POLITECHNIKA KRAKOWSKA IM. TADEUSZA KOŚCIUSZKI
WYDZIAŁ FIZYKI, MATEMATYKI I INFORMATYKI
KIERUNEK FIZYKA TECHNICZNA
KRAKÓW, POLSKA

PIOTR WESZKA, MARCIN BOCHEŃSKI,
MAREK GRUCHAŁA

METODA RKF45

Raport do projektu, wykonanego w ramach zajęć.

Kraków 2016

Spis treści

1 Wprowadzenie

Celem projektu, było stworzenie programu, który rozwiązywał by zadane zagadnienia z wykorzystaniem metody numerycznej Runge - Kutty 45. Metoda RK 45 należy do tzw. metod krokowych, oznacza to, że można ją wykorzystywać do iteracyjnego rozwiązywania równań różniczkowych.
Istnieje wiele metod RK, o wielu stopniach, wielu krokach, różnych rzędach, i różniących się między sobą innymi własnościami (jak stabilność, jawność, niejawność, metody osadzone, szybkość działania, itp.). Metoda Rungego-Kutty 4. rzędu jest powszechnie stosowana ze względu na prostotę implementacji, relatywnie proste wzory, dużą szybkość oraz wysoki rząd metody. Metoda ta w przeciwieństwie do innych, znanych i powszechnie wykorzystywanych, prostszych metod numerycznych (np. metoda Eulera) genruje mniejszy błąd numeyrczny. Nazwa metody pochodzi od nazwisk Niemieckich matematyków, którzy ją stworzyli, tj: Carla Rungego i Martina Wilhelma Kutta.

2 Opis działania metody

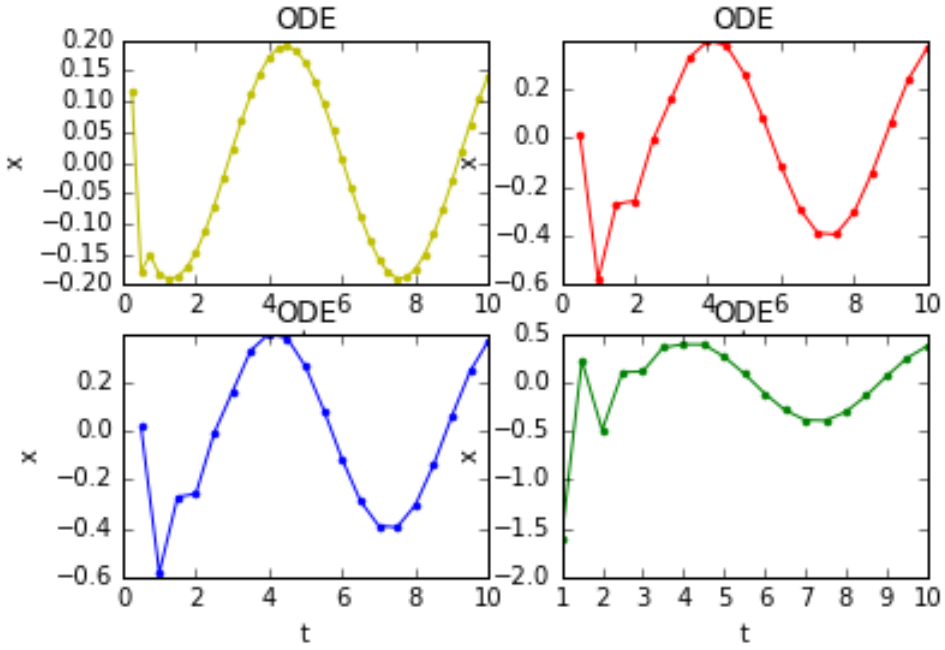
Metoda zakłada iteracyjne rozwiązywanie zadanego zagadnienia, posiada również "procedurę", która pozwala określić czy do rozwiązania zagadnienia wykorzystano odpowiednią wielkość kroku h. W każdej iteracji, wykonywane są dwa przybliżenia wyniku oraz ich wyniki są porównane. Jeśli wyniki pozostają ze sobą w zgodzie, uzyskane w ten sposób przybliżenie jest akceptowane, jeśli nie, wówczas krok numeryczny jest zmniejszany. W każdym krok wykorzystuje się sześć wartości, które wyliczane są zgodnie z poniższymi wzorami:

$$
\begin{aligned}
k_1 &= hf(x_i, y_i) \\
k_2 &= hf(x_i + \frac{h}{4}, y_i + \frac{k_1}{4}) \\
k_3 &= hf\left(x_i + \frac{3h}{8}, y_i + \left(\frac{3k_1}{32} + \frac{9k_2}{32}\right)\right) \\
k_4 &= hf\left(x_i + \frac{12h}{13}, y_i + \left(\frac{1932k_1}{2197} - \frac{7200k_2}{2197} + \frac{7296k_3}{2197}\right)\right) \\
k_5 &= hf\left(x_i + h, y_i + \left(\frac{429k_1}{216} - 8k_2 + \frac{3680k_3}{513} - \frac{845k_4}{4104}\right)\right) \\
k_6 &= hf\left(x_i + \frac{h}{2}, y_i + \left(-\frac{8k_1}{27} + 2k_2\right.\right. \\
&\quad \left.\left. - \frac{3544k_3}{2565} + \frac{1859k_4}{4104} - \frac{11k_5}{40}\right)\right) \\
\Delta y_i &= \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6 \\
y_{i+1} &= y_i + \Delta y_i
\end{aligned}
$$

Δy oraz wartość metody w kroku i+1 wyliczane są zgodnie z powyższymi wzorami.

3 Prezentacja wyników

Użyta metoda sprowadza się do porównania rozwiązań uzyskanych na drodze różnych kalkulacji. Jeśli różnica między wynikami jest większa od przyjętej przez nas tolerancji wtedy zmniejszamy krok aż do momentu kiedy rozwiązanie zawiera się w przedziale który jest przez nas akceptowalny. Takie metody nazywamy adaptacyjnymi, ze względu właśnie na adaptację kroku.
W ten sposób możemy porównać dowolne dwie metody jednak byłoby to nieekonomiczne ponieważ dla każdego kroku musimy obliczyć dwa rozwiązania, a w przypadku gdy błąd jest większy niż tolerancja musimy ponowić obliczenia ze zmniejszonym krokiem. Optymalnym wyjściem z takiej sytuacji jest zastosowanie algorytmu RKF45. Korzystając z tego schematu porównujemy rozwiązania wyliczone za pomocą metody RKIV oraz RKV. Niektóre ze współczynników w tychże metodach są identyczne przez co nie tracimy czasu na dublowanie obliczeń. Ttradycyjna metoda RKIV potrzebuje pięciu współczynników, metoda RKV – czterech współczynników jednak niektóre ze współczynników są takie same więc zamiast w każdym kroku obliczać 9 różnych współczynników obliczamy 6. Na zamieszczonym poniżej rysunku widzimy cztery wykresy przedstawiające rozwiązanie tego samego równania ale ze zwiększoną tolerancją. Jak widać lewy górny wykres najszybciej zaczyna być gładki, jest to wykres z najmniejszą tolerancją błędu. Warto też zauważyć że w miejscach gdzie funkcja ma punkty przegięcia kroki zostają zagęszczone aby jak najlepiej przybliżyć rozwiązanie, zawdzięczamy adaptacyjnemu dostosowywaniu kroku.


ODE
ODE
ODE
ODE
x
x
x
x
t
t

4 Kod programu

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Tue Oct 25 11:17:45 2016

@author: piotrek
"""

# -*- coding: utf-8 -*-


'''
Program rozwiazujacy rownanie rozniczkowe I-go rzedu zwyczajne
'''
import numpy as np
import csv
import matplotlib.pyplot as plt

def fun1(x,t):
    """Funkcja caŁkowana"""
    return x+np.sin(t)


k=[]
j=[]

"""wspolczynniki do merody numerycznej"""

n_2=3./8.;
n_3=3./32.;
n_4 =9./32.;
n_5 = 12./13.;
n_6 = 1932./2197.;
n_7 = 7200./2197.;
n_8 = 7296./2197.;
n_9 = 439./216.;
n_10 = 3686./513.;
n_11=845./4104.;
```

```
# n_12=0.5;
n_13=8./27.;
n_14=3544./2565.;
n_15=1859./4104.;
n_16=11./40.;
n_17=16./135.;
n_18=6656./12825.;
n_19=28561./56430.;
n_20=9./5.
n_21=2./55.;
n_22=25./216.
n_23=1408./2565.
n_24=2197./4104.
n_25=1./5.

def RK45(fun1, t0, tf, x0, h0, emin, emax, hmin, hmax):

    """Metda Rungego-Kutty-Fahlberga do rozwiÄ...zywania rownan postaci x'=
        z warunkiem poczatkowym


    DANE WJEŹšCIOWE:
        fun1    - funkcja
        t0      - lewy kraniec przedzialu
        tf      - prawy kraniec przedzialu calkowania
        x0      - wartosc poczatkowa
        h0      - krok poczatkowy
        emin    - minimalny blad
        emax    - maksymalny blad
        hmin    - minimalny krok calkowania
        hmax    - maksymalny krok caŁ,kowania

    DANE WYJSCIOWE:
        t       - tablica zawierajca czas
        x       - tablica zawierajaca rozwiazanie
    """

    h=h0
    x=x0
    t=t0
```

```
N=51
di=(tf-t0)/N
k=0
RKF4=0
RKF5=0
e=0
X=[]
T=[]

csvfile=open('data.csv','w')
fieldnames = [\
              'x',\
              't',\
              'h']
writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
writer.writeheader()

while (k<N and t<tf):

    if(h<hmin):
        h=hmin
    elif(h>hmax):
        h=hmax

    k0=h*fun1(x,t)

    k1=h*fun1(x+0.25*k0, t+0.25*h)

    k2=h*fun1(x+n_3*k0+n_4*k1,t+n_2*h)

    k3=h*fun1(x+n_6*k0+n_7*k1+n_8*k2,t+h*n_5)

    k4=h*fun1(x+n_9*k0-8.*k1+n_10*k2-n_11*k3,t+h,)

    k5=h*fun1(x-n_13*k0+2.*k1-n_14*k2+n_15*k3-n_16*k4,t+0.5*h)


    RKF4=n_17*k0+n_18*k1+n_19*k2-n_20*k4+n_21*k5
    RKF5=n_22*k0+n_23*k2+n_24*k3-n_25*k4
```

```
            e=abs(RKF4-RKF5)
            if(e>emax and h>hmin):
                h=h/2
            else:
                k+=di
                t+=h
                x=RKF4
                T.append(t)
                X.append(x)
                if(e<emin):
                    h=2*h

            #h=(pow((emax-emin)/(2*abs(RKF4-RKF5)),0.25))
           # print (x,t,h)
            csvfile=open('data.csv','a')
            writer.writerow({\
                             'x':x,\
                             't':t,\
                             'h':h})

    csvfile.close()
    return T,X
#RK45(fun1,t0,tf,x0,h0,emin,emax,hmin,hmax)

x=[]
t=[]
color=['y','r','b','g']

t,x=RK45(fun1,0,10,-1,1.0,0.001,1.0,0.01,1.0)
for i in range(4):
    t,x=RK45(fun1,0, 10, -1, 1.0, 0.001-i*0.1, 1.0+i*1, 0.01-i*0.1, 1.0+
    plt.subplot(2, 2, i+1)
    plt.plot(t, x, '{}.-'.format(color[i]))
    plt.title('ODE')
    plt.ylabel('x')
    plt.xlabel('t')

plt.savefig("wykres.png")
plt.show()
```