Sprawozdanie MNUM Projekt 04

Autor: **TOMASZ SACHANOWSKI**

Grupa: **czwartek 8-10**

Nr. Indexu: **276467**

Nr. Zadania**: 4.55**

Spis treści

[Treść zadań 2](#_Toc41159746)

[Cel: 3](#_Toc41159747)

[Teoria: 3](#_Toc41159748)

[Metody jednokrokowe 3](#_Toc41159749)

[Metoda Rungego-Kutty ze stałym krokiem 4](#_Toc41159750)

[Prezentacja wyników - metoda Rungego-Kutty ze stałym krokiem: 6](#_Toc41159751)

[Metoda Rungego-Kutty ze zmiennym krokiem: 9](#_Toc41159752)

[Prezentacja wyników - metoda RK4 ze zmiennym krokiem: 10](#_Toc41159753)

[Metody wielokrokowe 12](#_Toc41159754)

[Metody Adamsa 13](#_Toc41159755)

[Metody jawne (Adamsa-Bashforta) 14](#_Toc41159756)

[Metody niejawne (Adamsa-Moultona) 14](#_Toc41159757)

[Prezentacja wyników - metoda wielokrokowa predyktor-korektor Adamsa: 16](#_Toc41159758)

[Wynik: 19](#_Toc41159759)

[𝑥1(0)=10 𝑥2(0)=8 19](#_Toc41159760)

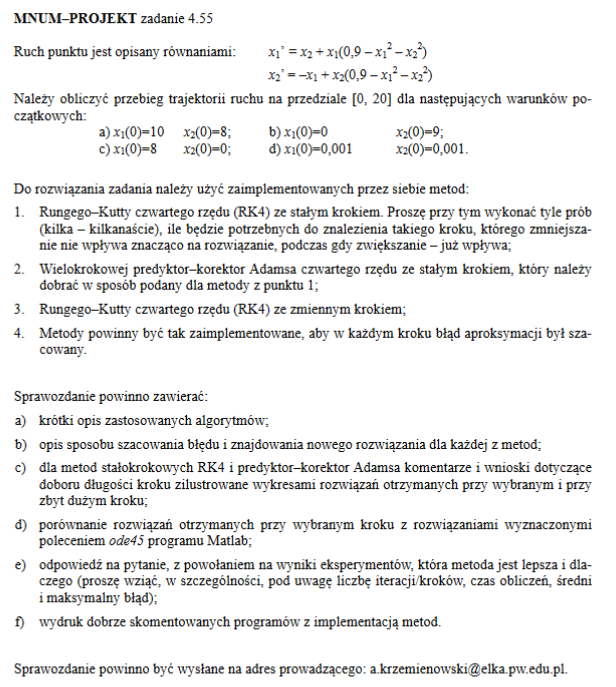
[𝑥1(0)=8 𝑥2(0)=0 19](#_Toc41159761)

[x1(0)=0.001, x2(0)=0.001 19](#_Toc41159762)

[Podsumowanie: 19](#_Toc41159763)

[Dodatek: 20](#_Toc41159764)

# Treść zadań



# Cel:

Celem zadania jest znalezienie przebiegu trajektorii punktu na zadanym przedziale i przy początkowych warunkach.

# Teoria:

Równania różniczkowe służą do modelowania fizycznych układów dynamicznych. Metoda numeryczne są sposobem rozwiązywania tych nieliniowych układów równań.

Wyróżniamy następujące metody numerycznego rozwiązywania równań różniczkowych:

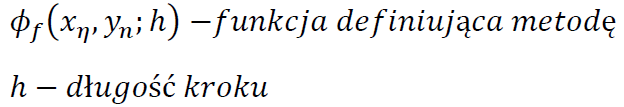
* metody jednokrokowe.
* metody wielokrokowe.

## Metody jednokrokowe

Metody jednokrokowe są zdefiniowane poprzez następujący wzór:



gdzie:



Metoda jest zbieżna gdy:



Jeżeli spełnione są założenia:

* funkcje są ciągłe na zbiorze: 𝐷={(𝑥,𝑦):𝑎≤𝑥≤𝑏,𝑦∈ℝ}
* funkcja spełnia warunki Lipschitza względem y, tzn.:‖𝑓(𝑥,𝑦)−𝑓(𝑥,𝑦̅)‖≤𝐿‖𝑦−𝑦̅‖,

to warunek powyższy jest koniecznym i dostatecznym warunkiem zbieżności metody jednokrokowej.

Metody te służą do rozwiązywania równań różniczkowych zwyczajnych podanych z warunkiem początkowym:

Mówimy, że metoda jest rzędu p, jeśli zachodzą równości:

### Metoda Rungego-Kutty ze stałym krokiem

Jest ona jedną z metod jednokrokowych. Do wykonania jednego kroku metody należy obliczyć wartości prawych stron dokładnie m razy (metoda m-etapowa).

Jeśli przez p(m) oznaczymy maksymalny rząd metody, to udowodniono, że:

𝑝(𝑚)=𝑚 𝑑𝑙𝑎 𝑚=1,2,3,4

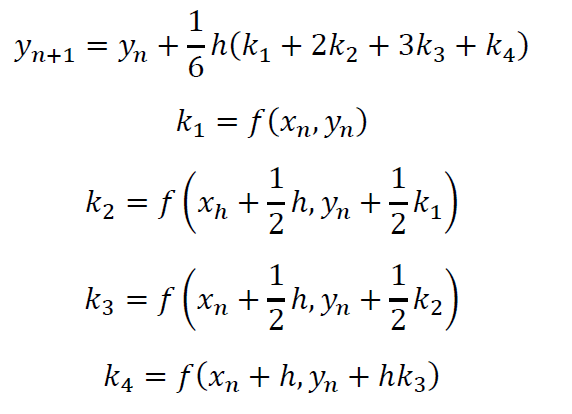
𝑝(𝑚)=𝑚−1 𝑑𝑙𝑎 𝑚=5,6,7

𝑝(𝑚)≤𝑚−2 𝑑𝑙𝑎 𝑚≥8.

Największe znaczenie praktyczne mają metody z 𝑚=4 i rzędu 4 – jest to kompromis między dokładnością (rząd metody) a nakładem obliczeń na jedną iterację i związanym z tym wpływem błędów zaokrągleń.

Jest ona metodą samosterującą, tzn. znajomość́ warunku początkowego wystarcza, by rozpocząć́ obliczenia. Niestety jest to też raczej metoda kosztowna czasowo (wymaga wielokrotnego obliczania wartości funkcji).

Metoda RK4



Gdzie:

x, y – argumenty funkcji f,

h – długość kroku.

Współczynnik jest pochodną rozwiązania w punkcie (,). Wartość pochodną rozwiązania wyznaczanego zwykłą metodą Eulera w punkcie (+ℎ,+) (środkowym przedziału). Z i jest podobnie. W taki sposób mamy wyznaczone cztery wartości pochodnych obliczonych w końcach przedziału i w jego środku. Aproksymacja pochodnej dla pewnego kroku wyznaczana jest jako średnia arytmetyczna tych wartości z wagami 1 dla wartości krańcowych i odpowiednio 2 i 3 dla wartości środkowych.

Wybór kroku jest jednym z trudniejszych zadań w te metodzie. Krok powinien być wystarczający do uzyskania założonej dokładności, lecz nie powinien być znacznie mniejszy przy której wymagana dokładność jest osiągana.

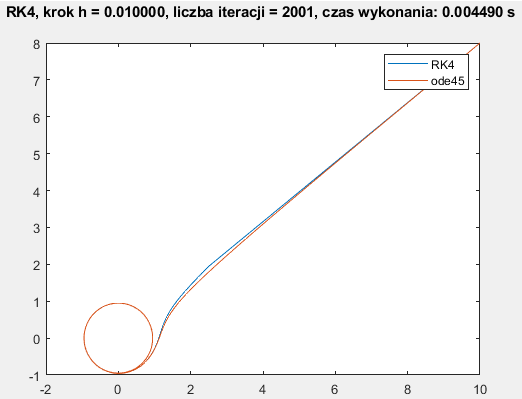
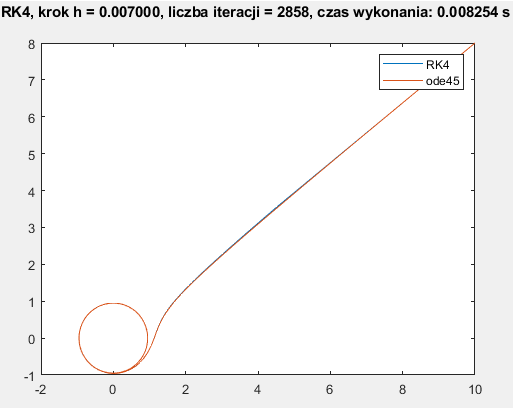
Przy wyznaczaniu długości kroku występują dwie przeciwstawne tendencje:

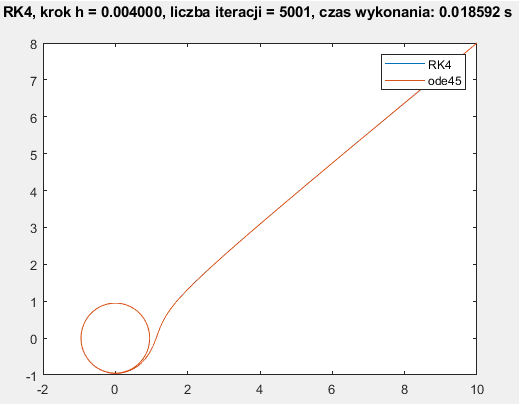
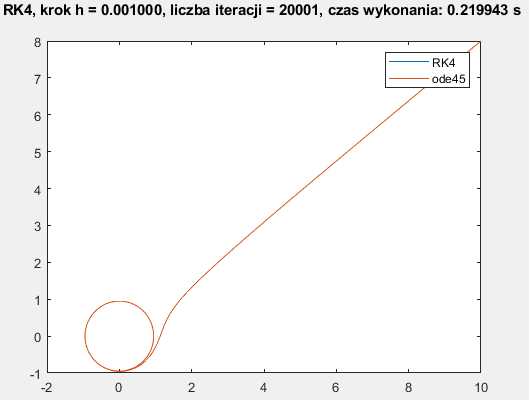
* jeśli krok maleje, to maleje błąd metody (błąd aproksymacji), dla metody zbieżnej błąd maleje do zera przy ℎ dążącym do zera,
* jeśli krok maleje, to zwiększa się liczba iteracji (liczba kroków) potrzebnych do wyznaczenia rozwiązania na zadanym odcinku [𝑎,𝑏], a stąd liczba obliczeń i związanych z nimi błędów numerycznych.

Wybór długości kroku h jest bardzo istotny. Jeśli maleje krok ℎ to maleje błąd metody, ale zwiększa się liczba iteracji, a zatem również wielkość błędów numerycznych. Wynika z tego, że istnieje pewny optymalny krok dla tej metody. Wyznaczyć go można empirycznie. Krok dobrany zostanie tak aby jego zmniejszenie nie wpływało znacząco na rozwiązanie, podczas gdy zwiększenie wpływa.

### Prezentacja wyników - metoda Rungego-Kutty ze stałym krokiem:

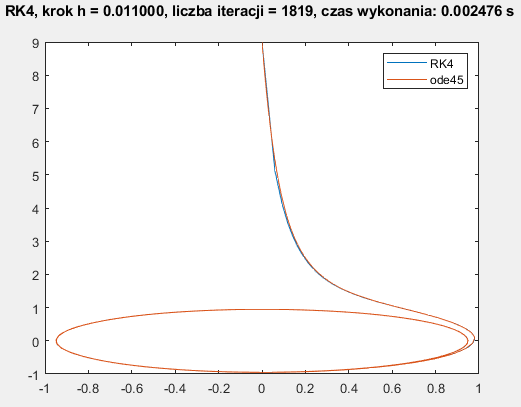
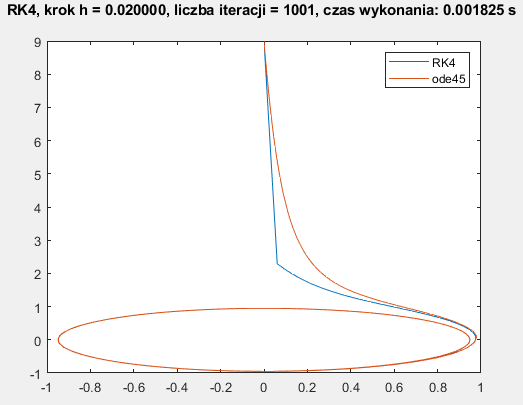
1. 𝑥1(0)=10 𝑥2(0)=8

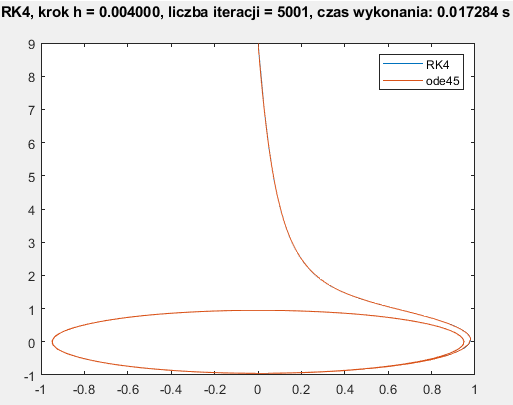
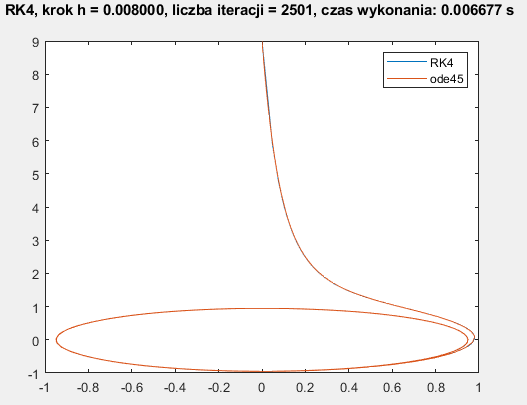
 

**Właściwa trajektoria dla h = 0.007:**

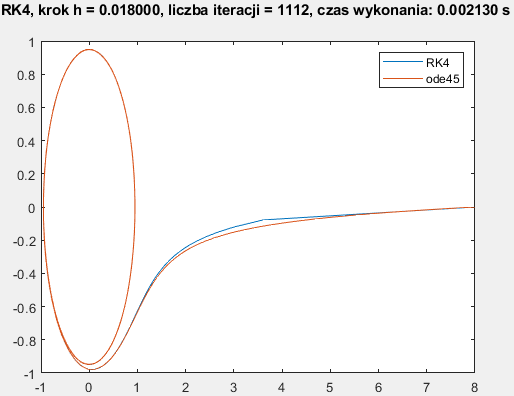
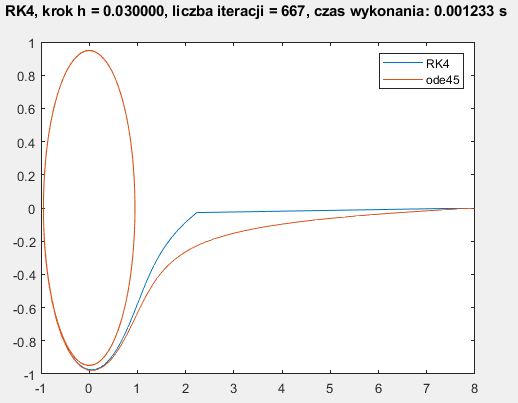
1. 𝑥1(0)=0 𝑥2(0)=9

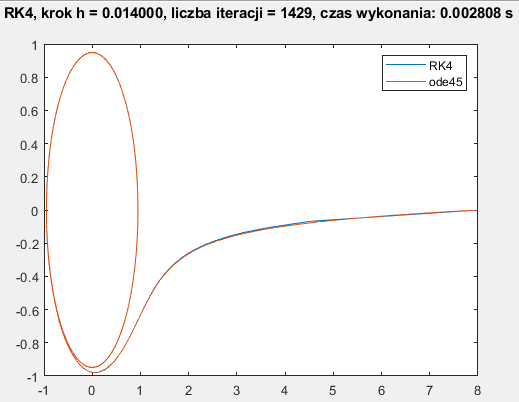
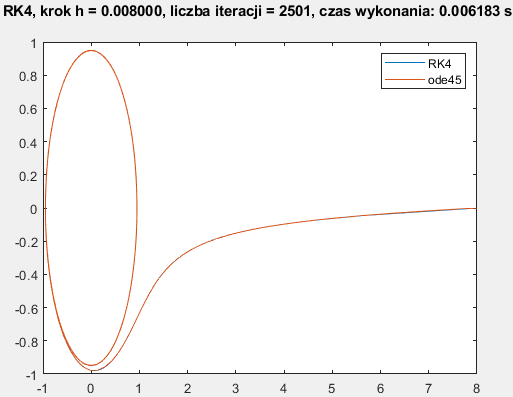




**Właściwa trajektoria dla h = 0.011:**

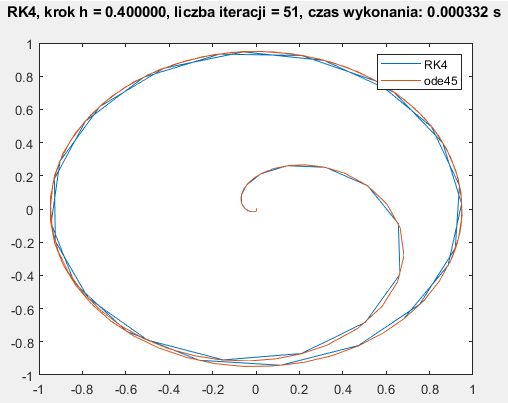
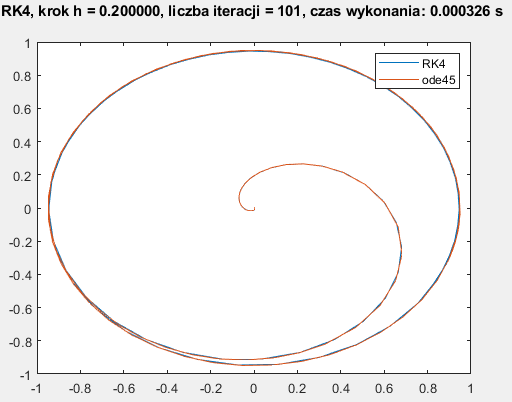
1. 𝑥1(0)=8 𝑥2(0)=0

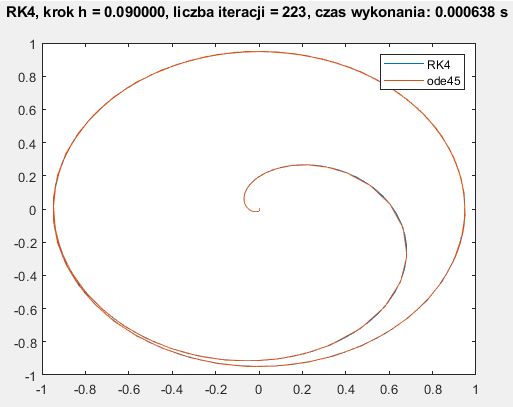
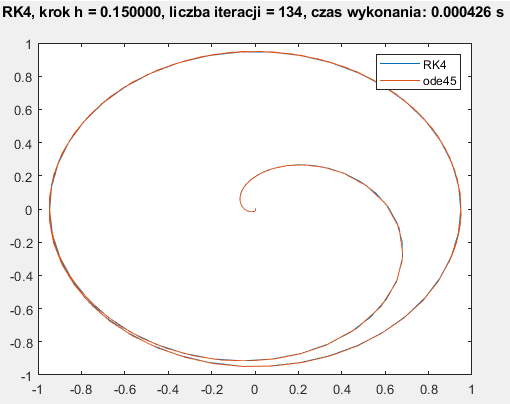


**Właściwa trajektoria dla h = 0.014:**

1. x1(0)=0.001, x2(0)=0.001



**Właściwa trajektoria dla h = 0.015:**

### Metoda Rungego-Kutty ze zmiennym krokiem:

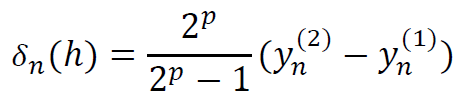
Wiadomo, że zmniejszając krok h, to maleje błąd metody, lecz wówczas zwiększa się liczba iteracji potrzebnych do wyznaczenia rozwiązania na żądanym odcinku, więc i liczba obliczeń i związanych z nimi błędów numerycznych.

W takim razie warto poszukać złotego środka, który będzie korzystał starał się zoptymalizować.

Lokalny błąd metody szacowany jest w każdym kroku. Definiuje się on w następujący sposób:



Przekształcając otrzymujemy:



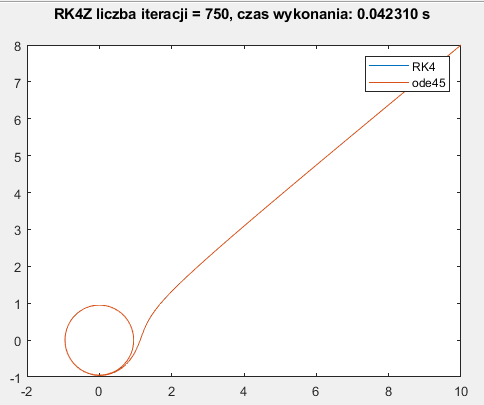
gdzie:

* h – wartość kroku,
* p – rząd metody,
* – nowy punkt wyznaczony metodą z 2 krokami o długości h/2,
* – nowy punkt wyznaczony metodą z krokiem o długości h.

Metoda ta polega na obliczaniu błędu aproksymacji w każdej iteracji. Na jego podstawie obliczany jest współczynnik przez który mnożony jest wcześniejszy krok. Jeśli okazuje się, że obecny krok nie jest wystarczająco dokładny to należy powtórzyć iterację z krokiem pomniejszonym o wyliczony współczynnik.

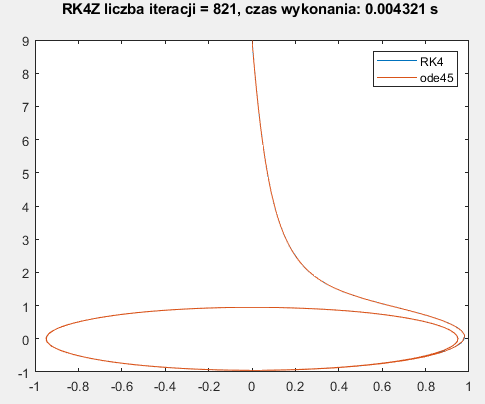
### Prezentacja wyników - metoda RK4 ze zmiennym krokiem:

1. 𝑥1(0)=10 𝑥2(0)=8



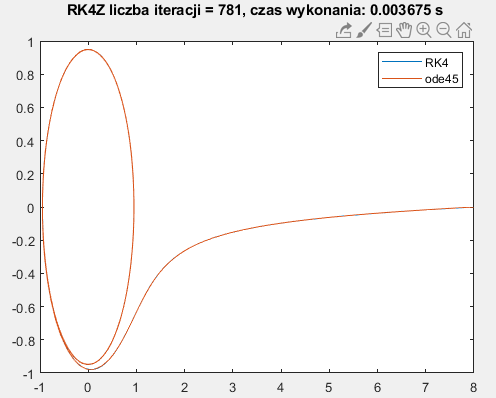
**Właściwa trajektoria dla h = 0.009:**

1. 𝑥1(0)=0 𝑥2(0)=9



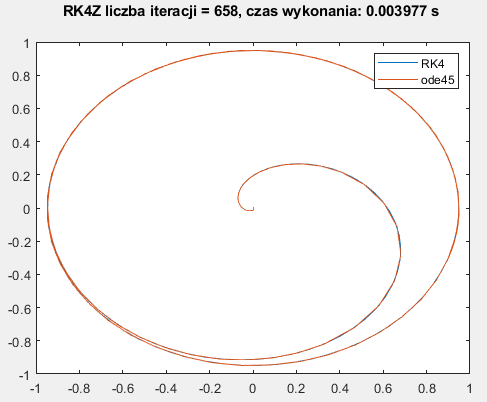
**Właściwa trajektoria dla h = 0.025:**

1. 𝑥1(0)=8 𝑥2(0)=0



**Właściwa trajektoria dla h = 0.014:**

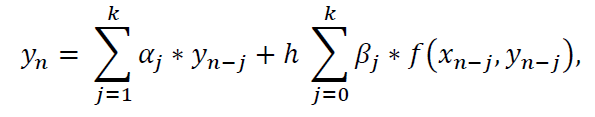
1. x1(0)=0.001, x2(0)=0.001



**Właściwa trajektoria dla h = 0.55**

## Metody wielokrokowe

Ogólna postać wzoru definiującego krok (iterację) metody 𝑘-krokowej liniowej, ze stałą długością kroku ℎ jest następująca:

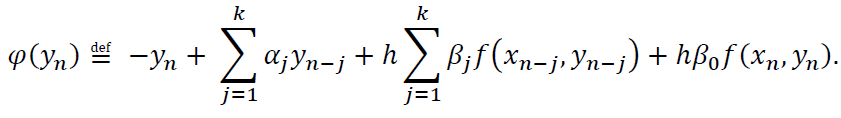


Gdzie:

* =𝑦(=)=
* =+𝑛ℎ
* =𝑎
* 𝑥 ∈[𝑎,𝑏].

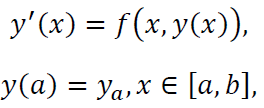
Metoda wielokrokowa jest jawna, jeśli . Dla metody jawnej wartość zależy jawnie od wartości 𝑦 i 𝑓(𝑥, 𝑦) jedynie w poprzednich (już obliczonych) punktach.

Metoda wielokrokowa jest niejawna, jeśli ≠. Dla metody niejawnej wartość 𝑦𝑛 obliczana jest na podstawie 𝑘 poprzednich wartości 𝑦 i 𝑓(𝑥,𝑦) oraz dodatkowo również wartości w punkcie bieżącym =𝑓(,) tj. do wyznaczenia 𝑦𝑛 trzeba w istocie rozwiązać równanie algebraiczne 𝜑 (nieliniowe, jeśli funkcja prawej strony 𝑓 jest nieliniowa), gdzie:

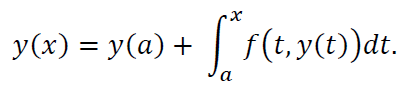


### Metody Adamsa

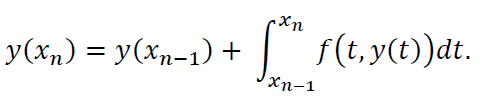
Równanie różniczkowe:



Równoważne jest równaniu całkowemu:

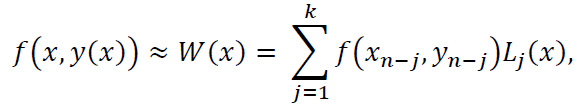


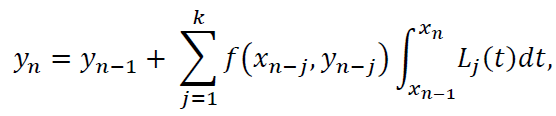
Metody Adamsa dostajemy, rozważając to równanie na przedziale [,]:



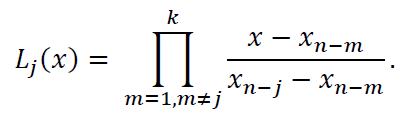
### Metody jawne (Adamsa-Bashforta)

Funkcję podcałkową 𝑓 z powyższego wzoru przybliżamy wielomianem interpolacyjnym 𝑊(𝑥) stopnia co najwyżej 𝑘−1 opartym na węzłach ,…,. Przyjmując przybliżenie 𝑦()≈ i stosując wzór interpolacyjny Lagrange’a mamy:





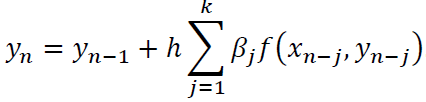
gdzie to wielomiany Lagrange’a,



Stąd po scałkowaniu, przy założeniu:

=−𝑗ℎ, 𝑗=1,2,…,𝑘,

Otrzymujemy:

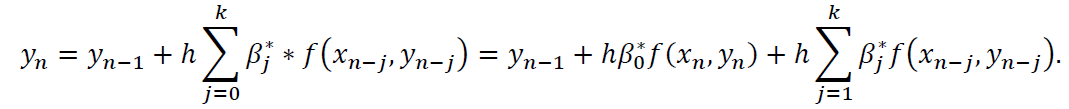


Wartości współczynników 𝛽 odczytujemy z tabeli parametrów metod jawnych Adamsa.

Aby otrzymać rząd metody równy 4 przyjmujemy 𝑘=4.

### Metody niejawne (Adamsa-Moultona)

Funkcję podcałkową przybliżamy wielomianem interpolacyjnym stopnia co najwyżej 𝑘 opartym na węzłach ,,…,, z wartościami rozwiązania 𝑦()≈. Następnie, postępując tak jak w przypadku metod Adamsa-Bashforta, otrzymamy:



Wartości parametrów odczytujemy z tabeli parametrów metod niejawnych Adamsa.

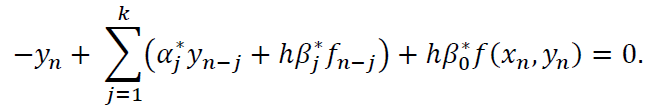
Aby otrzymać rząd metody równy 4 przyjmujemy 𝑘=3.

Metody predyktor – korektor

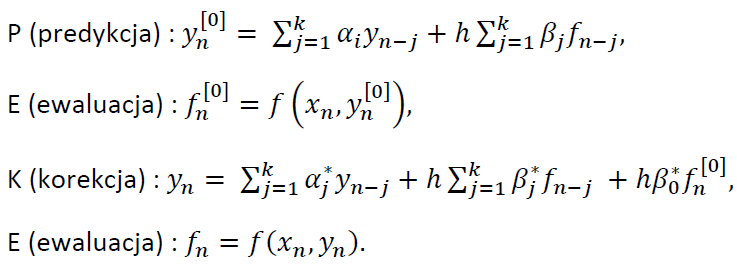
Najpraktyczniejszą metodą wielokrokową byłaby metoda o:

* wysokim rzędzie i małej stałej błędu,
* możliwie dużym obszarze absolutnej stabilności,
* możliwie małej liczbie obliczeń na iterację.

Metody jawne gorzej spełniają dwa pierwsze warunki, natomiast metody niejawne spełniają je znacznie lepiej, ale nie wypełniają warunku trzeciego, gdyż w każdej iteracji trzeba rozwiązywać względem równanie nieliniowe:

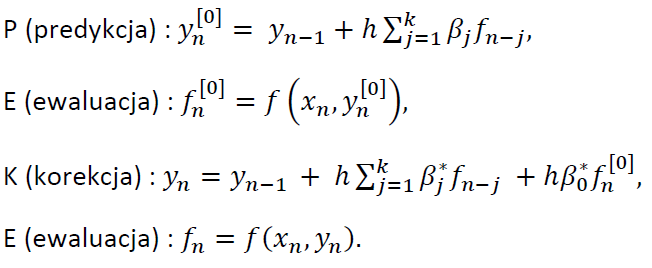


Praktyczne realizacje metod wielokrokowych to algorytmy typu predyktor – korektor (PK) będące połączeniem metod jawnych i niejawnych. Dla metody 𝑘-krokowej realizacja w postaci struktury predyktor-korektor 𝐸 ma postać:



Metoda PK to w istocie przybliżony sposób realizacji metody niejawnej (korektora). Algorytm predyktora gra tu rolę pomocniczą polegającą na efektywnym wyliczeniu dobrego punktu startowego.

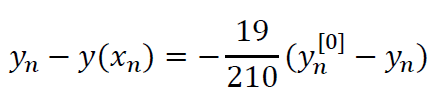
Dla metod Adamsa algorytm 𝐸 ma postać:



Iteracja predyktora ma na celu obliczenie dobrego punktu początkowego dla iteracji korektora rozwiązujemy nieliniowe równanie algebraiczne metody niejawnej korektora.

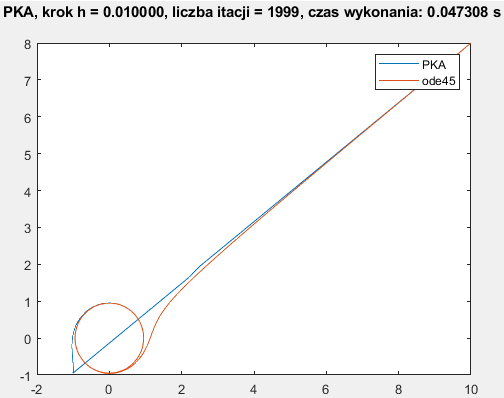
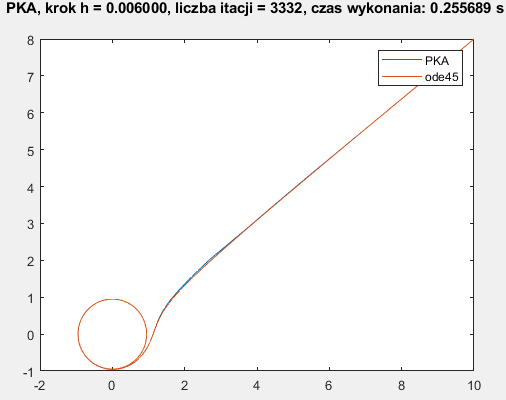
Jeżeli predyktor jest dostatecznie dokładny, to dla dostatecznie małych wartości kroku h, uzyskanie maksymalnego rzędu następuje w algorytmie PK już po jednej iteracji korektora. Natomiast dla mniej dokładnego predyktora potrzebna jest większa ilość iteracji.

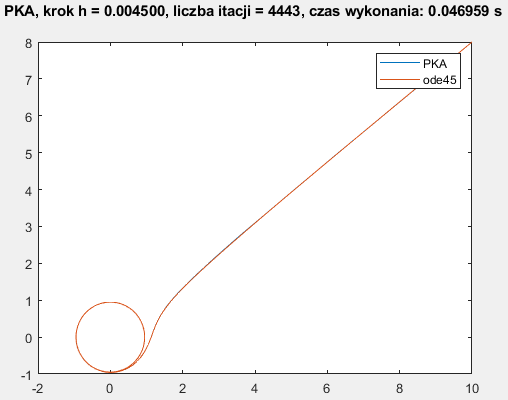
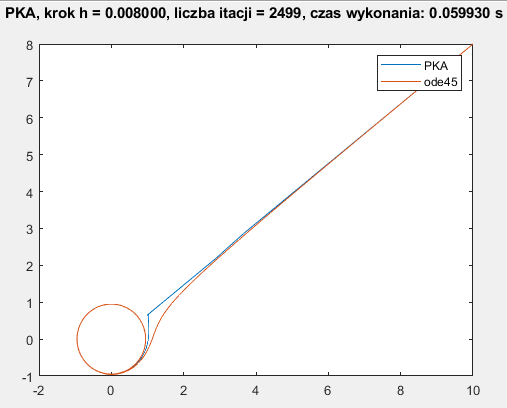
Dla metody PK Adamsa z 4-etapowym predykatorem i 3-etapowym korektorem błąd metody określony jest wzorem:



### Prezentacja wyników - metoda wielokrokowa predyktor-korektor Adamsa:

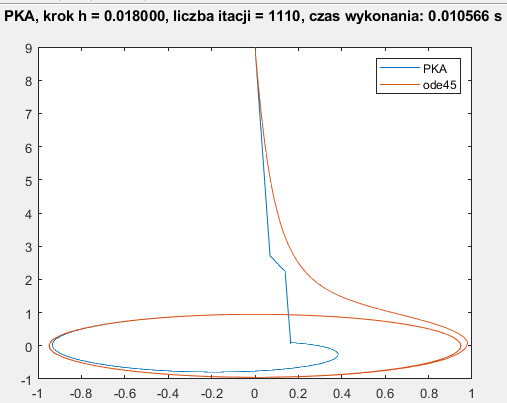
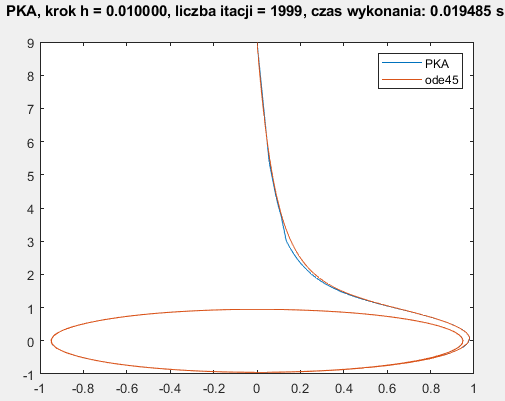
1. 𝑥1(0)=10 𝑥2(0)=8

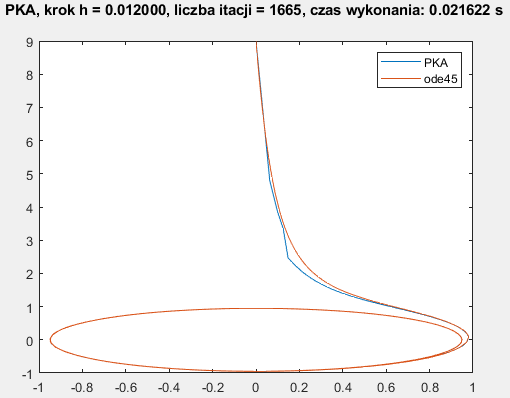
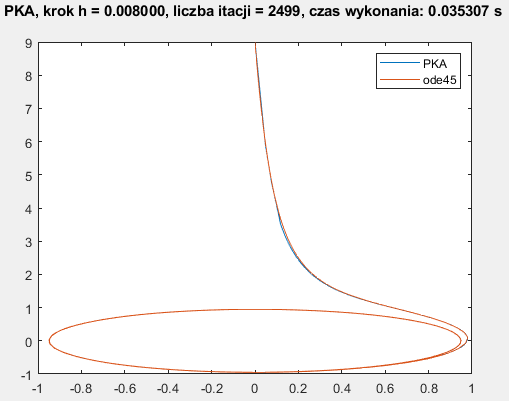




**Właściwa trajektoria dla h = 0.006:**

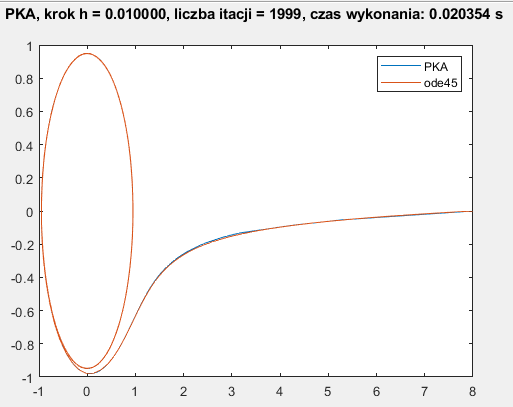
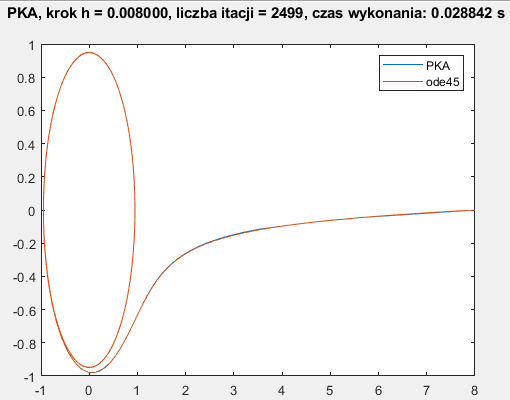
1. 𝑥1(0)=0 𝑥2(0)=9

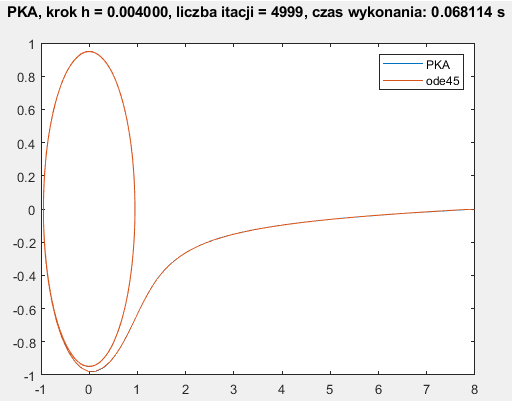
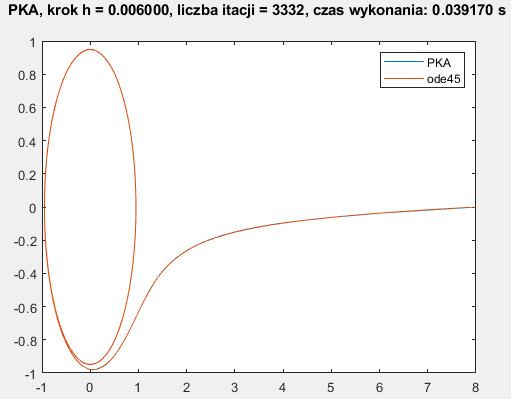


**Właściwa trajektoria dla h = 0.008:**

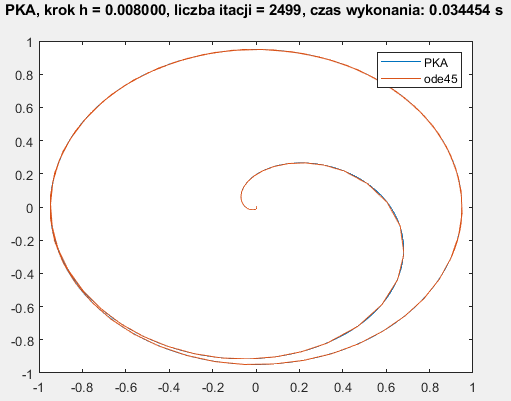
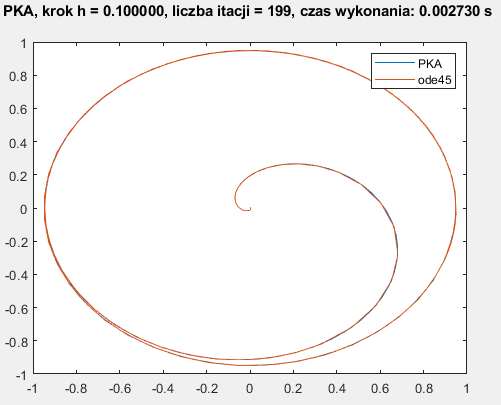
1. 𝑥1(0)=8 𝑥2(0)=0

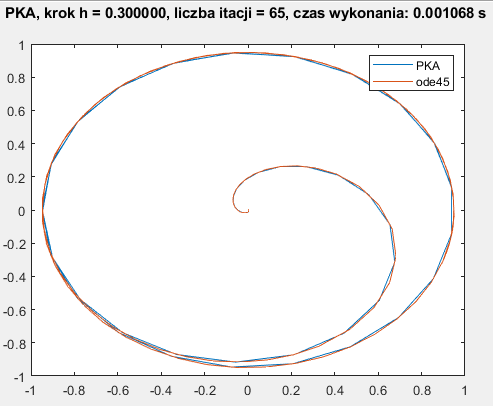
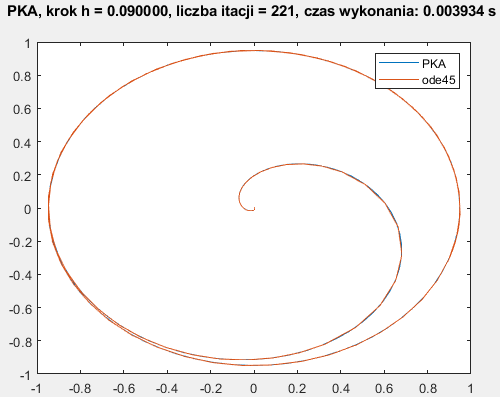


**Właściwa trajektoria dla h = 0.008:**

1. x1(0)=0.001, x2(0)=0.001

**Właściwa trajektoria dla h = 0.09:**

# Wynik:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | RK4 | | RK4Z | | ADAMS | |
|  | iteracje | czas | iteracje | czas | iteracje | czas |
| 𝑥1(0)=10 𝑥2(0)=8 | 2858 | 0.008254 | **750** | **0.042310** | 3332 | 0.255689 |
| 𝑥1(0)=0 𝑥2(0)=9 | 1819 | 0.002476 | **821** | **0.004321** | 2499 | 0.035307 |
| 𝑥1(0)=8 𝑥2(0)=0 | 1429 | 0.002808 | **781** | **0.003675** | 2499 | 0.028842 |
| x1(0)=0.001, x2(0)=0.001 | **134** | **0.000426** | 658 | 0.003977 | 221 | 0.003934 |

# Podsumowanie:

Wyniki zostały porównywane z wbudowaną funkcją ode45. Na podstawie jej zostały dobrane parametry dla poszczególnych metod i podpunktów.

Nie ma jednej metody, która byłaby najlepsza we wszystkich czterech przypadkach.

Metoda ze zmiennym krokiem była najlepsza w 3/ 4 przypadków. Okazała się słabsza w ostatnim przypadku.

Metody RK4 i Adamsa potrzebują podobnej liczby iteracji, jednak iteracje w tej drugiej metodzie są bardziej czasochłonne.

Metoda predyktor-korektor Adamsa, której rezultaty są dość podobne do metody RK4 ze stałym krokiem (są jednak zauważalne, iż metoda ta daje prawidłowy wynik trochę dla większych kroków niż RK4).

Metoda RK4 ze zmiennym krokiem, zmniejszająca ilość iteracji w porównaniu do metody RK4 ze stałym krokiem (rosnący krok zmniejsza liczbę iteracji). Jednak coś za coś – w każdej iteracji wykonywanych jest więcej obliczeń. Zdecydowaną zaletą tej metody jest automatycznie dobierany krok w zależności od danej trajektorii (np. w miejscach przegięcia wykresu zwiększa on swoją dokładność itd.). Najlepsza z tych metod!!

# Dodatek:

function [y1,y2] = func(x1,x2)

%Funkcje opisujace ruch pnktu

% y1 do pochodnej pierwszej

% y2 do drugiej pochodnej

y1 = x2 + x1\*(0.9 - x1^2 - x2^2);

y2 = -x1 + x2\*(0.9 - x1^2 - x2^2);

end

function [dydt] = odefun(t, x)

dydt = [x(2)+x(1)\*(0.9-x(1)^2-x(2)^2) ;-x(1)+x(2)\*(0.9-x(1)^2-x(2)^2)];

end

function [] = RK4(x1, x2, h)

it = 1;

tic;

for i = 0:h:20

% wpisujemy do macierzy wyników

results(it,1) = x1;

results(it,2) = x2;

it = it + 1;

% obliczenie współczynników k

[k11,k12] = func(x1,x2);

[k21,k22] = func(x1 + 1/2 \* h \* k11, x2 + 1/2 \* h \* k12);

[k31,k32] = func(x1 + 1/2 \* h \* k21, x2 + 1/2 \* h \* k22);

[k41,k42] = func(x1 + h \* k31, x2 + h \* k32);

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

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

end

results(it,1) = x1;

results(it,2) = x2;

it = it - 1;

t = toc;

% porównanie z funkcją ode45 programu Matlab

[~,y] = ode45(@odefun, [0 20], [results(1,1) results(1,2)]);

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

hold on;

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

legend('RK4', 'ode45');

hold off;

plotTitle = sprintf('RK4, krok h = %f, liczba iteracji = %d, czas wykonania: %f s \n', h, it, t);

title(plotTitle);

end

function [w,it] = RK4Z(x1,x2,h)

it = 1; s = 0.9; % współczynnik bezpieczeństwa

ew = 10^(-8); % dokładność względna

eb = 10^(-8); % dokładność bezwzględna

beta = 5; %maksymalna zmiana kroku w jednej iteracji

hMin = 10^(-12);

i=0;

tic;

while(true)

w(it,1) = x1;

w(it,2) = x2;

w(it,3) = h;

w(it,6) = i;

[k11,k12] = func(x1,x2);

[k21,k22] = func(x1 + 1/2 \* h \* k11, x2 + 1/2 \* h \* k12);

[k31,k32] = func(x1 + 1/2 \* h \* k21, x2 + 1/2 \* h \* k22);

[k41,k42] = func(x1 + h \* k31, x2 + h \* k32);

x1e = x1;

x2e = x2;

h=h/2;

for j=1:2

[k11e,k12e] = func(x1e,x2e);

[k21e,k22e] = func(x1e + 1/2 \* h \* k11e, x2e + 1/2 \* h \* k12e);

[k31e,k32e] = func(x1e + 1/2 \* h \* k21e, x2e + 1/2 \* h \* k22e);

[k41e,k42e] = func(x1e + h \* k31e, x2e + h \* k32e);

x1e = x1e + (1/6) \* h \* (k11e + 2 \* k21e + 2 \* k31e + k41e);

x2e = x2e + (1/6) \* h \* (k12e + 2 \* k22e + 2 \* k32e + k42e);

end

h=h\*2;

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

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

% błędy aproksymacji

d1 = (x1e-x1)/15;

d2 = (x2e-x2)/15;

w(it,4)=d1;

w(it,5)=d2;

e1 = abs(x1e) \* ew + eb;

e2 = abs(x2e) \* ew + eb;

alpha = min([((e1/abs(d1))^(1/5)),((e2/abs(d2))^(1/5))]); % 1/p+1 = 1/5

hNext = s \* alpha \* h;

x1 = x1e;

x2 = x2e;

if (s \* alpha >= 1)

if ( i + h >= 20 )

break;

else

i = i + h;

h = min([hNext, beta \* h, 20 - i]);

end

else

if (hNext < hMin)

error('Nie można rozwiązać z zadaną dokładnością');

else

h = hNext;

x1 = w(it,1);

x2 = w(it,2);

end

end

it = it + 1;

end

it = it - 1;

t = toc;

[~,y ] = ode45(@odefun, [0 20], [w(1,1) w(1,2)]);

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

hold on;

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

legend('RK4', 'ode45');

hold off;

plotTitle = sprintf('RK4Z liczba iteracji = %d, czas wykonania: %f s \n', it, t);

title(plotTitle);

end

function Adams(x1,x2,h)

it = 1;

tic;

for i = 1:3

% wpisujemy do macierzy wyników

results(it,1) = x1;

results(it,2) = x2;

it = it + 1;

% obliczenie współczynników k

[k11,k12] = func(x1,x2);

[k21,k22] = func(x1 + 1/2 \* h \* k11, x2 + 1/2 \* h \* k12);

[k31,k32] = func(x1 + 1/2 \* h \* k21, x2 + 1/2 \* h \* k22);

[k41,k42] = func(x1 + h \* k31, x2 + h \* k32);

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

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

end

for i = 5:floor(20/h)

results(it,1) = x1;

results(it,2) = x2;

x = [x1 x2]';

sigma = ((55/24) \* odefun(0,[x1 x2]) - (59/24) \* odefun(0,[results(it-1,1) results(it-1,2)]) + (37/24) \* odefun(0,[results(it-2,1) results(it-2,2)]) - (9/24) \* odefun(0,[results(it-3,1) results(it-3,2)]));

pe = x + h \* sigma;

sigma = ((9/24) \* odefun(0,pe) + (19/24) \* odefun(0,[x1,x2]) - (5/24) \* odefun(0,[results(it-1,1) results(it-1,2)]) + (1/24) \* odefun(0,[results(it-2,1) results(it-2,2)]));

x = x + h \* sigma;

x1 = x(1);

x2 = x(2);

it = it+1;

end

results(it,1) = x1;

results(it,2) = x2;

it = it - 1;

t = toc;

% porównanie z funkcją ode45 programu Matlab

[~,y] = ode45(@odefun, [0 20], [results(1,1) results(1,2)]);

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

hold on;

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

legend('PKA', 'ode45');

hold off;

plotTitle = sprintf('PKA, krok h = %f, liczba itacji = %d, czas wykonania: %f s \n', h, it, t);

title(plotTitle);

end