**Informatyka II**

Temat: Całkowanie numeryczne równań różniczkowych zwyczajnych

Autor: Adam Walocha

Numer indeksu: 304386

Kierunek: Energetyka

Nr grupy dziekańskiej: 9

Prowadzący: Michał Stachura

Zadanie

Dane jest zagadnienie początkowe postaci:

Jego dokładne rozwiązanie to:

Ćwiczenie 1-3

1. Napisz program, który rozwiąże dane zagadnienie początkowe z wykorzystaniem schematu Eulera.
2. Wykorzystaj schemat RK4 zaimplementowany w pliku rk4.cpp i ponownie rozwiąż zagadnienie.
3. Dla obu przypadków wyświetl na monitorze kolejne wartości i oraz względne wartości błędów: .

Rozwiązanie ćwiczenia 1-3:

#include <iostream>

#include <cstdlib>

#include <math.h>

#include "rk4.h"

//zagadnienie początkowe: {1} y'(t)=f(t,y);

//warunek początkowy: {2} y(t0)=y0;

double t0 = 0, tk = 2;//t nalezy do takiego przedzialu

double y\_0 = 1;

double lambda = 10;

double f(double t, double y);//f(t,y);

double y\_an(double t);

int main()

{

double t = t0, y = y\_0;

double h = 0.01, eps\_euler = 0, eps\_rk4 = 0;

FILE\* dane;

errno\_t err = fopen\_s(&dane, "dane.txt", "a+");

for (int i = 0; i < tk / h; i++)//metoda eulera

{

t += h;//czyli t[i+1] = t[i] + h;

y += h \* f(t, y);//czyli y[i+1] = y[i] + h \* f(t[i], y[i]);

eps\_euler = fabs(y - y\_an(t)) / fabs(y\_an(t));

printf("%lf\t%lf\t%lf\n", t, y, eps\_euler);

}

t = t0, y = y\_0;

printf("----------------------------------------\n");

for (int i = 0; i < tk / h; i++)//metoda rungego-kuty

{

t += h;

y = rk4(t, y, h, f);

eps\_rk4 = fabs(y - y\_an(t)) / fabs(y\_an(t));

printf("%lf\t%lf\t%lf\n", t, y, eps\_rk4);

}

fclose(dane);

}

double f(double t, double y)

{

return lambda \* y;

}

double y\_an(double t)

{

return y\_0 \* exp(lambda \* (t - t0));

}

Dla rozważanego przypadku i zadanych założeń, czyli i kroku całkowania h = 0,01 względna wartość błędu jest dużo mniejsza w przypadku metody RK4. Względna wartość błędu w przypadku metody Eulera dla małych wartości jest rzędu wielkości . Wraz z obliczaniem wartości dla kolejnych względna wartość błędu rośnie aż do rzędu wielkości .

Metoda Rungego-Kuty czwartego rzędu jest niemal doskonała. Względna wartość błędu dla początkowych wartości funkcji dla argumentu jest rzędu . Wraz ze wzrostem wartości argumentu rząd wielkości rośnie jedynie do .

Wniosek: Dla około 200 kroków całkowania na przedziale metoda Rungego-Kuty jest dużo lepsza. Wykorzystując metodę Eulera można zauważyć, że względna wartość błędu rośnie bardzo szybko.

Ćwiczenie 4-5:

1. Zmodyfikuj program tak aby wykonywał obliczenia jedną i drugą metodą dla zadanego i liczby kroków: . Wydrukuj do pliku: liczbę kroków , długość kroku , błąd metody Eulera i błąd metody RK4 dla ostatniego kroku czasowego.
2. Sporządź wykresy błędów obu metod w funkcji kroku i oszacuj ich rzędy zbieżności.

Rozwiązanie ćwiczenia 4-5:

#include <iostream>

#include <cstdlib>

#include <math.h>

#include "rk4.h"

//zagadnienie początkowe: {1} y'(t)=f(t,y);

//warunek początkowy: {2} y(t0)=y0;

double t0 = 0, tk = 2;//t nalezy do takiego przedzialu

double y\_0 = 1;

double lambda = 10;

double f(double t, double y);//f(t,y);

double y\_an(double t);

int main()

{

double t = t0, y = y\_0;

double h = 0.01, eps\_euler = 0, eps\_rk4 = 0;

FILE\* dane;

errno\_t err = fopen\_s(&dane, "dane\_do\_zadania.txt", "a+");

FILE\* wyniki\_euler;

errno\_t err2 = fopen\_s(&wyniki\_euler, "wyniki\_metoda\_eulera.txt", "a+");

FILE\* wyniki\_rk4;

errno\_t err3 = fopen\_s(&wyniki\_rk4, "wyniki\_metoda\_rk4.txt", "a+");

for (int i = 0; i < 7; i++)//wszystkie h zamienione na tk/n - nowy krok calkowania

{

double N = pow(2, i);//n - liczba krokow jakie nalezy wykonac

t = t0, y = y\_0; fprintf(wyniki\_euler, "N = %lf\n", N);

for (int j = 0; j < N; j++)//metoda eulera

{

t += (tk / N);//czyli t[i+1] = t[i] + h;

y += (tk / N) \* f(t, y);//czyli y[i+1] = y[i] + h \* f(t[i], y[i]);

eps\_euler = fabs(y - y\_an(t)) / fabs(y\_an(t));

fprintf(wyniki\_euler, "%lf\t%lf\t%lf\n", t, y, eps\_euler);

}

t = t0, y = y\_0; fprintf(wyniki\_rk4, "N = %lf\n", N);

for (int j = 0; j < N; j++)//metoda rungego-kuty

{

t += (tk / N);

y = rk4(t, y, (tk / N), f);

eps\_rk4 = fabs(y - y\_an(t)) / fabs(y\_an(t));

fprintf(wyniki\_rk4, "%lf\t%lf\t%lf\n", t, y, eps\_rk4);

}

fprintf(dane, "%lf\t%lf\t%lf\t%lf\n", N, (tk / N), eps\_euler, eps\_rk4);//dla ostatniego przedzialu czasowego

}

fclose(dane);

fclose(wyniki\_euler);

fclose(wyniki\_rk4);

printf("wszystkie wyniki mozna znalezc w plikach\n");

}

double f(double t, double y)

{

return lambda \* y;

}

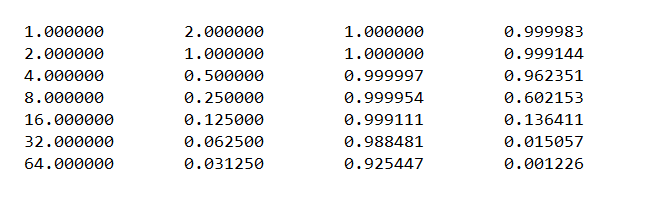
double y\_an(double t)

{

return y\_0 \* exp(lambda \* (t - t0));

}

Przeanalizujmy jak zmienia się względna wartość błędu dla zmieniającej się liczby kroków, długości kroku zarówno dla metody Eulera jak i metody RK4.

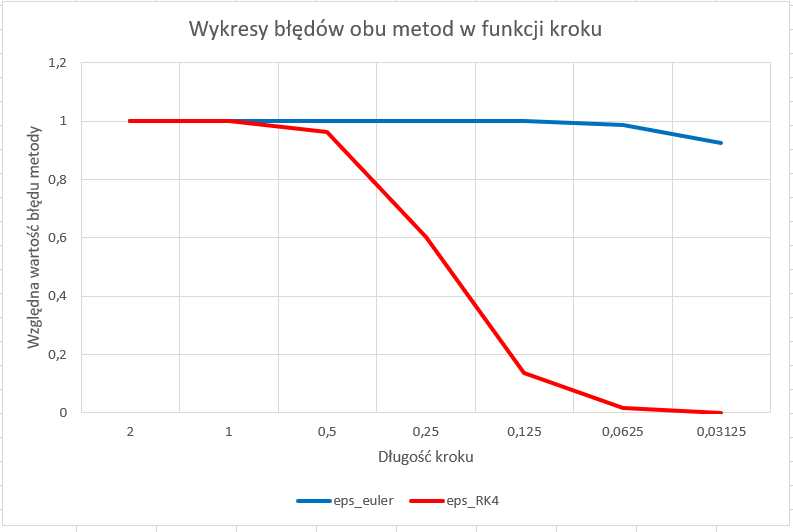


W kolumnie pierwszej możemy zobaczyć ilość kroków całkowania, w drugiej kolumnie długość kroku całkowania , w trzeciej kolumnie zebrane zostały względne wartości błędu dla metody Eulera, natomiast w czwartej kolumnie odpowiednio względne wartości błędu metody RK4.

Dla małej liczby kroków całkowania widać, że obie metody są równie słabe. Jednak dla wykładniczo zmieniającej się liczby kroków i odpowiednio zmniejszającej się wartości długości kroku wyraźnie widać że w przypadku metody Eulera względna wartość błędu właściwie niewiele się zmienia. W przypadku metody RK4 już dla około 32 kroków dla obranego przedziału wartości względne błędów są mniejsze o jeden rząd wielkości. Już dla 64 kroków metoda RK4 jest lepsza o dwa rzędy wielkości.

Wniosek:

Metoda RK4 pomimo niewiele bardziej skomplikowanego kodu pozwala uzyskać dużo dokładniejsze wyniki, jednak trzeba zastosować odpowiednio dużą liczbę kroków aby uzyskać widoczną przewagę nad metodą Eulera.



Na wykresie powyżej można zauważyć jak zmienia się względna wartość błędu dla obu metod w funkcji kroku . Względna wartość błędu dla metody Rungego-Kuty zbiega dużo szybciej niż metoda Eulera. W tej skali wyraźna zmiana dla metody Eulera następuje dopiero dla wartości, dla których metoda RK4 zaczyna być niemal doskonała.

Wniosek:

Metoda Rungego-Kuty czwartego rzędu jest dużo lepszą metodą. Jednak aby uzyskać niemal doskonałe wyniki, należy użyć powyżej 64 kroków. Aby osiągnąć identyczne rzędy zbieżności dla obu metod, należy wykonać około kroków.