CAŁKOWANIE NUMERYCZNE RÓWNAŃ RÓŻNICZKOWYCH ZWYCZAJNYCH

LABORATORIUM 4

JAKUB MROCZKOWSKI GR.9

NR. INDEKSU 304336

1.Temat zajęć.

Celem laboratorium numer 4 było zapoznanie z metodami numerycznymi za pomocą, których będę wstanie z określoną dokładnością rozwiązać równanie różniczkowe zwyczajne z zadanym zagadnieniem początkowym. Dzięki tym metodom mogę zauważyć jak za pomocą komputera rozwiązywać równania różniczkowe metodami innymi niż analitycznymi, co z pewnością przyda się w przyszłości. W ramach ćwiczenia muszę rozwiązać równanie z zadanym zagadnieniem początkowym oraz porównać wyniki przedstawiając je na odpowiednim wykresie.

2. Metody

W celu rozwiązania ćwiczenia musiałem skorzystać z zarówno metody Eulera oraz Runngego-Kutty, które pokrótce opiszę poniżej.

Metoda Eulera jest jedną z najprostszych metod rozwiązywania równań różniczkowych, co zresztą potwierdza data opublikowania metody, czyli rok 1768. Metoda Eulera jest metodą rzędu pierwszego i opiera się na ukierunkowaniu stycznej w kierunku szukanego punktu, a różnica pomiędzy wartością szukanego punktu, a wartością punktu przybliżonego to błąd metody.

Metoda Rungego-Kutty jest jedną z najpopularniejszych metod R-K i często używana przez inżynierów. Jest to metoda dokładniejsza od metody Eulera ponieważ jest rzędu 4.

3. Kod z laboratorium

W moim kodzie znajduje się rozwiązanie następującego równania różniczkowego: y’(t) = lambda*y(t), za lambdę przyjąłem 2, a rozwiązuje następujące zagadnienie początkowe y(0)=1. Moje h przyjąłem początkowe, jako 0.5. Wartość t początkowa to 0, a wartość t końcowa 5.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "rk4.h"

double funkcja(double t, double y);
double lambda;

void main()
{
    FILE *fp;
    lambda=2.0;
    double h =0.5;
    double tp=0;
    double yp=1;
    double tk=5.0;
    double ypa=1;
    double ya=1;
    double tpa=0;
    double yap=1;
    double epsilon;

    printf("Metoda Eulera\nt0=%lf\t\t\ty0=%lf\n" , tp,yp);

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

    {
        tp= tp + h;
        yp = yp + h*funkcja(tp,yp);
        ya = yap * exp(lambda*(tp-tpa));
        epsilon = ((fabs(ya-yp))/fabs(ya));
        printf("t%d=\t%lf\t\ty%d=\t%lf\t\teps=\t%lf\n\n",i+1,tp,i+1,yp,epsilon);
    }
```

Opisując po kolei mój kod, jako pierwszą rzecz oczywiście zaimplementowałem odpowiednie biblioteki oraz rk4.h pobrane ze strony naszego wydziału. Deklaruje po kolei wszystkie zmienne krok, t początkowe i końcowe. Zmienne ya, tpa, yap służą jako zmienne pomocnicze do obliczania sposobem analitycznym równania różniczkowego. Tworzę pętlę for, gdzie „i” musi być mniejsze od (tk/h) zgodnie ze wzorem, że h = (tk-tp)/n, jako że tp =0 wzór przybrał formę (tk/h). Następnie korzystam z danego wzoru metody Eulera, obliczam również epsilon między rozwiązaniem Eulera i analitycznym, zgodnie ze wzorem epsilon=|yan-ynum|/|yan|.

```
double rk=1;
double epsilon2;
tp=0;
yp=1;
printf("Metoda Kutty Rudego\nto=%lf\t\t\tyo=%lf\n" , tp,yp);

for(int i = 0 ; i <(tk/h);i++)
{
     tp = tp+h;
     rk=rk4(tp,rk,h,funkcja);
     ya = yap * exp(lambda*(tp-tpa));
     epsilon2 = (fabs(ya-rk))/fabs(ya);
     printf("t%d=\t%lf\t\ty%d=\t%lf\t\teps=\t%lf\n\n",i+1,tp,i+1,rk,epsilon2);
}
```

Następnie przechodzę do metody Runngego-Kutty, deklaruję pomocniczą zmienną rk=1, jako y początkowe, oraz epsilon2, który liczy błąd metody R-K. Kolejno tworzę identyczna pętlę for, gdzie iteruje tp , następnie korzystam z funkcji rk4, oraz ponownie wpisuje w pętlę funkcję liczącą wartości analitycznie.

```
double N;
double rk1=1;
double epsilone,epsilonrk;
fp=fopen("wyn.txt","w");

for(int i =0; i<7 ;i++)
{
    tp=0;
    yp=1;
    rk1=1;
    N = pow(2.,i);
    printf("Euler\n\nt0=%lf\t\ty0=%lf\n",tp,yp);
        for(int j =0 ; j <N;j++)
        {

            h=tk/N;
            tp=tp+h;
            yp = yp+h*funkcja(tp,yp);
            ya = yap * exp(lambda*(tp-tpa));
            epsilone=((fabs(ya-yp)/fabs(ya)));
            printf("t%d=\t%lf\t\ty%d=\t%lf\t\teps=\t%lf\n\n" ,j+1, tp,j+1,yp,epsilone);

        }
        printf("\n\n");


    tp=0;
    yp=1;

    printf("RUDY Kutta\n\nt0=%lf\t\ty0=%lf\n" , tp,yp);
    for(int j=0;j<N;j++)
    {
        h = tk/N;
        tp = tp+h;
        rk1=rk4(tp,rk1,h,funkcja);
        ya = yap * exp(lambda*(tp-tpa));
        epsilonrk = (fabs(rk1-ya))/fabs(ya);
        printf("t%d=\t%lf\t\ty%d=\t%lf\t\teps=\t%lf\n\n" ,j+1,tp,j+1,rk1, epsilonrk);
```

Zgodnie z poleceniem teraz zmieniać ma się liczba kroków, w potędze 2,a wykładnik rośnie od 0 do 6 zatem wprowadzam zmienną N, kolejną zmienna rk1 równą 1, tak jak y początkowe oraz kolejne epsilony dla metody Eulera oraz metody R-K4. Tworzę pętlę w pętli, żeby wykonać odpowiednią ilość iteracji dla metod. Na początku zeruje tp oraz yp przyrównuję do 1 , gdyż posiadają one wciąż wartości poprzednie i wykonuje pętlę dla metody Eulera, druga pętla w pętli głównej wykonuje identyczną czynność z tą różnicą ze dla metody R-K4. W tejże pętli również umieszczam funkcję fprintf służącą do zapisu moich wyników do pliku wyn.txt, a na samym dole jest funkcja zwracająca wynik prawej strony równania różniczkowego.

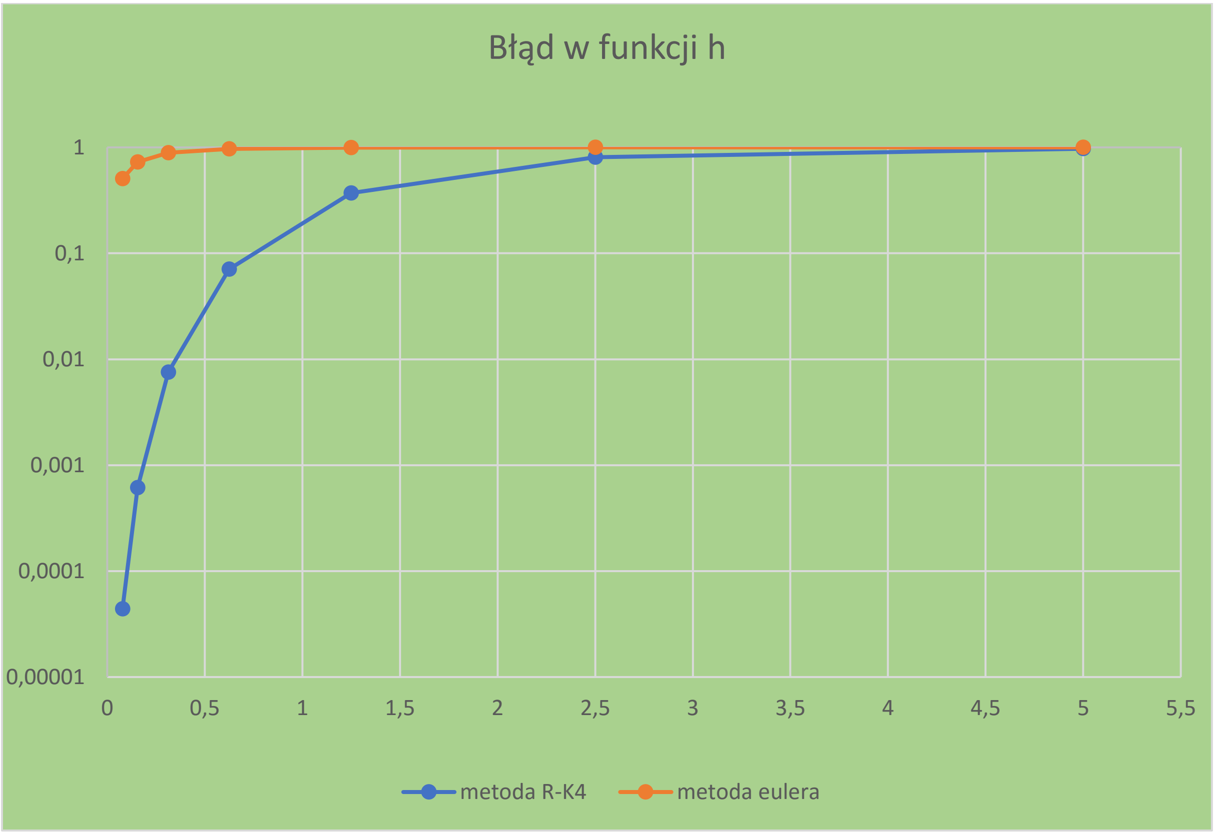
```
        fprintf(fp, "ilosc krokow=%lf\t\tdlugosc kroku=%lf\t\tblad eulera=%lf\t\tblad rudego kutty=%lf\n" , N,h,epsilone,epsilonrk);

        }

    system("PAUSE");
}

double funkcja(double t, double y)

{
    return lambda*y;
}
```

```
ilosc krokow=1.000000     dlugosc kroku=5.000000     blad eulera=0.999501     blad R-K4=0.970747
ilosc krokow=2.000000     dlugosc kroku=2.500000     blad eulera=0.998366     blad R-K4=0.805966
ilosc krokow=4.000000     dlugosc kroku=1.250000     blad eulera=0.993187     blad R-K4=0.369249
ilosc krokow=8.000000     dlugosc kroku=0.625000     blad eulera=0.970179     blad R-K4=0.070705
ilosc krokow=16.000000    dlugosc kroku=0.312500     blad eulera=0.892673     blad R-K4=0.007557
ilosc krokow=32.000000    dlugosc kroku=0.156250     blad eulera=0.726956     blad R-K4=0.000613
ilosc krokow=64.000000    dlugosc kroku=0.078125     blad eulera=0.507545     blad R-K4=0.000044
```

Powyżej wklejam screen z moimi wynikami podpunkt 4 w ćwiczeniu, wyniki uzyskane przedstawiam na poniższym wykresie.

4.Wnioski

Na poniższym wykresie widać wyraźną różnicę w zachowaniu poszczególnych metod. Metoda Eulera przy małej ilości kroków, co zarazem przekłada się na większą długość kroku całkowania jest prawie identycznie błędna jak metoda R-K4. Tempo wzrostu błędu dla metody R-K4 jest proporcjonalne do długości kroku. Metoda R-K4 szybko osiąga małą wartość, co wynika z większego rzędu zbieżności tej metody w stosunku do metody Eulera, gdzie wykres jest praktycznie liniowy, co również podkreśla, że metoda Eulera jest rzędu pierwszego.



5. Podsumowanie

Wyraźnie widać, że metoda R-K4 jest lepsza od metody Eulera jest dokładniejsza, dosyć szybko osiąga prawie niezauważalny błąd zmniejszając h o 2,5 razy mamy o wiele dokładniejszą wartość naszego równania różniczkowego w przypadku równania rozpatrywanego w tym raporcie.