5
%step – rozmiar kroku                             6
                                                    7
                                                    8
Y=zeros(10,3);                                     9
x1 = x;                                           10
x2 = x;                                           11
                                                    12
time = 0; %zmienna przechowująca aktualny przedział czasu                    13
i=1; %iterator indeksu wektorów                                              14
                                                    15
while(time<=timelimit)                                                            16
                                                    17
    k1 = f_x(x1);                                                                18
    k2 = f_x(x1+(stp/2)*k1);                                                    19
    k3 = f_x(x1+(stp/2)*k2);                                                    20
    k4 = f_x(x1+stp*k3);                                                         21
    x1=x1+(1/6)*stp*(k1+2*k2+2*k3+k4);                                          22
                                                    23
    k1 = f_x(x2);%obliczenie wartości z połowicznym krokiem                 24
    k2 = f_x(x2+(stp/4)*k1);                                                    25
    k3 = f_x(x2+(stp/4)*k2);                                                    26
    k4 = f_x(x2+(stp/2)*k3);                                                    27
    x2=x2+(1/6)*(stp/2)*(k1+2*k2+2*k3+k4);                                      28
    k1 = f_x(x2);                                                                29
    k2 = f_x(x2+(stp/4)*k1);                                                    30
```

```

k3 = f_x(x2+(stp/4)*k2);
k4 = f_x(x2+(stp/2)*k3);
x2=x2+(1/6)*(stp/2)*(k1+2*k2+2*k3+k4);

R = (x2-x1)/15;

if(abs(min(R))<0.00001) %kryterium bledu względnego
    stp = stp*1.1;
else
    if(stp>0.05)
        stp = stp*0.9;
    end
end
Y(i,1:2) = x2;
time = time+stp;
disp(stp);
Y(i,3) = time; %zapisanie czasu
i = i+1;
end
end

```

4.3 Funkcja generująca rozwiązania do zadania

```

%Realizacja podpunktu 3 metoda metoda RK4 zmienny krok
clear;
zero=[8 7; 0 0.4; 5 0; 0.01 0.001]; %wektor krokow
step = 0.01; %krok

for k = 1:4

    data = rk4dynamic(zero(k,:),20,step);

    h = figure('visible','off');
    plot(data(:,1),data(:,2),'-o');
    l = size(data,1);
    hold on;
    xl = get(gca,'xlim');
    yl = get(gca,'ylim');
    zl = get(gca,'zlim');
    %scatter3(data(:,1),data(:,2), repmat(zl(1),l,1),'.');
    %scatter3(data(:,1), repmat(yl(2),l,1),data(:,3),'.');
    %scatter3(repmat(xl(2),l,1),data(:,2),data(:,3),'.');
    grid on;
    name = ['metoda RK4 zmienny krok podpunkt:' num2str(k)];
    title(name);
    saveas(h,name,'jpg');
end

```

4.4 Winoski

Metoda RK4 ze zmiennym krokiem znacznie wydajniej radzi sobie z obliczeniami ze względu na istotne zmniejszenie liczby kroków na gładkich odcinkach funkcji. Zaimplementowana przeze mnie metoda doboru długości kroku nie należy jednak do wyrafinowanych. Przez co wzrost wydajności nie jest szczególnie mocno zauważalny.

5 Porównanie z wynikami funkcji ode45

Funkcja ode45 dobrała znacznie większe kroki co widać na wykresach, przez to czas obliczeń był rzędu razy mniejszy niż zaimplementowanych przeze mnie metod. Jeżeli chodzi o dokładność to wykresy wygenerowane funkcją ode45 są dużo mniej "gładkie" jednak krok jest dobierany równomiernie, i nie widać nakładających się na siebie punktów na wykresie, co świadczy o optymalności metody.

6 Wnioski końcowe

W przypadku metod jednopunktowych stałokrokowych najlepiej wypada metoda adamsa ze względu na najniższy czas wykonania obliczeń, jednak jest ona zbierzna w wąskim przedziale kroku. Metoda RK4 ze zmiennym krokiem również bardzo dokładnie wyznacza trajektoriam jednak zdecydowanie przegrywa z metodą ode45 ze zmiennym krokiem pod względem wydajności. Przypuszczem że zastosowanie zmiennego kroku dla metody predyktor-korektor Adamsa mogło by dać najlepszy rezultat dla danych z tego zdania, ponieważ wymaga najmniejszej liczby kroków przy wysokiej dokładności.