A New Seventh Order Runge-kutta Family: Comparison 
with the Method of Butcher and Presentation of a 
Calculation Software

Hippolyte Séka, Assui Richard Kouassi

Department of Mathematics and Informatics, Institut National Polytechnique Houphouet-Boigny, Yamoussoukro, Ivory Coast

Email address: hippolyte.seka@inphb.ci (H. Séka), r.assui@yahoo.fr (A. R. Kouassi)

To cite this article:
doi: 10.11648/j.mcs.20190403.12

Abstract: This paper is in the context of the numerical resolution of ordinary differential equations. Most equations are unsolved in the analytic aspect. The goal is to find among the existing methods, the best method of numerical resolution. Also to facilitate the implementation of methods by introducing a calculation software. To do this, we use the Runge-Kutta method which is one of the best methods of numerical resolutions. That is why a family of Runge-Kutta methods of order 7 is presented. This family depends on the parameter $b_8$ and contains the well known method of Butcher [8] ($b_8 = 77/1440$). To obtain convincing results, we compare methods according to the values of $b_8$ with those of Butcher. The stability region is also studied to essentially perceive the numerical behavior that manifests itself when the steps of discretization tend to 0. The study shows that the stability region of this method does not depend on the coefficient $b_8$. To get the values of $b_8$, Java programming is used. Finally, to facilitate the implementation of the resolution, very simple software for numerical resolution of the ordinary differential equations is given. This software is designed for all students, also for all those who have no basis in numerical analysis and java programming to be able to find a solution approached with error estimate to an ordinary differential equation.

Keywords: Ordinary Differential Equation, Runge-Kutta, Stability Region, Java Programming

1. Introduction

The RK method is one of the best methods for numerically solving ODE. Several method of order $\geq 4$ have been discovered. [1, 3, 4-7]. However the search for better methods is always up to date. In this paper, a new RK7 depending on the coefficients $b_8$ is presented. We show that this family contains the Butcher method [8]. The stability region is independent of $b_8$. By java programming, using an example we determine some values of $b_8$ for which methods are better compared to that of Butcher [8]. Finally, a presentation of calculation software in Java programming is made to determine an approximate value of the ordinary differential equation. This paper will be organized by the following plan: section 2 RK7 method for ODEs, section 3 Stability region, section 4 Numerical analysis, section 5 The RK7 java calculator, section 6 conclusion.

2. RK7 Method for ODEs

Let’s solve the following ODE.

$$y' = f(x, y(x)),$$  \hspace{1cm} (1)

with $x_0 \leq x \leq x_n$ and the initial condition $y(x_0) = y_0$. Consider the nodes:

$$x_0, x_1 = x_0 + h, x_2 = x_0 + 2h, \ldots, x_n = x_0 + nh,$$  \hspace{1cm} (2)

where $h$ is the fixed step size. The goal is to determine $y_1, y_2, y_3, \ldots, y_n$ using the new RK7 family method as defined by Butcher tableau (see Figure 1).
Figure 1. Butcher Tableau for RK7 method family depending on parameter $b_c$

and

$$y_{i+1} = y_i + h \left( \left( \frac{77}{1440} \right) - b_c \right) k_1 + h \left( \frac{32}{105} \right) k_4 + h \left( \frac{1771561}{6289920} \right) k_5 + h \left( \frac{243}{2560} \right) k_6 + h \left( \frac{16087}{74880} \right) k_7 + h(b_c)k_8 + h \left( \frac{11}{270} \right) k_9 \quad (3)$$

with

$$k_1 = f(x_i, y_i) \quad (4)$$

$$k_2 = f \left( \left( \frac{1}{6} \right) h + x_i, \left( \frac{1}{6} \right) h k_1 + y_i \right) \quad (5)$$

$$k_3 = f \left( \left( \frac{1}{3} \right) h + x_i, \left( \frac{1}{3} \right) h k_2 + y_i \right) \quad (6)$$

$$k_4 = f \left( \left( \frac{5}{8} \right) h + x_i, \left( \frac{3}{8} \right) h k_3 + y_i \right) \quad (7)$$

$$k_5 = f \left( \left( \frac{7}{12} \right) h + x_i, \left( \frac{148}{1331} \right) h k_1 + \left( \frac{150}{1331} \right) h k_3 - \left( \frac{56}{1331} \right) h k_4 + y_i \right) \quad (8)$$

$$k_6 = f \left( \left( \frac{2}{3} \right) h + x_i, \left( \frac{1}{105} \right) h k_1 - \left( \frac{170}{27} \right) h k_3 + \left( \frac{4024}{1701} \right) h k_4 + \left( \frac{10648}{1701} \right) h k_5 + y_i \right) \quad (9)$$

$$k_7 = f \left( \left( \frac{1}{2} \right) h + x_i, \left( \frac{2466}{2409} \right) h k_1 + \left( \frac{1242}{343} \right) h k_3 - \left( \frac{19176}{16807} \right) h k_4 - \left( \frac{51909}{16807} \right) h k_5 + \left( \frac{1053}{2401} \right) h k_6 + y_i \right) \quad (10)$$

$$k_8 = f \left( \left( \frac{1}{576b_c} \right) h k_1 + \left( \frac{1}{105b_c} \right) h k_4 - \left( \frac{1331}{279552b_c} \right) h k_5 - \left( \frac{9}{1024b_c} \right) h k_6 + \left( \frac{343}{149760b_c} \right) h k_7 + y_i \right) \quad (11)$$

$$k_9 = f \left( \left( \frac{71}{32} \right) - \left( \frac{270b_c}{11} \right) \right) h k_1 - \left( \frac{195}{22} \right) h k_3 + \left( \frac{32}{7} \right) h k_4 + \left( \frac{29403}{3584} \right) h k_5 - \left( \frac{229}{112} \right) h k_6 + \left( \frac{1029}{1408} \right) h k_7 + \left( \frac{270b_c}{11} \right) h k_8 + y_i \right) \quad (12)$$

If $b_c = \frac{77}{1440}$ then we find the particular method of Runge-Kutta of order 7 defined in the book [8] on page 196.

3. Stability Region

The determination of the stability region has been developed in the literature. [9-11]. Thus the following differential equation is used:

$$y' = -\lambda y(t) \quad (13)$$

with $y(0)$ and $\lambda \in \mathbb{C}$ whose solution is $y(t) = y(0)e^{-\lambda t}$.

Let's study the case of the scheme (13):

$$k_1 = -\lambda y_i \quad (14)$$
Using (3), we obtain:

\[
y_{i+1} = \left(1 + \frac{h^6}{181440}\right) y_0 (z^9 + 6z^8 - 36z^7 + 252z^6 - 1512z^5 + 7560z^4 - 30240z^3 + 90720z^2 - 181440z + 181440)^n
\]

Let

\[ z = h\lambda \]

We obtain

\[ y_n = \left(1 + \frac{h^6}{181440}\right)^n y_0 (z^9 + 6z^8 - 36z^7 + 252z^6 - 1512z^5 + 7560z^4 - 30240z^3 + 90720z^2 - 181440z + 181440)^n \]

We write \( z = x + iy \), the stability region such as \( \frac{|y_n|}{|y_0|} < 1 \). (see Figure 2):

---

4. Numerical Analysis

4.1. Example

Let us use the numerical example of the publication [12] illustrated in Figure 2.

The governing first order ODE is given by

\[
R \frac{di}{dt} + \frac{i}{C} = \frac{dE(t)}{dt}
\]

Where \( R \) is the resistance (ohms), \( C \) is the capacitance (farads), \( i \) is the current (ampere), and \( E(t) \) is the voltage (volts). Given \( E(t) = \sin(t) \) volts, \( R = 5 \) ohms, \( C = 0.1 \) farads and at the initial time \( t = 0 \) the initial current is \( i = 0 \). We want to solve the differential equation (2), which is the RC-circuit ODE, for the time interval \( 0 \leq t \leq 0.05 \) seconds with the time step size \( t = 0.01 \) seconds using the RK7 family method. If the exact solution is given by

\[
k_2 = \frac{3}{2} y_1 (h\lambda - 6)
\]

\[
k_3 = -\frac{1}{18} y_1 (h^2\lambda^2 - 6h\lambda + 18)
\]

\[
k_4 = \frac{1}{48} y_1 (h^3\lambda^3 - 6h^2\lambda^2 + 24h\lambda - 48)
\]

\[
k_5 = \frac{1}{7986} y_1 (7h^4\lambda^4 + 8h^3\lambda^3 - 132h^2\lambda^2 + 1452h\lambda - 7986)
\]

\[
k_6 = -\frac{1}{1458} y_1 (8h^4\lambda^5 + 81h^4\lambda^4 - 72h^3\lambda^3 + 324h^2\lambda^2 - 972h\lambda + 1458)
\]

\[
k_7 = \frac{1}{43218} y_1 (104h^6\lambda^5 + 1170h^5\lambda^4 + 225h^4\lambda^3 + 4536h^3\lambda^2 - 15876h^2\lambda^2 + 37044h\lambda - 43218)
\]

\[
k_8 = -\frac{1}{181440} y_1 (h^7\lambda^7 + 20h^6\lambda^6 + 90h^5\lambda^5 + 181440b_8)
\]

\[
k_9 = \frac{3}{7986} y_1 (h^8\lambda^8 + 7h^7\lambda^7 - 114h^6\lambda^6 - 666h^5\lambda^5 - 812h^4\lambda^4 + 1232h^3\lambda^3 - 3696h^2\lambda^2 + 7392h\lambda - 7392)
\]

Note that the stability region does not depend on \( b_8 \).
\[ i = \left( \frac{10}{2501} \right) \cos(100t) + \left( \frac{508}{2501} \right) \sin(100t) - \left( \frac{10}{2501} \right) e^{-2t}. \]  

(27)
to yield \[ \frac{di}{dt} = 20 \cos(100t) - 2i = f(t, i). \]  

(28)

Then, the absolute errors could be calculated at each iteration. The computation procedure of the RK7 family method is summarized as follows [12]:

Rewrite the ODE in (13) by substituting the given values to yield

\[ i_{k+1} = i_k + h \left( \frac{77}{1440} - b_0 \right) k_1 + h \left( \frac{32}{105} \right) k_4 + h \left( \frac{177161}{623692} \right) k_5 + h \left( \frac{243}{2560} \right) k_6 + h \left( \frac{16807}{74880} \right) k_7 + h(b_0)k_8 + h \left( \frac{11}{270} \right) k_9 \]  

(29)

With

\[ k_1 = -2i + 20 \cos (100t), \]  

(30)\[ k_2 = -\left( \frac{2909}{150} \right) i - \left( \frac{1}{15} \right) \cos(100t) + 20 \cos \left( 100t + \frac{i}{6} \right), \]  

(31)\[ k_3 = -\left( \frac{44701}{22500} \right) i + \left( \frac{1}{2250} \right) \cos(100t) - \left( \frac{2}{15} \right) \cos \left( 100t + \frac{i}{6} \right) + 20 \cos \left( 100t + \frac{1}{3} \right), \]  

(32)\[ k_4 = -\left( \frac{443867}{222742} \right) i - \left( \frac{7905}{177557} \right) \cos(100t) + \left( \frac{244}{809643} \right) \cos \left( 100t + \frac{1}{6} \right) - \left( \frac{6769}{103506} \right) \cos \left( 100t + \frac{1}{3} \right), \]  

(33)\[ k_5 = -\left( \frac{32259}{16346} \right) i + \left( \frac{1306}{16799} \right) \cos(100t) - \left( \frac{3661}{216946} \right) \cos \left( 100t + \frac{1}{6} \right) + \left( \frac{6313}{2494} \right) \cos \left( 100t + \frac{1}{3} \right), \]  

(34)\[ k_6 = -\left( \frac{31523}{23239} \right) \cos \left( 100t + \frac{1}{6} \right) + 20 \cos \left( 100t + \frac{2}{11} \right), \]  

(35)\[ k_7 = -\left( \frac{27298}{13885} \right) i - \left( \frac{12962}{8777} \right) \cos \left( 100t + \frac{1}{3} \right) + \left( \frac{25653}{20402} \right) \cos \left( 100t + \frac{2}{11} \right) + \left( \frac{34454}{73977} \right) \cos \left( 100t + \frac{1}{2} \right), \]  

(36)\[ k_8 = -\left( \frac{1}{320512820512} \right) i - \left( \frac{343}{3744000} \right) \cos \left( 100t + \frac{6}{7} \right) + \left( \frac{1248}{2324605} \right) \cos \left( 100t + \frac{1}{3} \right), \]  

(37)\[ k_9 = -\left( \frac{14603}{7449} \right) i + \left( \frac{68160}{18623} \right) \cos \left( 100t + \frac{1}{3} \right) - \left( \frac{56659}{16803} \right) \cos \left( 100t + \frac{2}{11} \right) - \left( \frac{33964}{18229} \right) \cos \left( 100t + \frac{1}{2} \right), \]  

(38)

For example, we assume that \( b_8 \in [-1, 1] \). Then increment with \( h = \Delta t = 0.01 \) and \( i = 0 \) when \( t = 0 \).

4.2. Determination of Some Values \( b_8 \)

In this part, we determine some values of \( b_8 \) for which methods are better compared to that of Butcher [8]. The method is as follows:

The error is calculated at each step of the Butcher method. That is, \( b_8 = 77/1440 \). We write \( E_i \), the error with each step. Calculate the error at each step of the method for \( b_8 \neq 77/1440 \).

We count all the number of error at each step that are less than or equal to the number of error at each step of the method of Butcher. We write \( \text{compt} \), this number.

If \( \text{compt} = 100\% \), then all local errors in the new method are less than or equal to all the local errors of the Butcher method.

For example, we assume that \( b_8 \in [-1, 1] \). Then increment
b₈ by step of 0:0001 and 0:00001. compt = 100% are obtained (see Table 1). The code of the java programming is given on the annex page (see Appendix):

Table 1. Determination of some values of b₈.

<table>
<thead>
<tr>
<th>Incrementation de b₈ de 0.0001</th>
<th>Incrementation de b₈ de 0.00001</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0040000086556510925</td>
<td>-0.0010700002312660217</td>
</tr>
<tr>
<td>-0.00479999296665192</td>
<td>-0.00220998667240143</td>
</tr>
<tr>
<td>-0.002519998657172241</td>
<td>-0.0031010004758834839</td>
</tr>
<tr>
<td>-0.003599997937679291</td>
<td>-0.007650040990084076</td>
</tr>
<tr>
<td>-0.00639999657889339</td>
<td>-0.0028900036456640052</td>
</tr>
<tr>
<td>-0.0067999958992604395</td>
<td>-0.008370019495487213</td>
</tr>
<tr>
<td>-0.007199995219707489</td>
<td>-0.00861001047094727</td>
</tr>
<tr>
<td>-0.007999993860721588</td>
<td>-0.009009945656440582</td>
</tr>
<tr>
<td>-0.008399993181228638</td>
<td>0.0010700002312660217</td>
</tr>
<tr>
<td>-0.008799992501735687</td>
<td>0.00220998667240143</td>
</tr>
<tr>
<td>0.0040000006556510925</td>
<td>0.00310004758834839</td>
</tr>
<tr>
<td>0.00479999296665192</td>
<td>0.005570024251937866</td>
</tr>
<tr>
<td>0.005199998617172241</td>
<td>0.0057300254702765805</td>
</tr>
<tr>
<td>0.005599997937679291</td>
<td>0.006210029125213623</td>
</tr>
<tr>
<td>0.00639999657869339</td>
<td>0.006367003034532562</td>
</tr>
</tbody>
</table>

5. The RK7 Java Calculator

Let’s design the RK7 Program Java.exe to compute digital solutions. Before installing the software, the user will have to install the JDK java package. Just copy it and paste it wherever you want it and the software is installed. By clicking on RK7 Program Java.exe, we get the following window (see Figure 3).

The values of a; b; b₈ are given by default. Note that b₈ = b₈. The value of h is automatically calculated according to the choices of a; b and n. Remember that x is between a and b.

The user will have to enter the range f(x, y) which is defined by: y’ = f(x, y). He will also have to enter the exact solution in Exact solution range if he knows. Otherwise he can leave the field empty.

If the user clicks on calculation without filling in the field f(x,y) there is an error and we obtain the following dialog box (see Figure 4).

If after filling in the field f(x, y) the user enters the field b₈ = 0, then there is an error in the method and we obtain (see Figure 5).

Let us know the differential equation of the publication [2] and take the table of Butcher [1], i.e. b₈ = 77/1440. We obtain (see Figure 6).

6. Conclusion

In this article, a new seventh order Runge-Kutta family has been discovered. This method depends on the coefficient b₈. For b₈ = 77/1440, we find the Butcher method [8] on page 196. It has been shown that the stability region does not depend on the coefficient b₈. However, the code in java programming allows us to determine values of b₈ so that all the local errors of these methods are better than those of the Butcher method. This code can be used to find the best method between any two proposed methods. RK7 Java Calculator can be recommended in teaching and also to all those who have no basis in numerical analysis to find a numerical solution of an ordinary differential equation.
Acknowledgements

We would like to express our deepest appreciation and gratitude to Professor Sergey Khashin of Ivanovo State University who provided us the possibility to coordinate and complete this article.

Appendix

```java
package Perfect;

import com.singularsys.jep.Jep;
import com.singularsys.jep.JepException;

public class Perfect {
    static double a = 0;
    static double b = 1;
    static int n = 100;
    static double h = (b - a) * Math.pow(n, -1);
    static int s = 0;
    static double b8 = 0;
    static double y0 = 0;

    // function f=f_abs. For example: y'=f(x,y)
    static String f_abs = "20*cos(100*x)-2*y";

    // exact solution
    static String f_ex = "(10/2501)*cos(100*x)+(500/2501)*sin(100*x)-(10/2501)*exp(-2*x)";

    public static void main(String[] args) {
        // incrementation of b8
        for(float i=0;i<n;i++) {
            k[0] = f(x[i], y[i], funct);
            x[i+1] = x[i] + h;
            for(int j=1;j<s;j++) {
                k[j] = f(x[i] + butcher(ordre, b8)[j][0]*h, y[i] + h*somprod(butcher(ordre, b8), j), funct);
            }
            y[i+1] = y[i] + h*somprod(butcher(ordre, b8), k, j), funct);
            if((y[i+1] - f(x[i+1], f_ex)) <= (g[i+1] - f(x[i+1], f_ex))) {
                compt++;
            }
        }
        System.out.println(b8 + ":" + (compt*100)/n);
    }

    private static double f(double x, double y, String funct) {
        float f = 0;
        try {
            Jep jep = new Jep();
            jep.addVariable("x", x);
            jep.addVariable("y", y);
            jep.parse(funct);
            f = Float.parseFloat(jep.evaluate().toString());
        } catch (JepException e) {
            return f;
        }
        return f;
    }

    private static double f(double x, String funct) {
        float f = 0;
        try {
            Jep jep = new Jep();
            jep.addVariable("x", x);
            jep.addVariable("y", y);
            jep.parse(funct);
            f = Float.parseFloat(jep.evaluate().toString());
        } catch (JepException e) {
            return f;
        }
        return f;
    }

    // determination of percentage
    private static void Methode(int ordre, String funct, String exact, double b8) {
        // ... (remaining code)
    }

    // method of Butcher[1]
    private static double f(double x, double y, String funct) {
        float f = 0;
        try {
            Jep jep = new Jep();
            jep.addVariable("x", x);
            jep.addVariable("y", y);
            jep.parse(funct);
            f = Float.parseFloat(jep.evaluate().toString());
        } catch (JepException e) {
            return f;
        }
        return f;
    }

    // ... (remaining code)
}
```

References


