ORIGINAL ARTICLE
DOI: 10.5433/1679-0375.2019v40n2p123

Low storage explicit Runge-Kutta method

Método explícito de Runge-Kutta de baixo armazenamento

Diomar Cesar Lobão[1]

Abstract

This paper we are dealing with the high order accurate low storage explicit Runge Kutta (LSERK) methods which mainly are used for temporal discretization and are stable regardless of its accuracy. The main objective of this paper is to compare traditional RK with different forms of LSERK methods. The numerical experiments indicate that such methods are highly accurate and effective for numerical purposes. It's also shown the CPU time consuming and its solution implications. The method is well suited to achieve high order accurate solution for the scalar second order IVP (Initial Value Problem) problem as it is discussed in the present paper.

Keywords: LSERK. Explicit method. Runge-Kutta. System of ODE.

Resumo

Neste artigo estamos tratando dos métodos explícitos Runge Kutta (LSERK) de alta ordem e baixo armazenamento, que são usados principalmente para a discretização temporal e são estáveis independentemente de sua precisão. O principal objetivo deste trabalho é comparar o RK tradicional com diferentes formas de métodos LSERK. Os experimentos numéricos indicam que tais métodos são altamente precisos e eficazes para propósitos numéricos. Também é mostrado o tempo de CPU e suas implicações na solução. O método é bem adequado para obter uma solução precisa de alta ordem para o problema escalar de segunda ordem do problema de valor inicial (PVI), como é discutido no presente artigo.

Palavras-chave: LSERK. Método explícito. Runge-Kutta. Sistema de EDO.

[1] Prof. Dr., Dept. Ciências Exatas, UFF/EEIMVR, Volta Redonda, RJ, Brasil; E-mail: lobaodiomarcesar@yahoo.ca

Introduction

We understand that Runge-Kutta (RK) methods are an important family of single step iterative methods for approximating the solutions of ordinary differential equations (ODEs) through a series of intermediate stage computation. In the method of lines for discretizing time dependent partial differential equations (PDEs) it is mainly used RK methods to discretize in time. Depending on the PDE in study such procedure can result in a very large system of ODEs. Regarding to the storage requirements for standard RK methods it is known that it is proportional to the number of stage computations. The other hand, low storage explicit RK (LSERK) methods are a subclass of RK methods whose storage requirements are independent of the number of computations stages (BUTCHER, 1987). Such characteristic allows increasing of the number of stages without increasing storage allocation. Based on this fact it is observed that as we increase the number of stages the stability limit gets larger and allowing to increase the time step size thus speeding up time to solution for solving ODEs and or PDEs (BUTCHER, 1987; BUTCHER, 2003).

Our main objective for this paper is to study and explore the development of LSERK coefficients and its conversion to the standard RK of fourth order. In the present paper, we implement and teste fourth order LSERK methods. We obtain the stability regions and the accuracy conditions for the implemented methods and its time performance. We also explored the efficiency of the LSERK methods along with the classic fourth order RK method using a simple second order linear and non-linear ODE. It is shown how to construct a Butcher Tableau from an LSERK Tableau of coefficients. Numerical calculations are done in order to compare methods and verify its accuracy.

The remainder of this paper is divided into four sections. The first section we provide a general review of the RK methods. The second section we describe the algorithm foundation of the methods. The third section we discusse the stability regions for each method. Finally, the application of these methods is considered in a simple numerical example.

Runge-Kutta Methods

Let us consider the following initial value problem (IVP), given by

$$\begin{cases} \dfrac{dy}{dt} = f(t,y) \\ y(t_0) = y_0 \end{cases} \tag{1}$$

where f and y are vector functions. We are seeking the numerical approximation of the solution $y(t)$ of the IVP over the time interval $t \in [t_0, t_n]$. The time interval is then subdivided into n equally spaced subintervals in which the integration is carried out. Such a choice of an equally spaced point is not required for RK method. The approximation points are defined as

$$t_i = t_0 + idt, i = 0, 1, 2.., n \tag{2}$$

with $dt = (t_n - t_0)$, where dt is the time step size.

One step of a general, explicit RK method for numerically solving equation (1) is given by

$$\begin{aligned} K_i &= f\left(t_{n-1} + c_i dt, y_{n-1} + dt \sum_{j=1}^{i-1} a_{ij} K_j\right) \\ y_n &= y_{n-1} + dt \sum_{i=1}^{s} b_i K_i, \end{aligned} \tag{3}$$

where the variable s is the number of stages related to each RK method. The coefficients a_{ij} are the intermediate weights at each RK stage, b_i are the final stage weights, and c_i are the intermediate time levels. It is required the following condition

$$c_i = \sum_{j=1}^{s} a_{ij}. \tag{4}$$

As we can see below the Table 1 shows the general, explicit Butcher Tableau for equations (3) (BUTCHER, 1987), which cover various RK methods. One of the most well known RK methods is a fourth order, four stage method, referred to as RK4.

Table 1 – Butcher Tableau for equations (3).

c_1	0			
$\vdots$	a_{21}	$\ddots$		
$\vdots$	$\vdots$			
c_s	a_{s1}	$\dots$	a_{ss-1}	0
	b_1	$\dots$	$\dots$	b_s

Source: The author.

The Tableau for RK4 is given in Table 2.

Table 2 – Explicit Butcher Tableau for RK4

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

Source: The author.

In order to solve equation (1) it is need to evaluate the right-hand side function, given by f, four different times when using RK4. It also stores a total of five different variables, one for each iteration of RK4. In the case of LSERK methods which are a specific class of RK methods, it requires fewer storage registers allocations. As described in (WILLIAMSON, 1980) who showed that some RK schemes can be implemented in a $2N$-storage format, where N is the dimension of the ODE. This format requires only two registers of length n to implement. One step of the general s-stage $2N$ LSERK method is given as follow

$$\begin{cases} U_1^i = y_n \\ V_2^{i+1} = A_i V_2^i + dt\, f\left(t_n + c_i dt, U_1^i\right), i = 1,..,s \\ U_1^{i+1} = U_1^i + B_i V_2^i, i = 1,..,s \\ y_{n+1} = U_1^s \end{cases} \tag{5}$$

As we can see the equation (5) shows only two variables, and which are required in order to implement the LSERK method. This algorithm gives an implementation of this LSERK method as described by (KETCHESON, 2010). The calculation to determine the LSERK coefficients is done via the definition of the relationship between the Butcher coefficients and the LSERK coefficients making use of the following relations (WILLIAMSON, 1980)

$$\begin{cases} B_j = a_{j+1,j} \quad \texttt{for} \quad j \neq n; \quad j = 1,...,s-1 \\ B_s = b_s \end{cases} \tag{6}$$

$$A_j = \begin{cases} \dfrac{b_{j-1} - B_j}{b_j} \quad \texttt{if} \quad j \neq 1 \quad \texttt{and} \quad b_j \neq 0 \\ \dfrac{a_{j+1,j-1} - c_j}{B_j} \quad \texttt{if} \quad j \neq 1 \quad \texttt{and} \quad b_j = 0 \end{cases} \tag{7}$$

for $j = 2,...,s$ and $A_1 = 0$.

In Matlab® language the following code fragment implements these relations

```
for j = 1 : (s-1)
  B(1,j) = a(j+1,j);
end
B(1,s) = b(1,s);
  for i = 1 : s
    for j = 2 : s
      if b(1,j) ~= 0
        A(j) = (b(1,j-1) - B(1,j-1))/b(1,j);
      elseif i == j+1
        A(j) = (a(i,j-1) - c(j))/B(1,j);
      end
    end
  end
```

We note that the relations given in equations (6) and (7) provides a relationship between the Butcher and LSERK coefficients, however such relations do not show that all RK methods have a low storage implementation. Those relations are used to build an algorithm which can transform a LSERK coefficient array into a Butcher Tableau (CARPENTER; KENNEDY, 1994).

Stability Regions

Let us consider an ODE method applied to the model problem defined as (BUTCHER, 2003)

$$\begin{cases} \dfrac{dy}{dt} = \lambda y \\ y(0) = y_0. \end{cases} \tag{8}$$

The region of stability of the ODE method is a region in the complex plane $\Re$, such that $dt\lambda$ *in* $\Re$ so $y(t) \to 0$ as $t \to \infty$ for all initial values given by y_0 . The absolute stability region can be said as a property of the ODE method. As we numerically solve the ODE it is important to know the absolute stability region for the method which turns out to be useful for estimating the time step size required to achieve a qualitatively correct solution. Picking the RK of first up to fourth order and applying to the model equation given by equation (8), the resulting difference equations are written as

$$\begin{aligned} &\textbf{RK1} \to y_{k+1} = (1 + dt\lambda) y_k \\ &\textbf{RK2} \to y_{k+1} = \left(1 + dt\lambda + \frac{(dt\lambda)^2}{2!}\right) y_k \\ &\textbf{RK3} \to y_{k+1} = \left(1 + dt\lambda + \frac{(dt\lambda)^2}{2!} + \frac{(dt\lambda)^3}{3!}\right) y_k \\ &\textbf{RK4} \to y_{k+1} = \left(1 + dt\lambda + \frac{(dt\lambda)^2}{2!} + \frac{(dt\lambda)^3}{3!} + \frac{(dt\lambda)^4}{4!}\right) y_k. \end{aligned} \tag{9}$$

Thus, the regions where are defined the absolute stability for the standard RK methods are those regions defined in the complex plane in such way that for RK4 it is given by

$$\textbf{RK4} \to \left|1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!}\right| < 1. \tag{10}$$

We use the following methods throughout this paper. The first method is a low storage LSERK(5, 4), where 5 means the number of stages e 4 means the order, proposed by (CARPENTER; KENNEDY, 1994). The second method is a low storage LSERK(7, 4), by (ALLANPALLI, *et al*, 2009).

The third method is a low storage LSERK(13, 4), by (NIEGEMANN; DIEHL; BUSCH, 2012). The last method is the classical RK(4,4). Tables 3, 4 and 5 show all of the coefficients for each method.

Table 3 – Coefficients for LSERK(5,4)

s	A	B
1	0	$\frac{1432997174477}{9575080441755}$
2	$-\frac{567301805773}{1357537059087}$	$\frac{5161836677717}{13612068292357}$
3	$-\frac{2404267990393}{2016746695238}$	$\frac{1720146321549}{2090206949498}$
4	$-\frac{3550918686646}{2091501179385}$	$\frac{3134564353537}{4481467310338}$
5	$-\frac{1275806237668}{842570457699}$	$\frac{2277821191437}{14882151754819}$

Source: The author.

Table 4 – Coefficients for LSERK(7,4)

s	A	B
1	0	0.117322146869
2	−0.647900745934	0.503270262127
3	−2.704760863204	0.233663281658
4	−0.460080550118	0.283419634625
5	−0.500581787785	0.540367414023
6	−1.906532255913	0.371499414620
7	−1.450000000000	0.136670099385

Source: The author.

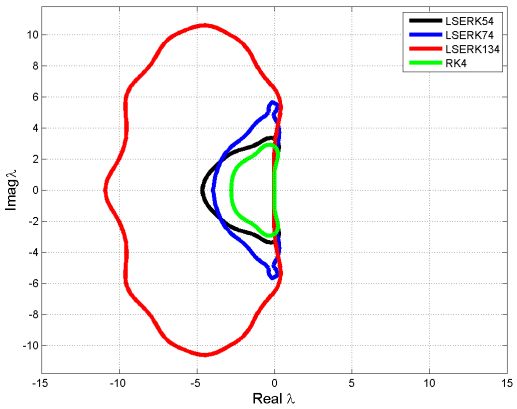
Table 5 – Coefficients for LSERK(13,4)

s	A	B
1	0	0.0271990297818803
2	−0.6160178650170565	0.1772488819905108
3	−0.4449487060774118	0.0378528418949694
4	−1.0952033345276178	0.6086431830142991
5	−1.2256030785959187	0.2154313974316100
6	−0.2740182222332805	0.2066152563885843
7	−0.0411952089052647	0.0415864076069797
8	−0.1797084899153560	0.0219891884310925
9	−1.1771530652064288	0.9893081222650993
10	−0.4078831463120878	0.0063199019859826
11	−0.8295636426191777	0.3749640721105318
12	−4.7895970584252288	1.6080235151003195
13	−0.6606671432964504	0.0961209123818189

Source: The author.

Making use of the relations given by equations (4), (6) and (7) the $a_{i,j}, b_j, c_i$ coefficients and stability regions are calculated, Figure 1.

Figure 1 – Stability Regions for LSERK and RK4.



Source: The author.

We can observe in Figure 1 that the region where each method will remain stable if their respective z values are within the boundary. If z is outside of the boundary, the solution is not stable. We also can notice from Figure 1 that RK4 includes a small portion of imaginary axis. However, the LSERK(13,4) includes a much larger stability region.

Using LSERK and RK4 to solve a ODE

Let us consider the linear IVP

$$\begin{cases} \dfrac{dy}{dt} = z \\ \dfrac{dz}{dt} = -4y \\ y(0) = 0 \\ z(0) = 1.0 \end{cases} \tag{11}$$

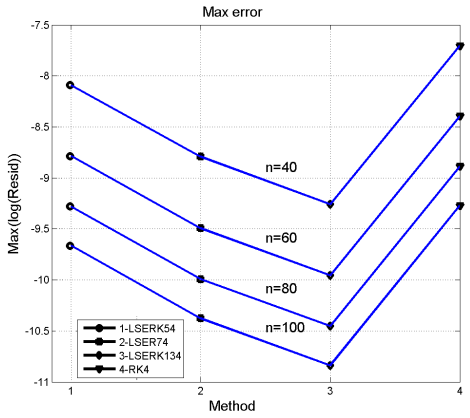
The exact solution can be obtained and is given as $y(t) = 0.5sin(2t)$.

We can compute the numerical solution of this ODE from the initial condition, $t_0 = 0$ to a final time $t_n = 1.0$. All algorithms LSERK and RK4 are used to solve the above ODE. The code is written in Matlab® language and all are vectorized. In order to compare the methods, we define the absolute error as follow

$$\texttt{Error} = \log\left(|(\texttt{Method})y_i - (\texttt{Exact})y_i|\right)_{\texttt{max}}. \tag{12}$$

The maximum error for different n is shown in Figure 2 as follow.

Figure 2 – Maximum error, linear OD.



Source: The author.

Observing the Figure 2 as expected the maximum absolute error is due the RK4 method, and the better numerical solution is obtained by LSERR(13,4) com 13 stages.

The second equation is a second order non-linear ODE, writing in system form

$$\begin{cases} \dfrac{dy}{dt} = z \\ \dfrac{dz}{dt} = tcos(t) \\ y(0) = 0 \\ z(0) = 1.0 \end{cases} \tag{13}$$

The exact solution is obtained and is given as $y(t) = 2sin(t) - tcos(t)$.

The maximum error for different n is shown in Figure 3 as follow.

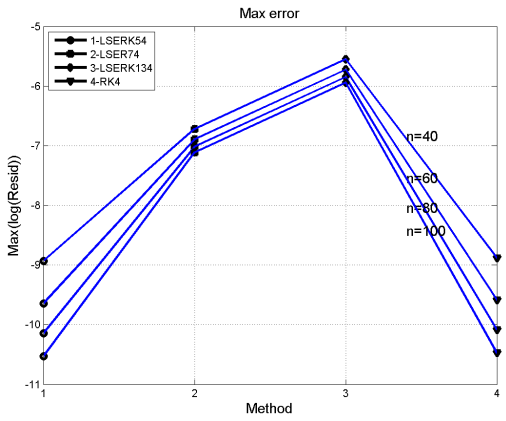
We can observe from Figure 3 that in a complete different behavior the maximum absolute error is due the LSERK(13,4) and the minimum error are to LSERK(5,4) followed by RK4.

Using the linear equation the time simulation is done consi-dering a sequence of time steps as defined by the set

$$n = 500150050001100016000220003000 0. \tag{14}$$

Figure 4 shows the average of 5 simulations in order to decrease the variations caused by the operation system tasks shared by the CPU when running the code. The hardware we used is a Dell Inspiron Serie 3000, Intel® Core™, 4GB RAM and HD of 0.5TB.

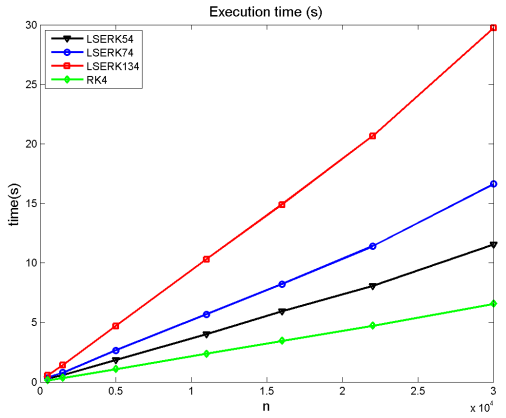
Figure 3 – Maximum error, non-linear ODE.



Source: The author.

We also show in Figure 4 the execution time for the simulation using different methods.

Figure 4 – Execution time.



Source: The author.

The best result, which is provided by the LSERK(13,4) is the most time consuming and it is related to the 13 stages performed by the algorithm (5) during the numerical solution. The computational time consuming cost of a LSRK or RK4 method is related to the number of times f must be called. Such issue is especially true for high dimensional ODEs such as those from the discretization of PDEs.

In order to discuss this time consuming cost, the operation count (number of f evaluations) for all set value for n was about 2494000 calls (comes from the profile Matlab® command). So, in order to use a larger time step, it is required to decrease the number of times the code calls f, which in consequence will reduce the computational time consuming cost of solving the problem in study. Consider what the potential savings for this present problem are. Taking the number of time steps as for example, $n = 1000$, we can show, Table 6, the following results

Table 6 – LSERK and RK4 costs.

	LSERK (5,4)	LSERK (7,4)	LSERK (13,4)	RK4
Calls f	5000	7000	13000	4000
Memoy (Mb)	16.0	16.5	17.0	15.8
Time (s)	0.6	0.8	1.4	0.4

Source: The author.

We can see in Table 6 the costs (Calls, memory and time) involved in the numerical simulation.

In the present case to numerically solve the second order ODE given by equation (11) it must be split in a system of two first orders ODE in order to apply the methods. Observing when a bigger system is intended to be solved the required storage for RK4 needed by equation (3) will be largely increased. This does not happen with the algorithm given by equation (5), such savings is one reason why any low storage methods are an attractive alternative to the standard RK4 method.

Conclusion

We know that the low-storage Runge-Kutta methods were designed for integration of general nonlinear or time variant systems, which is efficient in the use of the memory storage. In the present paper we carried out the implementation of three LSERK methods with low storage requirements, the properties of them are discussed under the computational view point and compared with the standard RK4 method. We also observed that: LSERK(13,4) admit two storage registers for its implementation, while requiring significantly more execution time is still a better choice for linear ODE. Although the general LSERK method consumes longer computational time, more than the forth order classical Runge-Kutta method, its accuracy pays off.

References

ALLANPALLI, V.; HIXON, R.; NALLASAMY, M.; SAWYER, S. D. High-accuracy large-step explicit Runge–Kutta (HALE-RK) schemes for computational aeroacoustics. *Journal of Computational Physics*, Orlando, v. 228, p. 3837–3850, 2009.

BUTCHER J. C. *The Numerical analysis of ordinary differential equations*: Runge-Kutta and general linear methods. New York: Wiley-Interscience, 1987.

BUTCHER J. C. *The numerical analysis of ordinary differential equations*. Chichester: John Wiley Sons, 2003.

CARPENTER, M. H.; KENNEDY C. A. Fourth-order 2N-storage Runge-Kutta schemes. NASA *Technical Memorandum*, Hampton, n. 109112, 1994.

KETCHESON, D. I. Runge-kutta methods with minimum storage implementations. *Journal of Computational Physics*, Orlando, v. 229, n. 5, p. 1763 – 1773, 2010.

NIEGEMANN, J.; DIEHL, R.; BUSCH, K. Efficient low-storage Runge–Kutta schemes with optimized stability regions. *Journal of Computational Physics*, Orlando, v. 231, p. 364–372, 2012.

WILLIAMSON, J. H. Low-storage Runge-Kutta schemes. *Journal of Computational Physics*, Orlando, v. 35, n. 1, p. 48–56, 1980.

Received: May 17, 2019
Accepted: Aug. 17, 2019