mgr Agata Emilia WITEK

University of Szczecin: Faculty of Humanities, Institute of Philology *(graduate)* and
Faculty of Mathematics and Physics, Department of Informatics & Technical Education *(graduate)*
Uniwersytet Szczeciński: Wydział Nauk Humanistycznych, Instytut Filologii *(absolwentka)* oraz
Wydział Matematyczno-Fizyczny, Katedra Edukacji informatycznej i Technicznej *(absolwentka)*

RUNGE - KUTTA ALGORITHM PRESENTATION IN C^{++} LANGUAGE

Abstract

Introduction and aim: The aim of this paper is to describe the basics of C^{++} for educational purposes, taking into account issues such as the origins of language, comparing the languages C and C^{++}. There will be presented Runge-Kutta algorithm in the C^{++} source code. The results obtained will be compared with the results obtained on the same input data, calculated in *MathCAD* and presented by W. Lipinski in the book: Numerical calculation-connection in the theory of signals and electrical circuits.
Material and methods: Differential equation of the fourth degree - Runge-Kutta will be written in C^{++} based on the results obtained in the program *MathCAD*. Shown as a flowchart, there will be presented an input data, and there will be described the source code as the results.
Results: Based on a comparison of results there has been presented the simplicity of this execution and the possibility of its further analysis.
Conclusion: There has been given the source code of the program C^{++}, and results were compared with the results obtained in the program *MathCAD*, with using the same data.

Keywords: Runge-Kutta algorithm, C^{++}, *MathCAD*.
(Received: 21.08.2016; Revised: 27.08.2016; Accepted: 31.08.2016)

PREZENTACJA ALGORYTMU RUNGEGO - KUTTY W JĘZYKU C^{++}

Streszczenie

Wstęp i cel: *Celem niniejszej pracy będzie opisanie podstaw języka C^{++} w celach dydaktycznych, uwzględniając takie zagadnienia, jak: początki języka, porównanie języków C oraz C^{++}. W dalszej części pracy przedstawiony zostanie algorytm Rungego-Kutty w kodzie źródłowym języka. Wyniki uzyskane, zostaną porównane z wynikami, uzyskanymi na takich samych danych wejściowych, wyliczonych w programie MathCAD, przedstawione przez W. Lipińskiego w książce: Obliczenia numeryczne w teorii sygnałów i obwodów elektrycznych.*
Materiał i metody: *Równanie różniczkowe czwartego stopnia - algorytm Rungego-Kutty - zostanie napisane w języku C^{++} na podstawie wyników otrzymanych w programie MathCAD. Pokazany zostanie schemat blokowy algorytmu, przedstawione zostaną dane wejściowe, oraz opisany zostanie kod źródłowy algorytmu z wynikami.*
Wyniki: *Na podstawie porównania wyników została przedstawiona akuratność kodu, zanalizowana prostota jego wykonania i możliwość jego dalszej analizy.*
Wiosek: *Napisano kod źródłowy programu C^{++}, a wyniki porównano z wynikami uzyskanymi w programie MathCAD, z wykorzystaniem tych samych danych.*

Słowa kluczowe: *Algorytm Rungego-Kutty, C^{++}, MathCAD.*
(Otrzymano: 21.08.2016; Zrecenzowano: 27.08.2016; Zaakceptowano: 31.08.2016)

1. Introduction

The ability to use a programming language is a useful skill in the face of the increasing computerization and the demand for this kind of knowledge. Computerization and the creation of programs to support new technologies is a challenge for future programmers and computer scientists.

Although C^{++} was "superstructure" for the C language, the understanding of the presented text, examples and programs, does not require programming skills in C language. The official father of C^{++} is considered Bjarne Stroustrup. The most important change introduced in C^{++} compared to C was object-oriented programming and a number of other improvements designed to make the language more comfortable and flexible than its original. Some of the changes in the standard C language were inspired by the language C^{++} afterwards[1].
C^{++} allows to write program code in the smallest number of characters. For example> if one would like to increase a given variable with the amount of 5, in, the following languages are occupying a certain number of characters:

$$X = X + 5 \quad \text{(Basic, 5 digits)}, \tag{1}$$

$$X: = X + 5 \quad \text{(Pascal, 6 digits)}, \tag{2}$$

$$X = 5^{++} \quad \text{or even shorter } X^{+} = 5 \quad (C^{++}\text{, only 4 digits}). \tag{3}$$

The most important functions that are necessary to create the algorithm are: *main ()* function, and instructions: *if* conditional instructions, instruction *while, do ... while* loop, *for* loop, a *switch* statement, and the ability to use simple functions.

Programming in C^{++} carries a lot of opportunities. An interesting fact that is possible in this language is object-oriented programming. The differential equation of the fourth degree - Runge-Kutta algorithm can be written in a different way, using object-oriented programming. The results of doing so, presented in this work, will be compared with the results obtained on the same input data, calculated in *MathCAD* program, presented by W. Lipinski book: *Obliczenia numeryczne w teorii sygnałów i obwodów elektrycznych (Numerical calculations in the theory of signals and electrical circuits)*[2]. However, it is crucial to know the basics of C^{++} programming, there has been released many teaching books about that topic that is why the basic C^{++} language explanation will be abandoned in this article.

1. Runge-Kutta methods – theory

If there is given a known $y(x_n)$, and the task is to express an approximation y_{n+1} of $y(x_n+h)$, then Runge-Kutta method is used, which consists calculating of the value f (x,y) in some particularly selected points, lying near the curve in the range of solutions (x_n, x_n+h) and creation of such a combination of these values, which gives a fairly good accuracy increase of (y_{n+1}-y_n).

The Runge - Kutta method is simple in its programming, according to A and G. Björck Dalquist[3], the most popular method is four-steps open method, which is calculated by using the following formulas:

$$k_1 = h \cdot f(x_n, y_n), \tag{4}$$

$$k_2 = h \cdot f\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1\right), \tag{5}$$

[1] Majczak A.K.: *C^{++} w 48 godzin*, Warszawa 1993, p. 7.
[2] Lipiński W.: *Obliczenia numeryczne w teorii sygnałów i obwodów elektrycznych*, Szczecin 2010, pp. 123-124.
[3] Bjorck A., Dahlquist G.: *Metody Numeryczne*, Warszawa, PWN 1987, pp. 335-336.

$$k_3 = h \cdot f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2), \tag{6}$$

$$k_4 = h \cdot f(x_n + h, y_n + k_3), \tag{7}$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{8}$$

Full evidences of the above formulas justify heuristic solution, but for the exact solution which is given, however, the formula is being presented as the following:

$$y_{n+1} - y_n = \int_{x_n}^{x_n+h} \frac{dy}{dx} dx = \int_{x_n}^{x_n+h} f(x, y(x))\, dx. \tag{9}$$

Runge-Kutta methods are based on the converging integral with the available data[4].

1.1. A schematic flow diagram

The block diagram of the recording of a program in C++, which would be use to describe the algorithm in C++ is shown in the figure 1:

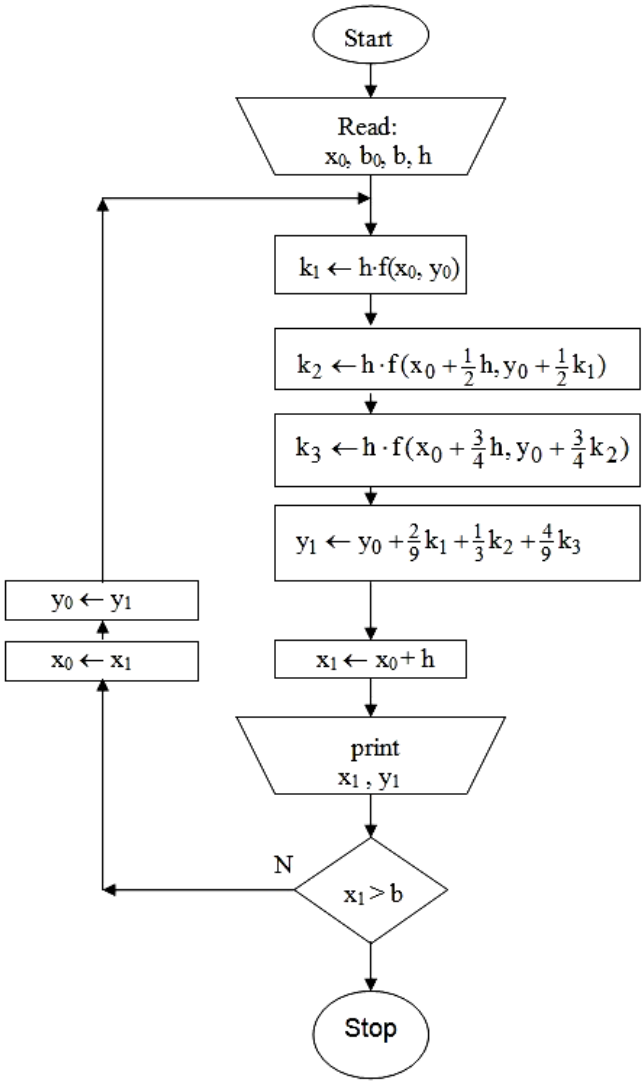


Fig. 1. Flowchart of Runge - Kutta

Source: Jaracz K., Folta W., Moszner K., Pękala M.: *Wprowadzenie do automatyki, informatyki i cybernetyki*, Kraków 1985, pp. 46

[4] Op. cit., pp. 335-336.

According to Wojciech Lipinski, Runge-Kutta algorithm (Runge-Kutta method) is an iterative numerical method to solve ordinary differential equations. The method of Runge - Kutta is known for the calculation of the next n-th time step $n \cdot \Delta t = n \cdot h$ that takes into account the values of the function within this time step. In the fourth order method, it is being taken into account the four function values corresponding to the change k1, k2, k3, k4. The calculation can be performed with a variable time step, using a less time step at intervals where the function changes are larger. The calculated column zero which is known by using Runge -Kutty of the matrix A specifies the time, the first column is the current: i(t), the second column is the charge: q(t), zero row of the matrix A defines the initial conditions (Fig. 2)[5].

RK(h,N):=

$x_0 \leftarrow 0$
$y_0 \leftarrow 0$
$t \leftarrow 0$
for $i \in 0..2$
  $A_{0,i} \leftarrow 0$
for $n \in 0..N$
  $k1 \leftarrow F(t, x_n, y_n) \cdot h$
  $K1 \leftarrow G(t, x_n, y_n) \cdot h$
  $k2 \leftarrow F(t+0.5 \cdot h, x_n+0.5 \cdot k1, y_n+0.5 \cdot K1) \cdot h$
  $K2 \leftarrow G(t+0.5 \cdot h, x_n+0.5 \cdot k1, y_n+0.5 \cdot K1) \cdot h$
  $k3 \leftarrow F(t+0.5 \cdot h, x_n+0.5 \cdot k2, y_n+0.5 \cdot K2) \cdot h$
  $K3 \leftarrow G(t+0.5 \cdot h, x_n+0.5 \cdot k2, y_n+0.5 \cdot K2) \cdot h$
  $k4 \leftarrow F(t+h, x_n+k3, y_n+K3) \cdot h$
  $K4 \leftarrow G(t+h, x_n+k3, y_n+K3) \cdot h$
  $t \leftarrow t+h$
  $x_{n+1} \leftarrow \frac{1}{6} \cdot (k1+2 \cdot k2+2 \cdot k3+k4)+x_n$
  $y_{n+1} \leftarrow \frac{1}{6} \cdot (K1+2 \cdot K2+2 \cdot K3+K4)+y_n$
  $A_{n+1,0} \leftarrow t$
  $A_{n+1,1} \leftarrow x_n+1$
  $A_{n+1,2} \leftarrow y_n+1$
A

Fig. 2. *MathCAD* program

Source: Lipiński W.: Numerical calculations in the theory of signals and electrical circuits, Szczecin 2010, pp. 123-124.

1.2. Data

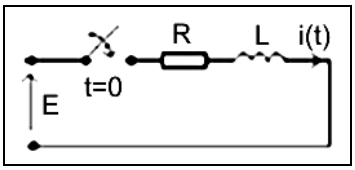
The simple case of a serial circuit RL connected to the DC shows the details of the algorithm Runge-Kutta. W. Lipinski analyzed the case in the program *MathCAD*. The results were compared with the results of the procedure *rkfixed*, where the course and the rules are not the subject of this analysis. The procedure described in *MathCad* will be placed in the source code in C^{++} using object-oriented programming. Based on data from one of the examples described by W. Lipinski, the results from the program written in C^{++} will be compared with results obtained in the program *MathCAD*.

According to the above data and a flow chart shown in figure 1 and the data shown in the figure 3, there has been established a program written in C^{++}, which includes the fourth row.

The results that professor W. Lipinski had received are shown on the figure 4 .

[5] Lipiński W.: *Numerical calculations in the theory of signals and electrical circuits*, Szczecin 2010, p. 123.

Runge-Kutta algorithm for calculating waveforms in the circuit RL bundled into a DC voltage, an initial condition zero i(0)=x(0)=0.



$$L\cdot\frac{d}{dt}x(t)+R\cdot x(t)=E$$

$$\frac{d}{dt}x(t)=-\frac{R}{L}\cdot x(t)+\frac{E}{L}$$

$(R\ \ L\ \ E)\equiv(5\ \ 4\cdot10^{-3}\ \ 29)$

$$T:=\frac{L}{R}\qquad t_{end}:=5\cdot T\qquad \Delta T:=\frac{t_{end}}{N}$$

Solution of the differential equation

$$\frac{d}{dt}x(t)=F(t,x(t))$$

$$F(t,x(t)):=-\frac{R}{L}\cdot x(t)+\frac{E}{L}$$

$$\int_{t_n}^{t_{n+1}}\frac{d}{dt}x(t)\,dt=\int_{t_n}^{t_{n+1}}F(t,x(t))\,dt$$

$$x_{n+1}=x_n+\int_{t_n}^{t_{n+1}}F(t,x(t))\,dt$$

Closest integrals option corresponds to the solution

$$x_{n+1}=x_n+F(t_n,x(t_n))\cdot h$$

$h=\Delta t$

The function F describes the Circuit, the unknown f: x=i(t). RK procedure depends on the time h=Δt and N, one must declare function F(t,x). Rkfixed procedure for the function F is given as D.

RK(h,N):=

$x_0 \leftarrow 0$
$t_0 \leftarrow 0$
for $m \in 0..1$
  $A_{0,m} \leftarrow 0$
for $n \in 0..N$
  $k1 \leftarrow F(t_n, x_n)\cdot h$
  $k2 \leftarrow F(t_n+0.5\cdot h, x_n+0.5\cdot k1)\cdot h$
  $k3 \leftarrow F(t_n+0.5\cdot h, x_n+0.5\cdot k2)\cdot h$
  $k4 \leftarrow F(t_n+h, x_n+k3)\cdot h$
  $t_{n+1} \leftarrow t_n+h$
  $x_{n+1} \leftarrow \frac{1}{6}\cdot(k1+2\cdot k2+2\cdot k3+k4)+x_n$
  $A_{n+1,0} \leftarrow t_{n+1}$
  $A_{n+1,1} \leftarrow x_{n+1}$
A

Rkfixed procedure
Function D
Initial condition (v=0)

$$D(t,v):=-\frac{R}{L}\cdot v+\frac{E}{L}$$

v:=0 N=4000
n:=0..N

ORIGIN (Default = 0)

This represents the starting index of all. To change it choose from the Math menu
ORIGIN = 0
Column zero determines the time
Column one defines electricity

State = RK(Δt, N)

$$Result=rkfixed(v,0,N,\Delta t,N,D)$$

$(Result^{\langle1\rangle})_{200}=1.28296$ $(Stan^{\langle1\rangle})_{200}=1.28296$

Fig. 3. Runge-Kutta algorithm to calculate waveforms in a circuit RL attached to DC voltage
Source: Lipiński W.: Numerical calculations in the theory of signals and electrical circuits, Szczecin 2010, pp. 123-124.

Result =

	0	1
0	OE+000	OE+000
1	1E-006	7.24547-003
2	2E-006	14.48189-003
3	3E-006	21.70927-003
4	4E-006	28.92762-003
5	5E-006	36.13695-003
6	6E-006	43.33728-003
7	7E-006	50.52861-003
8	8E-006	57.71096-003
9	9E-006	64.88434-003
10	10E-006	72.05876-003

State =

	0	1
0	OE+000	OE+000
1	1E-006	7.24547-003
2	2E-006	14.48189-003
3	3E-006	21.70927-003
4	4E-006	28.92762-003
5	5E-006	36.13695-003
6	6E-006	43.33728-003
7	7E-006	50.52861-003
8	8E-006	57.71096-003
9	9E-006	64.88434-003
10	10E-006	72.05876-003

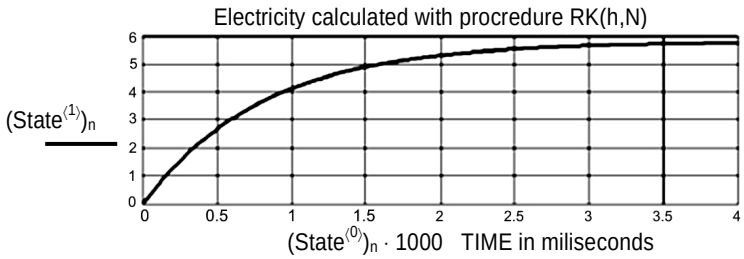


Fig. 4. Transient in the serial circuit RL
Source: Lipiński W.: Numerical calculations in the theory of signals and electrical circuits, Szczecin 2010, p. 124.

1.3. Runge-Kutta algorithm in C^{++} - Object-Oriented Programming

Runge-Kutta methods in C^{++} can be presented in a way that object using classes. Basic information about classes in C^{++} can be found in many positions and publications[6]. The source code of this algorithm in the programming language C^{++} is shown in the figure 5:

Runge-Kutta.cpp

```
#include<iostream>
#include<conio.h>
#include<cmath>
#include<fstream>

using namespace std;

class rungekutta
(
public:
    double X,Y,N,L,R,h;
    double function (double a, double b)
    (
    double y=(a+b);
    return y;
    )

    void calculate()
    (
        ofstream result;
        result.open("wynik.txt");
        h = ((5000 * (L/R))/N);
        cout<<h;
    for (X=0 ; X<4 ; X=X+h)
    (
    double k1 = (function(X,Y)*h);
    double k2 = (function(X+0.5*h,Y+0.5*k1)*h);
    double k3 = (function(X+0.5*h,Y+0.5*k2)*h);
    double k4 = (function(X+h,Y+k3)*h);

    Y=Y+((k1+2*k2+2*k3+k4)/6);

    cout<<Y<<"`\t"<<X<<endl;
    result<<Y<<"`\t"<<X<<endl;
    )
result.close();
    )
);


main()
(
rungekutta RK1;

RK1.X = 0;
RK1.Y = 0;
RK1.N = 4000;
RK1.R = 5;
RK1.L = 0.004;

cout<<"FOr X= "<<RK1.X<<" And Y= "<<RK1.Y<<endl
    <<"There will be computed an equasion with using \n"
    <<"Runge-Kutta method.\n\n";

    RK1.calculate();

    getch();
)
```

Fig. 5. Source code counting Runge-Kutta algorithm in C^{++}. The program is using an object-library <fstream>
Source: Elaboration of the Author

[6] J. Gręboszowski J.: *Symfonia C, Vol. 1 & 2*. Cracow 1999 or Majczak A.K.: *C^{++} in 48 hours*. Warsaw 1993.

The program has used the library <fstream> that is allowing to print out the results to the file. All results are saved to the file output.txt. The beginning of the lines program is shown in Figure 6. Within this results a graph can be made with using the output file *output.txt*. With using the Gnuplot programme, one can make a graph that is shown in the figure 7.

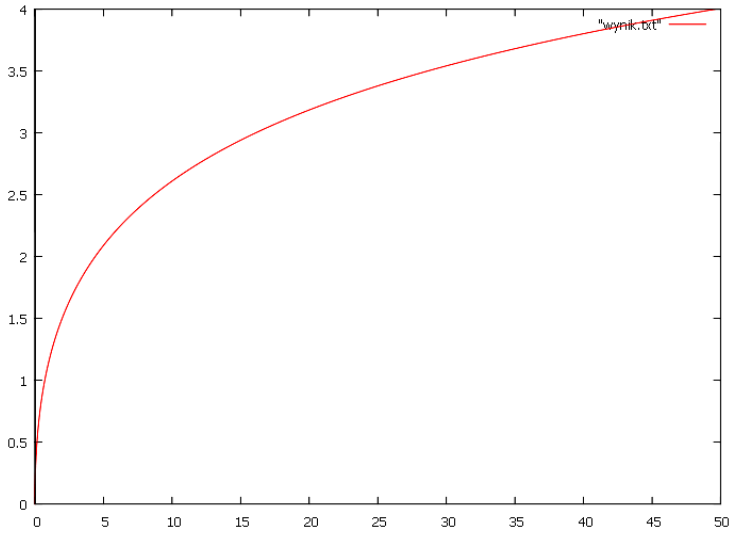


Fig. 6. The beginning lines of the finish programme
Source: Elaboration of the Author

```
0.0015.00167e-007       0
2.00133e-006    0.001
4.5045e-006     0.002
8.01068e-006    0.003
1.25209e-005    0.004
1.80361e-005    0.005
2.45573e-005    0.006
3.20855e-005    0.007
4.06218e-005    0.008
5.01671e-005    0.009
6.07224e-005    0.01
7.22889e-005    0.011
8.48674e-005    0.012
9.84589e-005    0.013
0.000113065     0.014
0.000128685     0.015
0.000145322     0.016
0.000162976     0.017
0.000181649     0.018
0.00020134      0.019
```

Fig. 7. Graph executed in the program Gnuplot
Source: Elaboration of the Author

2. Conclusions

- In this paper, there was proposed a program source code, which analyzes the Runge-Kutta algorithm.
- There has been given the source code of the program, and the results given by using it was compared to the results obtained in the program *MathCAD*, with using the same data.
- To create this program, however, it is required some basic knowledge of the syntax of C++, but the wide range of books and courses available on the market should facilitate the challenge of learning the programming language.
- The source code for this analysis is short and neat which shows how helpful the programming can be also in the circuit analysis.
- The ability to create the source code, for example, in C++ can be very useful, because it is a language quite often used in computer software.

Literature

[1] Aho A. V., Sethi R., Ullman J. D.: *Kompilatory : reguły, metody i narzędzia*. Warszawa: Wyd. Naukowo-Techniczne, 2002.

[2] Banachowski L., Diks K., Rytter W.: *Algorytmy i struktury danych*. Warszawa: Wyd. Naukowo-Techniczne, 2006.

[3] Bjorck A., Dahlquist G.: *Metody Numeryczne*. Warszawa: PWN, 1987.

[4] Gręboszˇ J.: *Symfonia* C^{++}. Kraków 1999.

[5] Jaracz K., Folta W., Moszner K., Pękala M.: *Wprowadzenie do automatyki, informatyki i cybernetyki*. Kraków 1985.

[6] Kisilewicz J.: *Język* C^{++} *- programowanie obiektowe*. Wrocław: Oficyna Wyd. Politechniki Wrocławskiej, 2005.

[7] Lipiński W.: *Obliczenia numeryczne w teorii sygnałów i obwodów elektrycznych.* Szczecin 2010.

[8] Lippman S. B., Lajoie J.: *Podstawy języka* C^{++}. Warszawa: Wyd. Naukowo-Techniczne, 2000.

[9] Majczak A. K.: C^{++} *w 48 godzin*. Warszawa 1993.

[10] Porębski W. M.: *Programowanie w języku* C^{++}. Gdańsk 1998.