Metody Numeryczne

Kacper Marchlewicz

Projekt 4 Zadanie 4.7

1. 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] dal następujących warunków początkowych:

a) $x_1(0) = 10$ $x_2(0) = 9$
b) $x_1(0) = 0$ $x_2(0) = 7$
c) $x_1(0) = 7$ $x_2(0) = 0$
d) $x_1(0) = 0{,}001$ $x_2(0) = 0{,}001$

Do rozwiązania zadania należy użyć zaimplementowanych przez siebie w języku Matlaba (w formie solwerów) metod:

1. Rungego-Kutty czwartego rzędu (RK4) ze stałym krokiem. Proszę przy tym wykonać tyle prób (kilka – kilkanaście), ile będzie potrzebnych do znalezienia takiego kroku, którego zmniejszanie nie wpływa znacząco na na rozwiązanie, podczas gdy jego zwiększanie – już wpływa;

2. Wielokrokowej predyktor – korektor Adamsa czwartego rzędu ze stałym krokiem, który należy dobrać w sposób z punktu 1.;

3. Rungego-Kutty czwartego rzędu (RK4) ze zmiennym krokiem. W każdym kroku należy szacować błąd aproksymacji.

Metoda Rungego-Kutty czwartego rzędu

Jest jedną z metod jednokrokowych. Polega na obliczaniu wartości prawych stron równań różniczkowych dokładnie m razy, wtedy mówimy o m - etapowym kroku iteracji. W tym przypadku $m = 4$. Metody czwartego rzędu z $m = 4$ to udany kompromis między dokładnością (rząd metody) a nakładem obliczeń na jedną iterację, co za tym idzie z wpływem błędów powodowanych przez zaokrąglanie. W RK4 wzory na poszczególne współczynniki wyglądają następująco:

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

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

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

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

$$k_4 = f(x_n + h, y_n + hk_3),$$

Współczynnik k_1 jest pochodną rozwiązania w punkcie (x_n, y_n) . Wartość k_2 wyznaczamy jak w zmodyfikowanej metodzie Eulera – jako pochodną rozwiązania wyznaczanego zwykłą metodą Eulera w punkcie $(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1)$ (środkowym przedziału). Z k_3 *i* k_4 jest podobnie. W ten sposób mamy wyznaczone 4 wartości pochodnej rozwiązania: po jednej na końcach przedziału i dwie w jego środku. Aproksymacja pochodnej dla pełnego kroku wyznaczana jest jako średnia arytmetyczna tych wartości.

Krok metody powinien być dobierany tak, aby był to największy możliwy krok dla którego metoda jest zbieżna. Wynika to z faktu, ze zbyt mały krok powoduje nawarstwianie się błędów numerycznych obliczeń, co skutkuje zmniejszeniem się dokładności metody.

Implementacja w programie matlab:

```
function[y] = rk4(dx1,dx2,x1,x2,h)
tic;
t=21/h; %liczba kroków
y(:,1) = [x1 x2]; %początkowe dane do wykresu
for i=1:t
    k11=dx1(x1,x2);
    k12=dx2(x1,x2);
    k21=dx1(x1+0.5*h,x2+0.5*h*k11);
    k22=dx2(x1+0.5*h,x2+0.5*h*k12);
    k31=dx1(x1+0.5*h,x2+0.5*h*k21);
    k32=dx2(x1+0.5*h,x2+0.5*h*k22);
    k41=dx1(x1+h,x2+h*k31);
    k42=dx2(x1+h,x2+h*k31);
    x1=x1+(h/6)*(k11+2*k21+2*k31+k41);
    x2=x2+(h/6)*(k12+2*k22+2*k32+k42);
    y(:,i+1)=[x1 x2]; %zwiększenie bazy danych do wykresu
end
toc;
end
```

Metoda predyktor – korektor Adamsa czwartego rzędu

Jest to metoda wielokrokowa, ze stałym krokiem. Opiera się na algorytmie typu predyktor – korektor, będącym połączeniem metod jawnych i niejawnych. Dzięki temu posiada następujące zalety:

1) Wysoki rząd i małą stałą błędu

2) Możliwie duży obszar absolutnej stabilności

3) Możliwie małą liczbę obliczeń na iterację

Realizacja w postaci struktury PK dla metody k-krokowej:

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

E – ewaluacja: $$f_n^{[0]} = \mathrm{f}(x_n, y_n^{[0]})$$

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

E – ewaluacja: $$f_n = (x_n, y_n)$$

Iteracja predyktora ma na celu obliczenie dobrego punktu początkowego (tym lepszy, im mniejszy krok h i wyższy rząd predykatora) dla iteracji korektora rozwiązującego nieliniowe równanie algebraiczne metodą iteracji prostej.

Dla metod Adamsa algorytm $P_k \mathrm{E} K_k \mathrm{E}$ ma postać

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

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

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

E: $$f_n = (x_n, y_n)$$

Jeśli predyktor jest dostatecznie dokładny (tj. jego rząd nie jest mniejszy od rzędu korektora), to dla wystarczająco małych wartości kroku h, tzn. gdy część główna błędu jest dominująca, uzyskanie maksymalnego rzędu metody korektora następuje w algorytmie PK już po jednej iteracji korektora. Jeśli natomiast predyktor jest mniej dokładny, to do uzyskania maksymalnego rzędu potrzeba więcej iteracji korektora.

Implementacja w programie matlab:

```
function [y] = pca4(dx1,dx2,x1,x2,h)
tic;
t=21/h; %liczba kroków
y(:,1) = [x1 x2]; %początkowe dane do wykresu
for i=1:3
    k11=dx1(x1,x2);
    k12=dx2(x1,x2);
    k21=dx1(x1+0.5*h,x2+0.5*h*k11);
    k22=dx2(x1+0.5*h,x2+0.5*h*k12);
    k31=dx1(x1+0.5*h,x2+0.5*h*k21);
    k32=dx2(x1+0.5*h,x2+0.5*h*k22);
    k41=dx1(x1+h,x2+h*k31);
    k42=dx2(x1+h,x2+h*k31);
    x1=x1+(h/6)*(k11+2*k21+2*k31+k41);
    x2=x2+(h/6)*(k12+2*k22+2*k32+k42);
    y(:,i+1)=[x1 x2]; %zwiększenie bazy danych do wykresu
end
for i = 4:t
    %etap predykcji
    tmp1 = x1 + (h/24)*55*dx1(x1,x2) - 59*(h/24)*dx1(y(1,i-1),y(2,i-1)) +
37*(h/24)*dx1(y(1,i-2),y(2,i-2)) - 9*(h/24)*dx1(y(1,i-3),y(2,i-3));
    tmp2 = x2 + (h/24)*55*dx2(x1,x2) - 59*(h/24)*dx2(y(1,i-1),y(2,i-1)) +
37*(h/24)*dx2(y(1,i-2),y(2,i-2)) - 9*(h/24)*dx2(y(1,i-3),y(2,i-3));
    %etap korekcji
    x1 = x1 + (h/720)*646*dx1(x1,x2) - 264*(h/720)*dx1(y(1,i-1),y(2,i-1)) +
106*(h/720)*dx1(y(1,i-2),y(2,i-2)) - 19*(h/720)*dx1(y(1,i-3),y(2,i-3)) +
h*(251/720)*dx1(tmp1, tmp2);
    x2 = x2 + (h/720)*646*dx2(x1,x2) - 264*(h/720)*dx2(y(1,i-1),y(2,i-1)) +
106*(h/720)*dx2(y(1,i-2),y(2,i-2)) - 19*(h/720)*dx2(y(1,i-3),y(2,i-3)) +
h*(251/720)*dx2(tmp1, tmp2);
    y(:,i)=[x1 x2]; %wstawienie do bazy danych do wykresu
end
toc;
end
```

Metoda Rungego-Kutty ze zmiennym krokiem

Do zmniejszenia liczby iteracji w tych momentach w których nie jest potrzeba ich duża ilość wykorzystuje się zmienną długość kroku dla metody RK4. Polega ona na obliczaniu błędu aproksymacji w każdej iteracji, a następnie na jego podstawie obliczany jest współczynnik przez który mnożony jest poprzedni krok. Jeśli okazuje się, ze obecny krok nie jest wystarczająco dokładny, żeby osiągnąć wynik z zadaną dokładnością należy powtórzyć iterację z pomniejszonym o wyliczony współczynnik krokiem. W celu oszacowania błędu, oprócz kroku o długości h wykonujemy również 2 kroki o dwa razy mniejszej długości.

Wprowadzając oznaczenia $y_n^{(1)}$ – punkt uzyskany po kroku o długości h, oraz $y_n^{(2)}$ – punkt uzyskany po 2 krokach o długości 0.5 h – ze wzoru na błąd

oszacowania otrzymujemy (po przekształceniach) ostateczny wzór na oszacowanie błędu pojedynczego kroku o długościach h oraz 0.5 h:

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

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

Następnie współczynnik modyfikacji kroku:

$$\alpha = \left(\frac{\varepsilon}{|\delta_n(h)|}\right)^{\frac{1}{p+1}}, \qquad \text{gdzie } \varepsilon = |y_n|\varepsilon_w + \varepsilon_b$$

$\varepsilon_w, \varepsilon_b$,czyli wartość względna i bezwzględna to zadane wartości, przyjąłem za nie kolejno 0.1 i 0.001. Proponowana korekta długości kroku wynosi:

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

Wartość s jest to współczynnik bezpieczeństwa, którego wartość w RK4 wynosi 0.9.

Definicje i wzory na podstawie książki Metody numeryczne prof. P. Tatjewskiego; OWPW, Warszawa 2013

Implementacja w programie matlab:

```
function [y] = rk4z(dx1, dx2, x1, x2, h, epsw, epsb, hmax, hmin, s)
tic;
t=21/h;
y(:,1) = [x1 x2];
yd(:,1) = [x1 x2];
for i=1:t
    %pełny krok
    k11=dx1(x1,x2);
    k12=dx2(x1,x2);
    k21=dx1(x1+0.5*h,x2+0.5*h*k11);
    k22=dx2(x1+0.5*h,x2+0.5*h*k12);
    k31=dx1(x1+0.5*h,x2+0.5*h*k21);
    k32=dx2(x1+0.5*h,x2+0.5*h*k22);
    k41=dx1(x1+h,x2+h*k31);
    k42=dx2(x1+h,x2+h*k31);
    xfull1=x1+(1/6)*h*(k11+2*k21+2*k31+k41);
    xfull2=x2+(1/6)*h*(k12+2*k22+2*k32+k42);
    y(:,i+1) = [xfull1 xfull2];
    %pierwszy półkrok
    h2 = h/2;
    khf21=dx1(x1+0.5*h2,x2+0.5*h2*k11);
    khf22=dx2(x1+0.5*h2,x2+0.5*h2*k12);
    khf31=dx1(x1+0.5*h2,x2+0.5*h2*khf21);
    khf32=dx2(x1+0.5*h2,x2+0.5*h2*khf22);
    khf41=dx1(x1+h2,x2+h2*khf31);
    khf42=dx2(x1+h2,x2+h2*khf31);
    xhalf1=x1+(1/6)*h2*(k11+2*khf21+2*khf31+khf41);
    xhalf2=x2+(1/6)*h2*(k12+2*khf22+2*khf32+khf42);
    %drugi półkrok
    khs11 = dx1(xhalf1, xhalf2);
    khs12 = dx2(xhalf1, xhalf2);
```

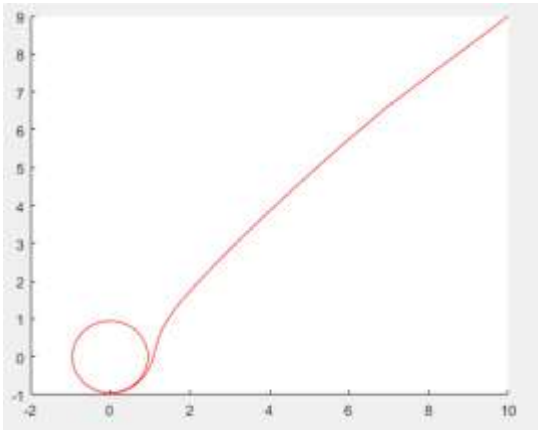
```
    khs21=dx1(x1+h2,x2+h2*k11);
    khs22=dx2(x1+h2,x2+h2*k12);
    khs31=dx1(x1+h2,x2+h2*khs21);
    khs32=dx2(x1+h2,x2+h2*khs22);
    khs41=dx1(x1+h2,x2+h2*khs31);
    khs42=dx2(x1+h2,x2+h2*khs31);
    xhalfsec1=xhalf1+(1/3)*h2*(khs11+2*khs21+2*khs31+khs41);
    xhalfsec2=xhalf2+(1/3)*h2*(khs12+2*khs22+2*khs32+khs42);
    yd(:,i+1) = [xhalfsec1 xhalfsec2];

    d1 = (1/15)*(abs(xhalfsec1-x1));
    d2 = (1/15)*(abs(xhalfsec2-x2));
    e1 = abs(xhalfsec1)*epsw + epsb;
    e2 = abs(xhalfsec2)*epsw + epsb;
    alfa = min((e1/d1)^(1/5), ((e2/d2)^(1/5)));
    h=h*alfa*s;
    if h<hmin
        h=hmin;
    end
    if h>hmax
        h=hmax;
    end
    x1 = xfull1;
    x2 = xfull2;
end
toc
end
```

Wyniki:

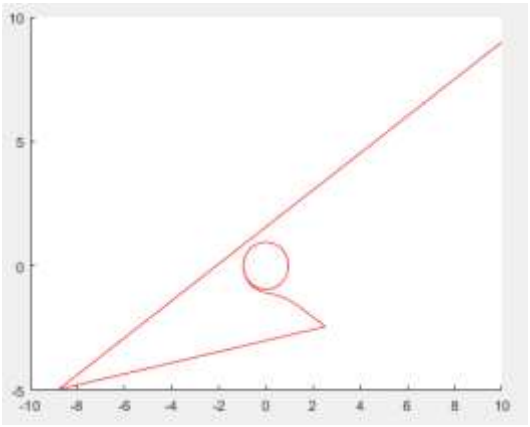
Kroki graniczne dla których zmniejszanie kroku nie zmienia znacząco ilościowych efektów działania algorytmu

a) $x_1 = 10$ $x_2 = 9$

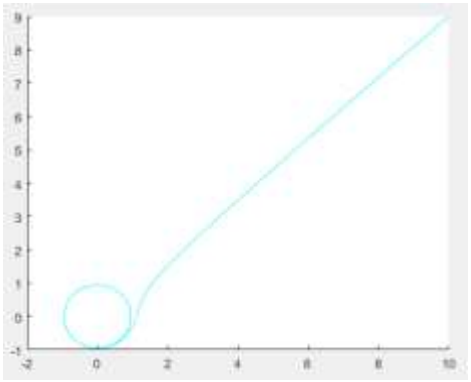
Prawidłowa trajektoria:
h=0.002



Zła trajektoria:
h=0.015

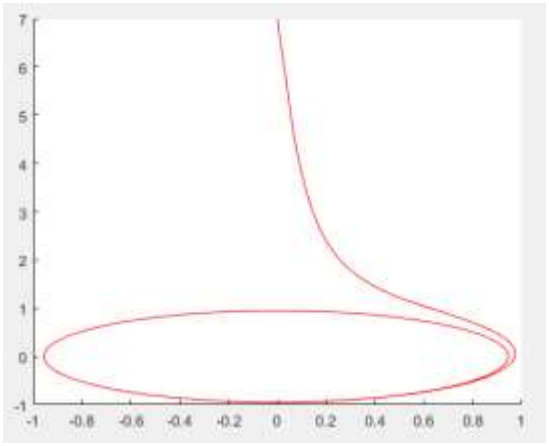


Wykres ode45:

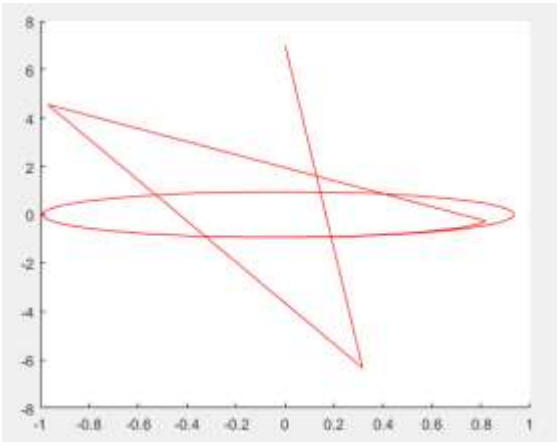


b) $x_1 = 0$ $x_2 = 7$

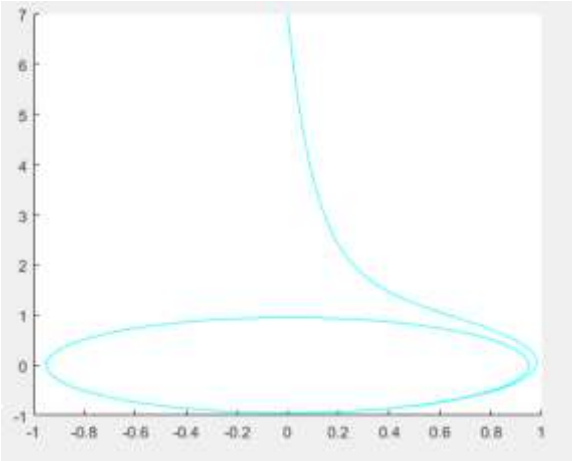
Prawidłowa trajektoria:
h=0.01



Zła trajektoria:
h=0.055



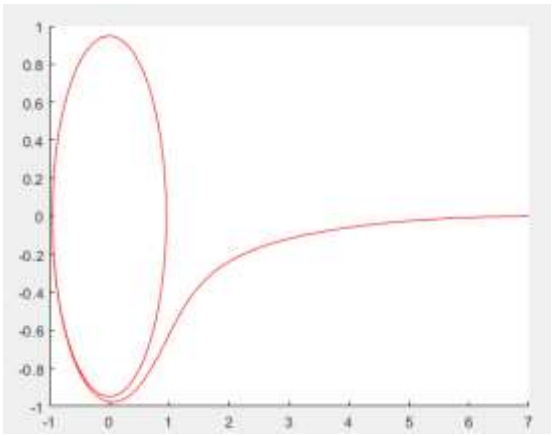
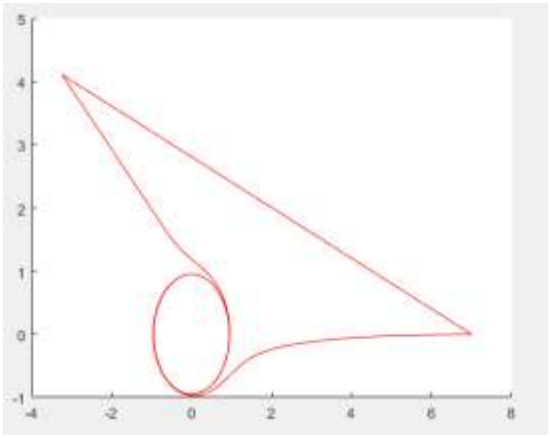
Wykres ode45:



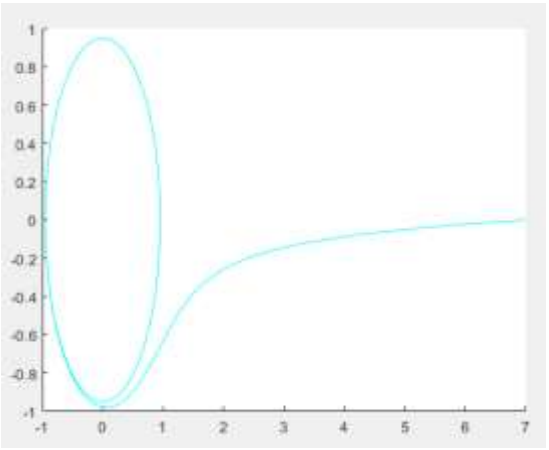
c) $x_1 = 7$ $x_2 = 0$

Prawidłowa trajektoria:
h=0.002

Zła trajektoria:
h=0.02

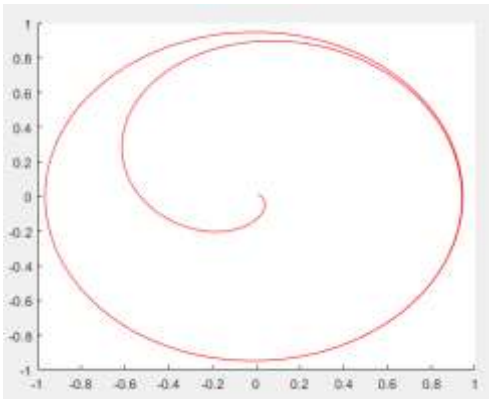
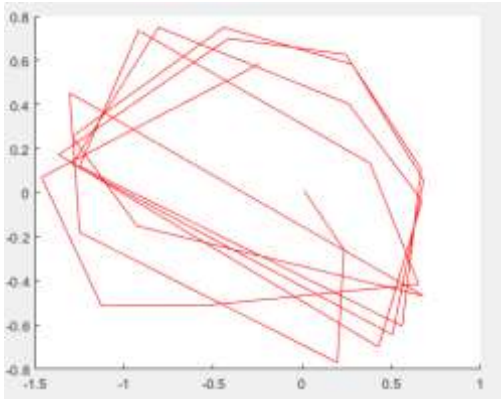
Wykres ode45:



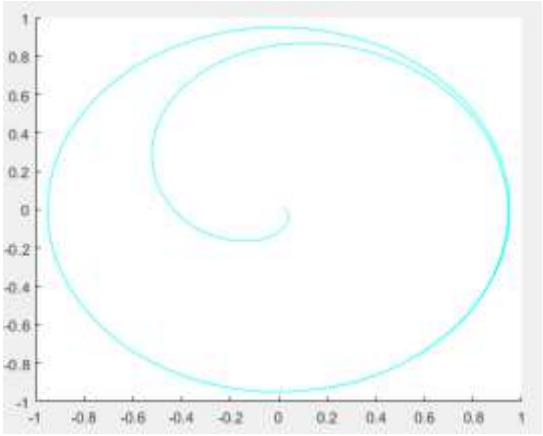
d) $x_1 = 0.001$ $x_2 = 0.001$

Prawidłowa trajektoria:
h=0.02

Zła trajektoria:
h=0.71

Wykres ode45:



Punkty startowe	10,9	0,7	7,0	0.001,0.001
krok	0.002	0.01	0.002	0.02
		czas		
RK4	0.051046	0.005998	0.016758	0.001427
Adams	0.031806	0.011380	0.018391	0.004934
RK4zmienny krok	0.042048	0.013936	0.027460	0.004940
Ode45	0.115397	0.113813	0.114012	0.127022

RK4 ze stałym krokiem:

Możemy zaobserwować jak bardzo dobranie punktu początkowego ma wpływ na otrzymaną trajektorię i na stosunek wielkości błędu do długości kroku. Jeśli punkt startowy jest bliski początkowi układu współrzędnych, możemy używać bardzo dużych kroków (rzędu nawet liczb naturalnych), a jeśli punkt początkowy jest dalej, to konieczne jest zastosowanie mniejszego kroku czyli większe ilości iteracji.

Adams:

Metoda ta daje bardzo zbliżone rezultaty jak metoda RK4 z zadania 1, jednakże zawodzi przy mniejszych długościach kroku niż dzieje się to w przypadku metody RK4.

RK4 ze zmiennym krokiem:

W każdej iteracji dokonywanych jest więcej obliczeń, często wielokrotnie zmieniany jest krok na odpowiadający założonej dokładności. Zaletą tej metody jest fakt automatycznego dobierania kroku w zależności od przebiegu trajektorii, program sam dostosowuje się do warunków funkcji, np. w momentach przegięć funkcji zwiększa swoją dokładność, żeby lepiej ją zaproksymować. Metoda ta odciąża również użytkownika z obowiązku samodzielnego ustalania optymalnego kroku metodą prób i błędów, co jest dużo większą oszczędnością czasu niż ewentualne straty poniesione podczas obliczeń numerycznych.