**MNUM – Projekt 4.56**

**Wojciech Rokicki gr. AR2**

**Zadanie 1**  
Ruch punktu opisany jest równaniami:

Obliczyć przebieg trajektorii ruchu na przedziale dla podanych warunków początkowych, z użyciem metod: Rungego-Kutty czwartego rzędu oraz wielokrokowej predyktor-korektor Adamsa czwartego rzędu ze stałym krokiem.

**Metoda RK4:**

Metodę tę można zdefiniować następująco:

Współczynnik jest pochodną rozwiązania w punkcie . Wartość wyznaczamy jak w zmodyfikowanej metodzie Eulera - jako pochodną rozwiązania wyznaczanego zwykłą metodą Eulera w punkcie (środkowym przedziału). Następnie, wartość wyznaczamy podobnie, jak , ale tym razem w punkcie , tzn. startując z początku przedziału z nachyleniem . W końcu, z nachyleniem tej stycznej startujemy z punktu początkowego do punktu , tzn. wyznaczamy pochodną rozwiązania w punkcie . Mamy w ten sposób wyznaczone 4 wartości pochodnej rozwiązania: po jednej na końcach przedziału i dwie w jego środku. Aproksymacja pochodnej dla pełnego kroku metody wyznaczana jest jako ważona średnia arytmetyczna tych wartości, z wagami 1 na krańcach i wagą 2 w punkcie środkowym.

Krok był zmniejszany do momentu aż wykres prezentował wystarczającą dokładność (gładkość). Błąd pojedynczego kroku był szacowany na podstawie wzoru:

Gdzie:

– nowy punkt uzyskany w kroku o długości

– nowy punkt wyznaczony przez dwa dodatkowe kroki o długościach

– rząd metody.

Kod algorytmu:

function[y] = rk4()

h = 0.002; %krok

x1=8; %pierwszy pkt

x2=7; %drugi pkt

%% 8 7; 0 0.4; 5 0; 0.01 0.001

tic;

t=0:h:20; %ilosc kroków

y(:,1) = [x1 x2]; %dane do wykresu

for i=1:(length(t)-1)

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+k41+2\*(k21+k31));

x2=x2+(h/6)\*(k12+k42+2\*(k22+k32));

y(:,i+1)=[x1 x2]; %zapisanie danych

end

plot(y(1,:),y(2,:));

toc;

end

**Metoda predyktor-korektor Adamsa czwartego rzędu:**

Metody Adamsa:

Równanie różniczkowe

Równoważne jest równaniu całkowemu

Metody Adamsa dostajemy, rozważając to równanie na przedziale :

Metody jawne:

Funkcję podcałkową przybliżamy wielomianem interpolacyjnym stopnia co najwyżej k-1 opartym na węzłach . Przyjmując przybliżenie Lagrange’a mamy:

Gdzie to wielomiany Lagrange’a:

Stąd po scałkowaniu, przy założeniu otrzymujemy:

Wartości są stablicowane.

Metody niejawne:

Funkcję podcałkową przybliżamy wielomianem interpolacyjnym stopnia co najwyżej k opartym na węzłach , z wartościami rozwiązania . Następnie postępując tak jak w przypadku metody jawnej otrzymujemy

Wartości są stablicowane.

Metoda predyktor-korektor:

Realizacja metody PK polega na połączeniu metod jawnych i niejawnych w jeden algorytm. W naszym przypadku:

P:

E:

K:

E:

Dla metody PK oszacowanie błędu:

Kod algorytmu:

function [y] = pc()

tic;

h = 0.005; %krok

x1=8; %pierwszy pkt

x2=7; %drugi pkt

t=0:h:20; %ilosc kroków

y(:,1) = [x1 x2]; %zapis pierwszych danych 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+k41+2\*(k21+k31));

x2=x2+(h/6)\*(k12+k42+2\*(k22+k32));

y(:,i+1)=[x1 x2]; %zapisanie danych

end

for i = 4:(length(t))

%predykcja z ewaluacja

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));

%korekcja z ewaluacja

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];

end

plot(y(1,:),y(2,:));

%plot(0:h:20,y(1,:),'-',0:h:20,y(2,:),'-')

toc;

end

**Metoda RK4 ze zmiennym krokiem:**

Aby zmniejszyć liczbę 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 – punkt uzyskany po kroku o

długości h, oraz – punkt uzyskany po 2 krokach o długości 0.5h – ze wzoru na błąd

oszacowania otrzymujemy (po przekształceniach) ostateczny wzór na oszacowanie błędu

pojedynczego kroku o długościach h oraz 0.5h:

Mówimy, że funkcja jest rzędu p, gdy wartość rn(0) oraz wszystkie jego pochodne od 1 do

p-tej włącznie w punkcie 0, są równe 0, natomiast pochodna p+1 rzędu w punkcie 0 jest

różna od zera.

Kod algorytmu:

function [] = rk4zmien(xs1,xs2,step)

tic

stepd=step/2; %krok o dlugosci 2 razy mniejszej

i=int32(15/stepd); %ustalamy liczbe kroków, startowy przedzial wynosi [0 15]

Error1=zeros(int32(i/2)+1,1); %wektory na bledy aproksymacji

Error2=zeros(int32(i/2)+1,1);

X1=zeros(int32(i/2)+1,1); %wektor na rozwiazania pierwszej zmiennej

X2=zeros(int32(i/2)+1,1); %jw. na druga zmienna

Y=zeros(int32(i/2)+1,1); %wektor ze zmienna czasu

x1=xs1; %x1,x2,x1d,x1d - punkty które bedziemy obliczac dla pelnego

x2=xs2; %i polowicznego kroku

x1d=xs1;

x2d=xs2;

X1(1,1)=x1;

X2(1,1)=x2;

Y (1,1)=0;

Error1(1,1)=xs1;

Error2(1,1)=xs2;

halfstep=step/2; %"pólkroki" potrzebne w algorytmie

halfstepd=stepd/2;

for n=1:i

if(mod(n,2) == 0)

k11=dx1(x1,x2);

k12=dx2(x1,x2);

k21=dx1((x1+halfstep\*k11),(x2+halfstep\*k12));

k22=dx2((x1+halfstep\*k11),(x2+halfstep\*k12));

k31=dx1((x1+halfstep\*k21),(x2+halfstep\*k22));

k32=dx2((x1+halfstep\*k21),(x2+halfstep\*k22));

k41=dx1((x1+step\*k31),(x2+step\*k32));

k42=dx2((x1+step\*k31),(x2+step\*k32));

x1=x1+(1/6)\*step\*(k11+2\*k21+2\*k31+k41);

x2=x2+(1/6)\*step\*(k12+2\*k22+2\*k32+k42);

X1((n/2)+1,1)=x1;

X2((n/2)+1,1)=x2;

end

k11d=dx1(x1d,x2d);

k12d=dx2(x1d,x2d);

k21d=dx1((x1d+halfstepd\*k11d),(x2d+halfstepd\*k12d));

k22d=dx2((x1d+halfstepd\*k11d),(x2d+halfstepd\*k12d));

k31d=dx1((x1d+halfstepd\*k21d),(x2d+halfstepd\*k22d));

k32d=dx2((x1d+halfstepd\*k21d),(x2d+halfstepd\*k22d));

k41d=dx1((x1d+stepd\*k31d),(x2d+stepd\*k32d));

k42d=dx2((x1d+stepd\*k31d),(x2d+stepd\*k32d));

x1d=x1d+(1/6)\*stepd\*(k11d+2\*k21d+2\*k31d+k41d);

x2d=x2d+(1/6)\*stepd\*(k12d+2\*k22d+2\*k32d+k42d);

if(mod(n,2) == 0)

Error1((n/2)+1,1)=(16/15)\*abs(x1d-X1((n/2)+1,1)); %15 to 2^k-1 , gdzie k to stopien (czyli 4)

Error2((n/2)+1,1)=(16/15)\*abs(x2d-X2((n/2)+1,1));

Y ((n/2)+1,1)= double((n/2)+1)\*step;

end

end

toc;

%plot(Y,Error1,'--',Y,Error2,'-');

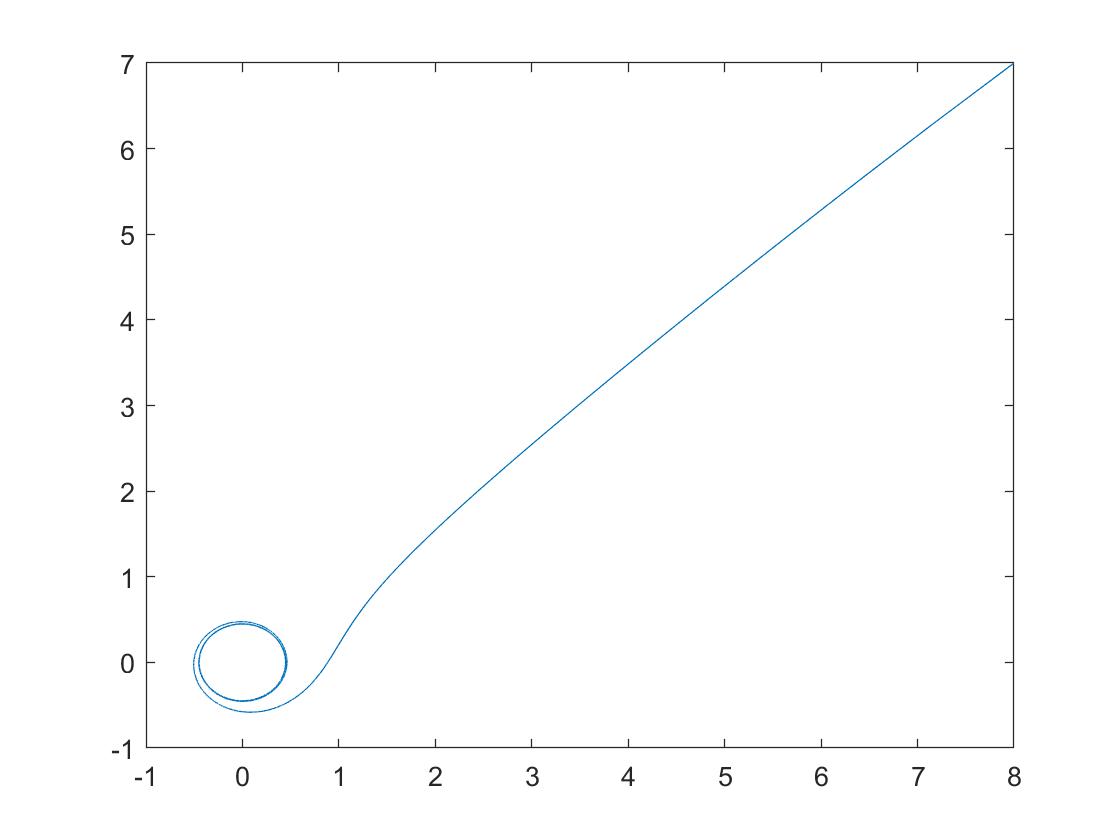
axis([0 15 0 0.2]); %ograniczenie skali x-ów i y-ów

%plot(Y,X1,'--',Y,X2,'-');

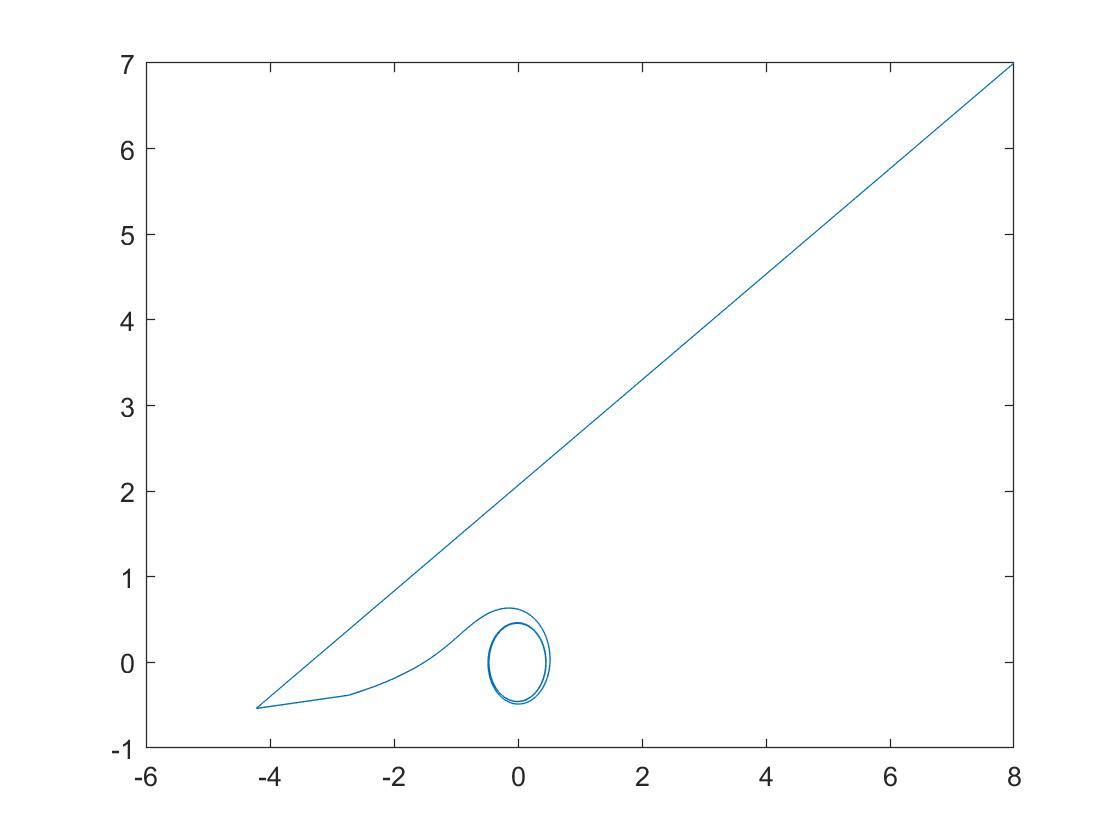
plot(X1,X2);

end

W tym przypadku krok, dla którego metoda RK4 dawała poprawne rezultaty wynosił 0.001



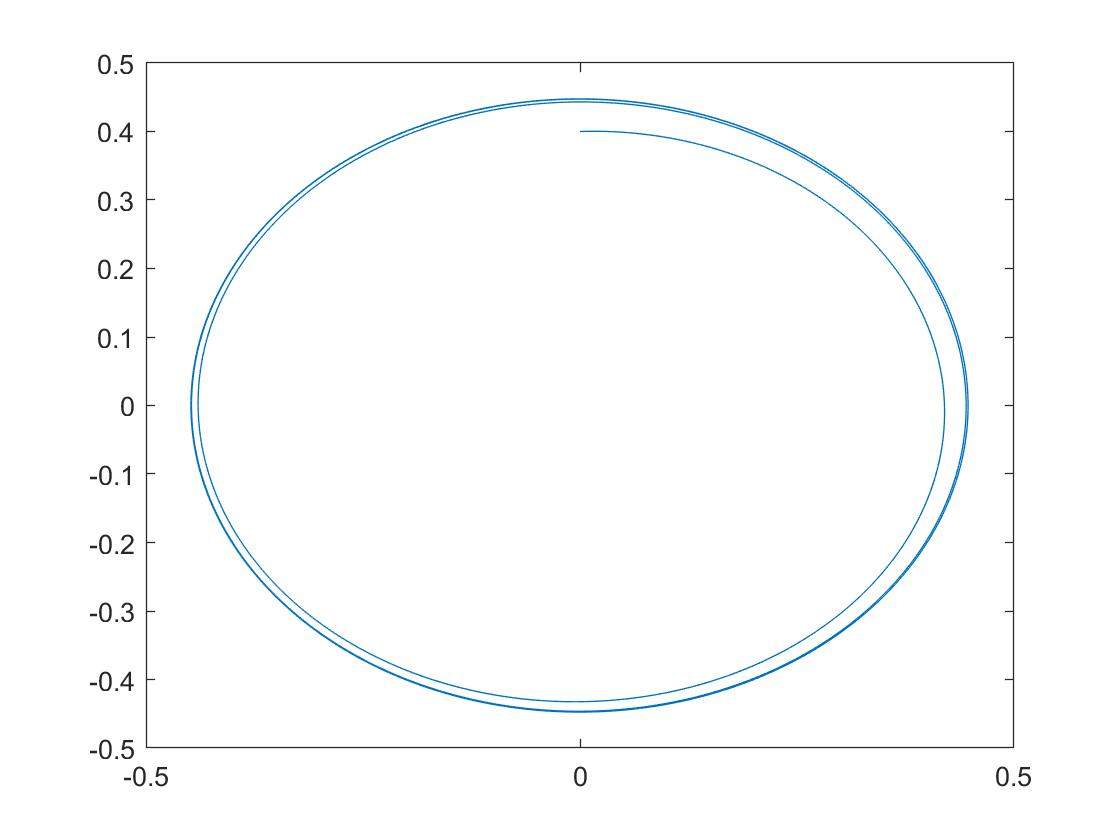
RK4 krok: 0.001, RK4 zmienne krok(granicz.): 0.015



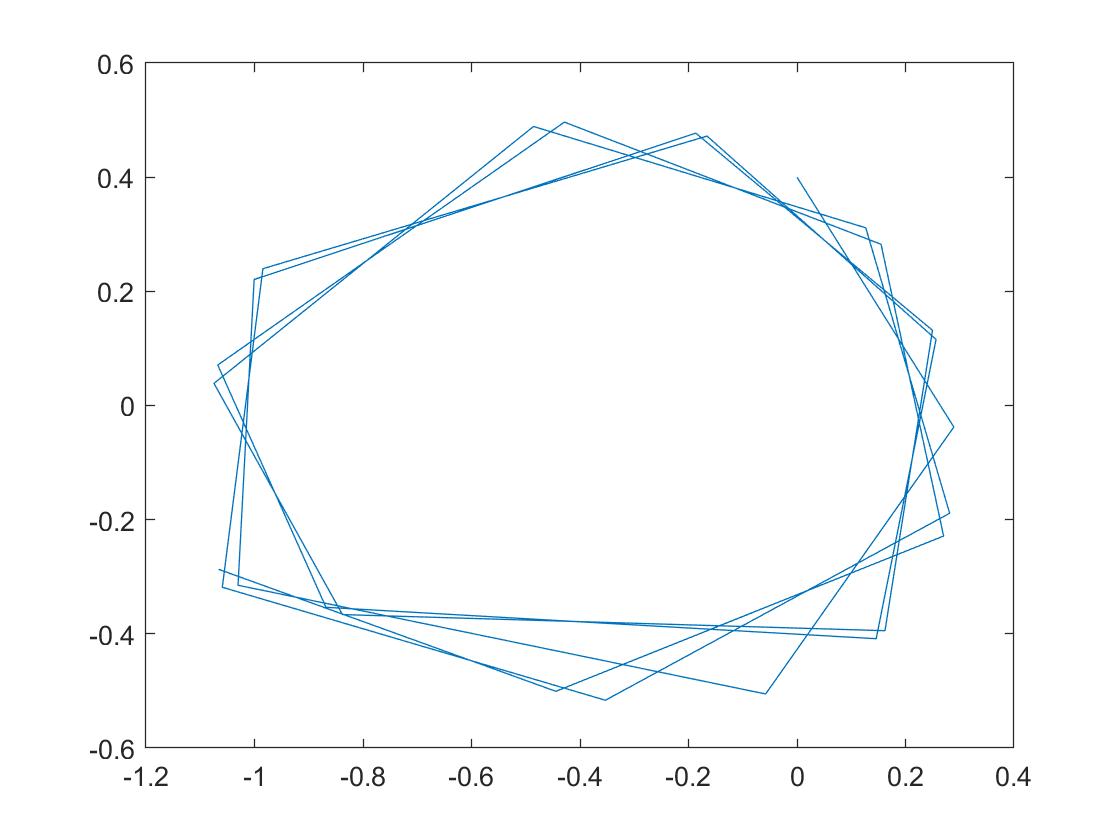
RK4 krok: 0.02

W tym przypadku krok, dla którego metoda RK4 dawała poprawne rezultaty wynosił 0.2

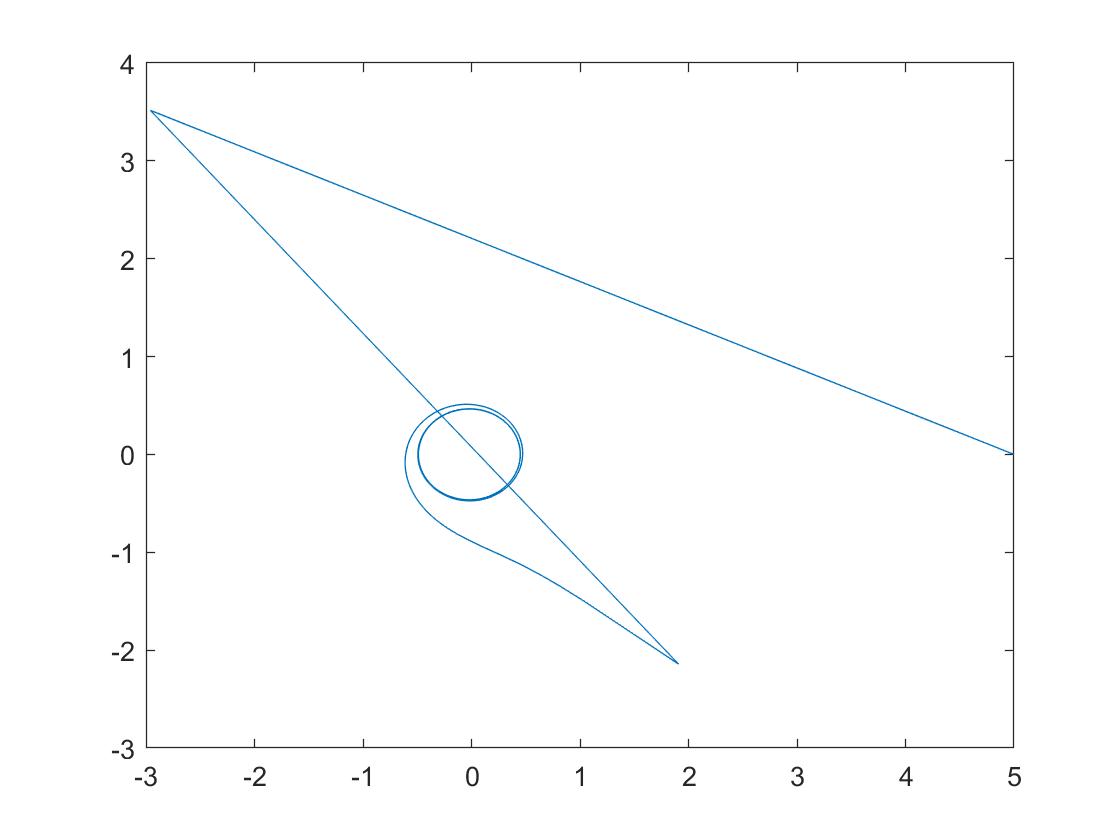
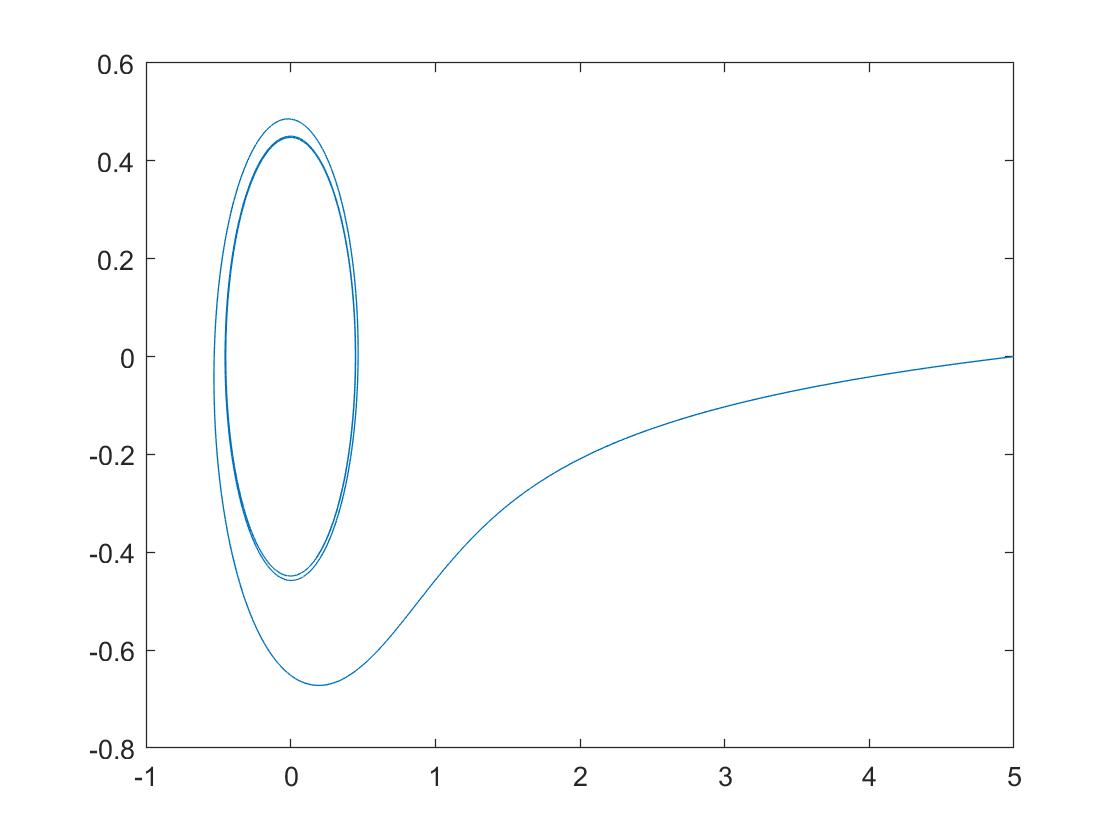
RK4 krok: 0.2, RK4 zmienne krok(granicz.): 0.3



RK4 krok: 0.8

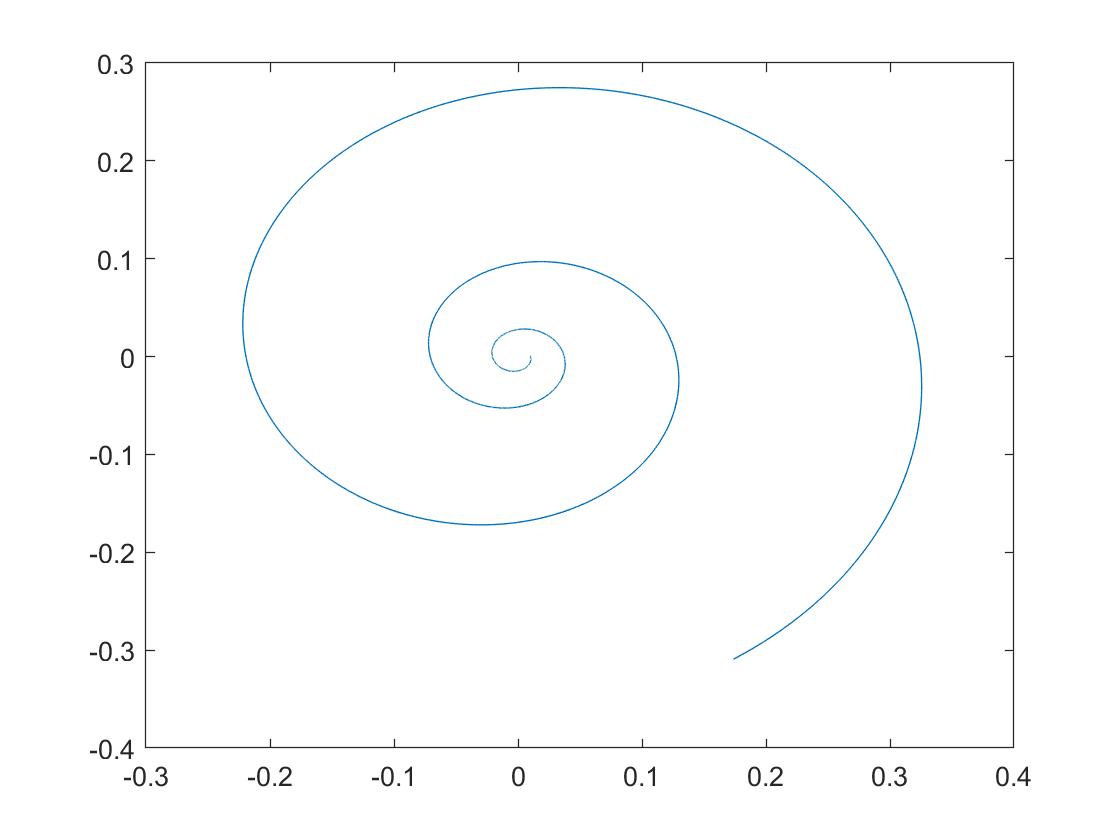


W tym przypadku krok, dla którego metoda dawała poprawne rezultaty wynosił 0.01



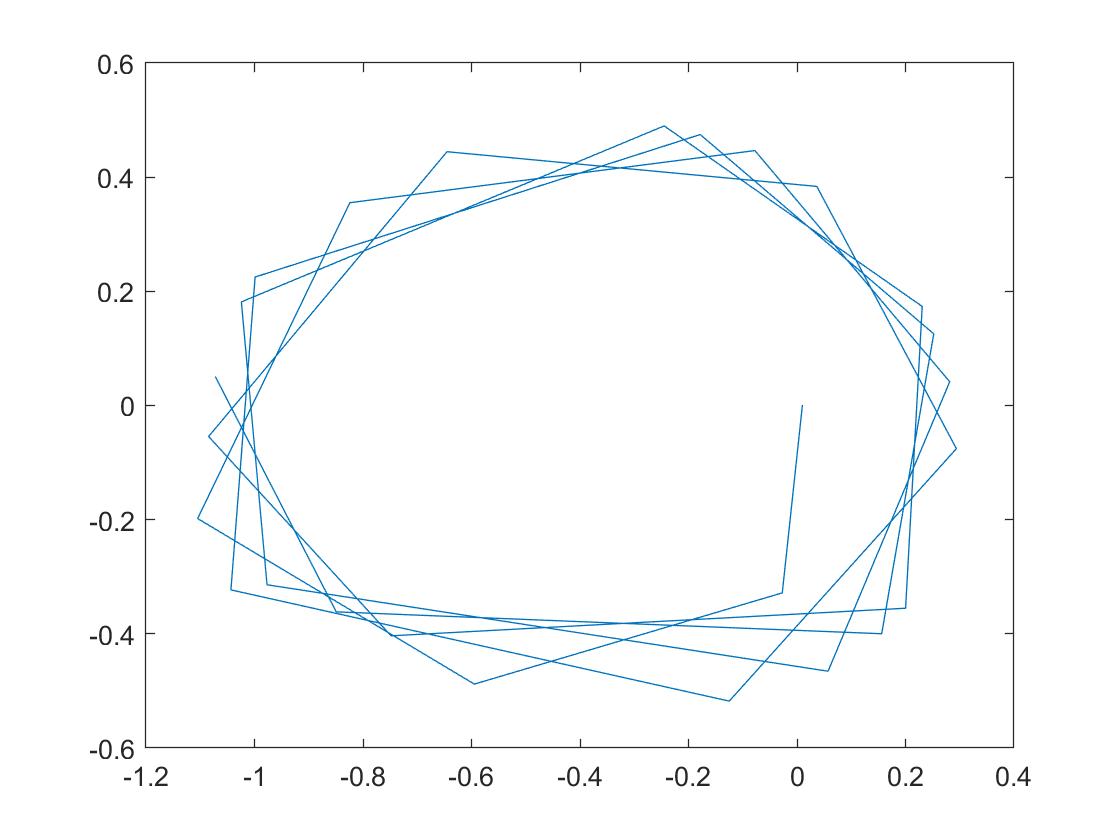
RK4 krok: 0.05

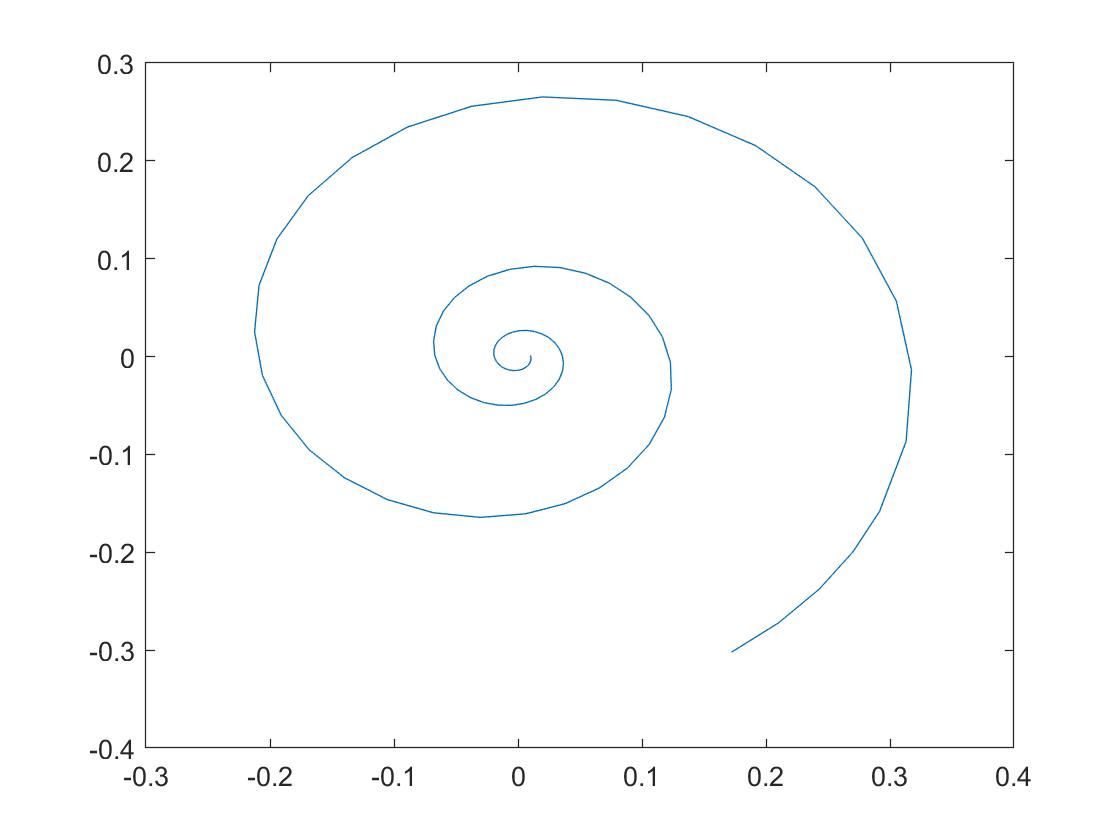
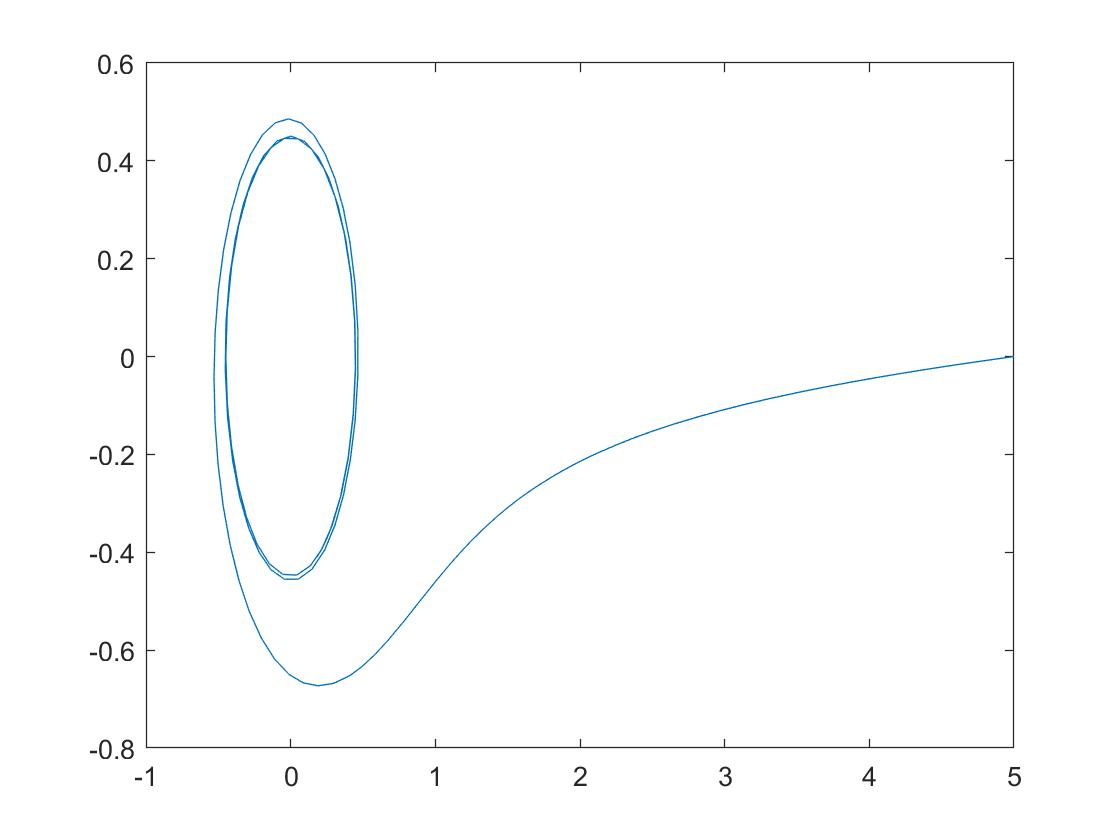
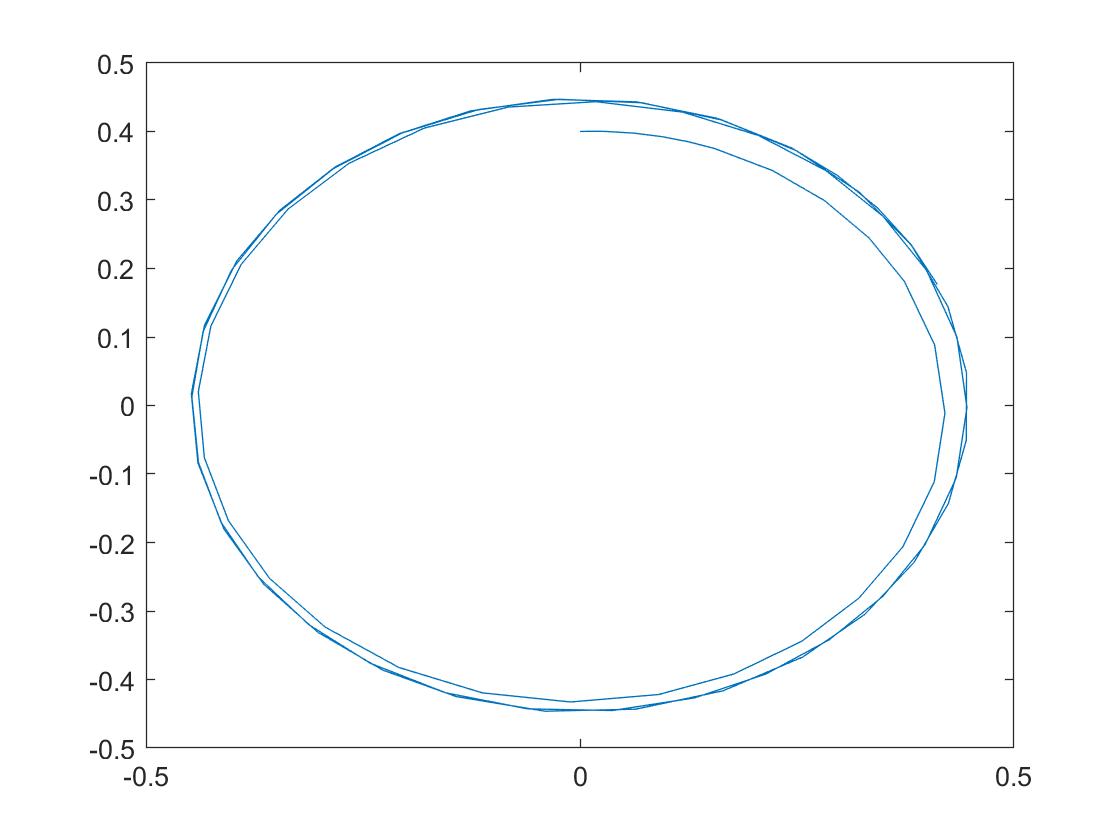
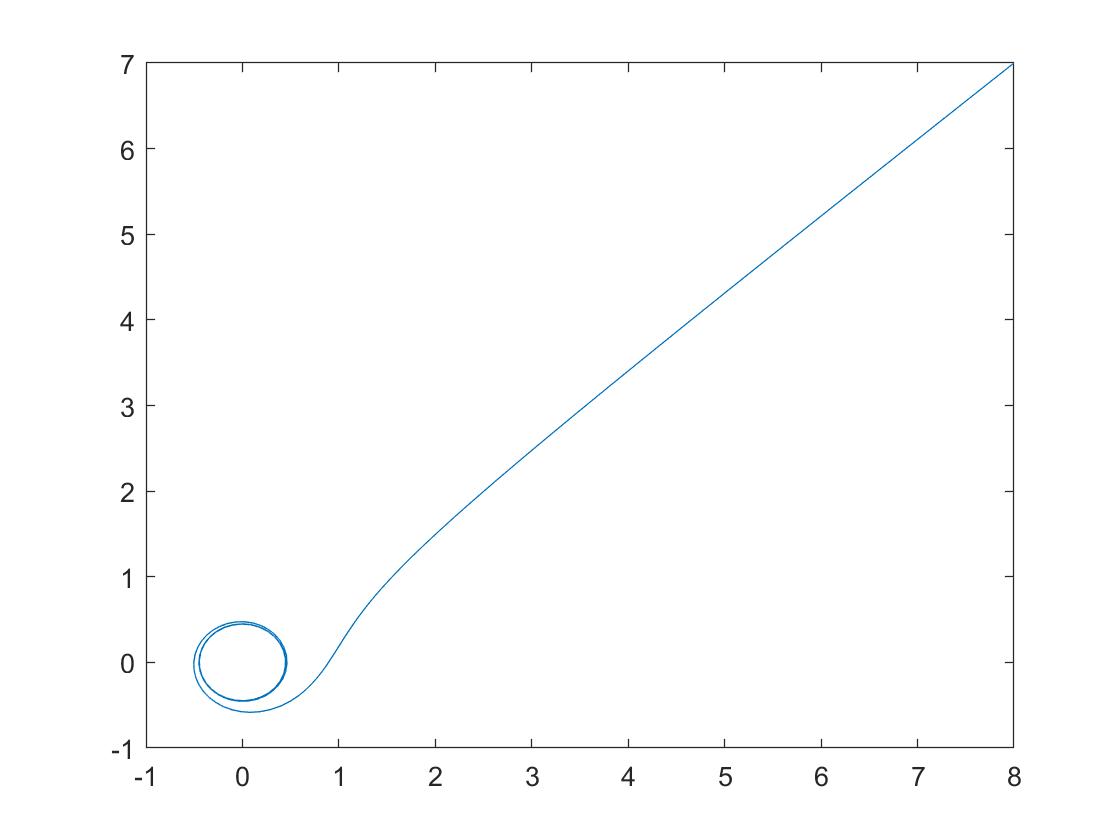
RK4 krok: 0.01, RK4 zmienne krok(granicz.): 0.02

W tym przypadku krok, dla którego metoda dawała poprawne rezultaty wynosił 0.03

RK4 krok: 0.03 RK4 zmienne krok(granicz.): 0.04

RK4 krok: 0.8



**Funkcją ode45:**

**Wnioski:**

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 większej ilości iteracji.

Adams (predyktor/korektor):

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:

Metoda ta znacznie zmniejsza liczbę wykonywanych iteracji, w stosunku do metody

niezmodyfikowanej (krok rośnie – liczba iteracji maleje a co za tym idzie również liczba

związanych z nimi obliczeń i błędów numerycznych).

Jednakże 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.

Porównanie czasów wykonania (w sekundach):

|  |  |  |  |
| --- | --- | --- | --- |
| Podpunkt/Metoda | RK4 | Adams (predyktor/korektor) | RK4 zmienny krok |
| a) | 0,0210 | 0,0132 | 0,0393 |
| b) | 0,0022 | 0,0016 | 0,0031 |
| c) | 0,0182 | 0,0130 | 0,0262 |
| d) | 0,0018 | 0,0013 | 0,0045 |

Jak widać w każdym przypadku metody dawały rezultaty podobne do rezultatów uzyskanych za pomocą polecenia *ode45().* Jednakże we wszystkich przypadkach metoda predyktor-korektor szybciej znajdowała wynik. Ponadto za każdym razem generowała ona mniejsze błędy: w przypadku pierwszych dwóch na samym początku są większe, lecz należy pamiętać że pierwsze cztery punkty w metodzie PK są obliczane za pomocą metody RK4, potem błędy szybko się stabilizują i osiągają bardzo małe wartości. W dwóch ostatnich podpunktach błędy są mniej więcej o rząd wielkości mniejsze. Jednakże metoda PK wymaga większego nakładu obliczeń niż RK4, ze względu na podwójną ewaluację wartości funkcji. Można ją zredukować badając zmienność funkcji i wprowadzając zmienny krok (większy dla mało zmiennych przedziałów i mniejszy dla przedziałów o dużej zmienności).