# Extrapolated Stabilized Explicit Runge–Kutta methods

J. Martín-Vaquero[†], A. Kleefeld [*]

University of Salamanca. Salamanca. E37008 Spain[†]

Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, 52425 Jülich, Germany[*]

September 20, 2018

## 1 Introduction

Traditionally classical explicit methods have not been used for stiff ordinary differential equations due to their stability limitations. However, very often, the dimension is high and the eigenvalues of the Jacobian matrix are known to be in a long narrow strip along the negative real axis. This situation typically arises when discretizing spatially parabolic equations or hyperbolic-parabolic equations such as advection-diffusion-reaction equations (with dominating diffusion or reaction). In this case, stabilized explicit Runge-Kutta methods were demonstrated to be very efficient (see [1, 3, 4, 7] and references therein).

Extrapolated Stabilized Explicit Runge-Kutta methods (ESERK) are proposed, in this work, to solve multi-dimensional non-linear partial differential equations (PDEs). For such methods it is necessary to evaluate the function $n_t$ times per step, but the stability region is $O(n_t^2)$. Hence, the computational cost is $O(n_t)$ times lower than for a traditional explicit algorithm. In that way stiff problems can be integrated by the use of simple explicit evaluations in which case implicit methods usually had to be used.

We first calculate the first-order SERK method. Later, we compute the numerical results of the initial value problem (IVP) by performing $n_i$ steps with step size $\Delta t_i$ to obtain $y_{\Delta t_i}(x_0 + h) := RE_{i,1}$ from $y(x_0)$. We do these calculations with this method for various length-step values $\Delta t_1 > \Delta t_2 > \Delta t_3 > \ldots$ (taking $\Delta t_i = \Delta t/n_i$, $n_i$ being a positive integer). This idea has been considered to develop fourth- (ESERK4, [6]) and fifth-order (ESERK5, [5]) ESERK method.

During the next years, we are working in two different issues:

[*]e-mail: jesmarva@usal.es, a.kleefeld@fz-juelich.de

i) We are planning to develop from second- to sixth-order codes based on ESERK methods, and combine all of them in one code. In several papers, including [6], the authors showed how depending on the prescribed tolerance, but also the stiffness of the problem (and also some functions or intervals considered), lower-order methods have better results sometimes, and in others it was necessary to obtain good approximations faster.

ii) Since we need $s$ stages to obtain the first-order stabilized explicit Runge-Kutta approximation ($RE_{1,1}$), the total number of function evaluations of the fourth-order method is $n_t = 10s$, and for the fifth-order scheme is $n_t = 15s$. To increase the speed of these higher-order methods, we are working on parallelized versions of the codes described in [5, 6].

In this work, we briefly explain how we derived sixth-order ESERK (ESERK6) methods and how we are parallelizing all these codes: ESERK4, ESERK5, and ESERK6 algorithms.

# 2 Stabilized Explicit Runge-Kutta methods

For the construction of these kinds of algorithms two problems need to be solved:

i) Finding stability functions with extended stability domains along the negative real axis;

ii) Finding explicit Runge-Kutta methods with those polynomials as stability functions.

## 2.1 Stability functions with extended stability domains

The main ingredient for these methods is a Chebyshev polynomial of the first kind:

$$T_s(x) = \cos(s \arccos(x)).$$

If we now consider

$$R_{s,p}(z) = \frac{T_s(w_{0,s,p} + w_{1,s,p} z)}{T_s(w_{0,s,p})}, \quad w_{0,s,p} = 1 + \frac{\mu_p}{s^2}, \quad w_{0,s,p} = \frac{T_s(w_{0,s,p})}{T_s'(w_{0,s,p})}, \quad (1)$$

($s$ being the number of stages and $p$ the order of the final extrapolated scheme) we obtain polynomials oscillating between $-\lambda_p$ and $\lambda_p$ in a region which is $O(s^2)$, and $R_{s,p}(z) = 1 + z + O(z^2)$.

The parameter $\lambda_p$ is always a value smaller than 1, which depends on the order of convergence, $p$, of the final extrapolated method, see [6]. For example, we calculated $\lambda_3 \leq 0.368008$, $\lambda_4 \leq 0.311688$, $\lambda_5 \leq 0.277923$, and $\lambda_6 \leq 0.25658$ numerically. We took $\mu_4 = 27/16$, $\mu_5 = 192/100$ and $\mu_6 = 208/100$ in Equation (1) to develop fourth-, fifth- and sixth-order schemes, respectively.

As for the new sixth-order schemes, the number of stages of the built Runge-Kutta methods were: $s = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,$

20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300, 350, 400, 450, 500, 600, 700, 800, 900, 1000, 1200, 1400, 1600, 1800, 2000, 2200, 2400, 2600, 2800, 3000, 3200, 3400, 3600, 3800, and 4000.

## 2.2 Derivation of the explicit Runge-Kutta method with those stability functions

We first construct the first-order methods for $s = qm$ using the following theorem:

**Theorem 1** *Let the stabilized explicit Runge-Kutta method be:*

$$\begin{aligned}
g_0 &= y_0, \\
g_1 &= g_0 + \alpha \cdot \Delta t \cdot f(g_0), \\
g_j &= 2g_{j-1} - g_{j-2} + 2\alpha \cdot \Delta t \cdot f(g_{j-1}) \quad j = 2, \ldots, m, \\
g_{m+1} &= g_m + \alpha \cdot \Delta t \cdot f(g_m), \\
g_j &= 2g_{j-1} - g_{j-2} + 2\alpha \cdot \Delta t \cdot f(g_{j-1}) \quad j = m+2, \ldots, 2m, \\
&\ldots \\
g_{(q-1)m+1} &= g_{(q-1)m} + \alpha \cdot \Delta t \cdot f(g_{(q-1)m}), \\
g_j &= 2g_{j-1} - g_{j-2} + 2\alpha \cdot \Delta t \cdot f(g_{j-1}) \quad j = (q-1)m+2, \ldots, s,
\end{aligned} \tag{2}$$

*and* $y_1 = \sum_{j=0}^{s} b_j g_j$ *where* $\alpha = \alpha_4 = 2/s^2$ *for the fourth-order method,* $\alpha = \alpha_5 = 100/(49s^2)$ *for the fifth-order method,* $\alpha_6 = 100/(47s^2)$*; and* $b_j$ *are the solutions of the linear system*

$$R_{s,p}(z) = b_0 T_0 + \sum_{j=1}^{q} \sum_{i=1}^{m} \left( b_{i+m(j-1)} T_i T_m^{j-1} \right), \tag{3}$$

*where* $T_i = T_i(1 + \alpha_p z)$ *are the shifted Chebyshev polynomials.*

*This algorithm has* $R_{s,p}(z)$ *as its stability function and, therefore, it is first-order accurate.*

Later extrapolated techniques are employed to increase the order of convergence of the methods, in our case we utilized the Aitken-Neville algorithm with the "harmonic sequence". Thus, for example, the previous fourth- (given as $RE_{4,4} = y(x_{n+1}) + O(h^4)$), and fifth-order ($RE_{5,5} = y(x_{n+1}) + O(\Delta t^5)$) schemes were calculated as:
- Fourth-order methods:

$$RE_{4,4} = \frac{64 y_{\Delta t/4}(x_{n+1}) - 81 y_{\Delta t/3}(x_{n+1}) + 24 y_{\Delta t/2}(x_{n+1}) - y_{\Delta t}(x_{n+1})}{6}.$$

- Fifth-order methods:

$$RE_{5,5} = \frac{625 y_{\Delta t/5}(x_{n+1}) - 1024 y_{\Delta t/4}(x_{n+1}) + 486 y_{\Delta t/3}(x_{n+1}) - 64 y_{\Delta t/2}(x_{n+1}) + y_{\Delta t}(x_{n+1})}{24}.$$

In the case of the new sixth-order schemes the Aitken-Neville algorithm provides us the following formula:

$$RE_{6,6} = 324/5 y_{\Delta t/6}(x_{n+1}) - 3125/24 y_{\Delta t/5}(x_{n+1}) +$$

$$+256/3 y_{\Delta t/4}(x_{n+1}) - 81/4 y_{\Delta t/3}(x_{n+1}) + 4/3 y_{\Delta t/2}(x_{n+1}) - 1/120 y_{\Delta t}(x_{n+1}). \quad (4)$$

*Example:* Let us consider the case $p = 6$ (order), $s = 4$ (number of stages), $m = 2$, $q = 2$, with $\alpha_6 = 100/(47s^2)$ for all the sixth-order methods.

(1.) We first calculate $R_4(z)$ taking $\mu_6 = 208/100$ in (1). Precisely, we obtain $w_{0,4,6} = 1.13$, and

$$w_{1,4,6} = \frac{T_4\,(1.13)}{T_4'\,(1.13)} = 0.136284,$$

and hence

$$R_4(z) = 1 + z + 0.25853z^2 + 0.02391z^3 + 0.00072083z^4.$$

(2.) Now, we can write $R_4(z)$ as a combination of the modified Chebyshev polynomials for $x = 1 + 100z/(47s^2)$:

$$R_4(z) = 0.14788\,T_0(x) - 0.82357\,T_1(x) + 0.51564\,T_2(x) +$$

$$+0.99979\,T_3(x) + 0.16025\,T_4(x).$$

(3.) The stabilized explicit Runge-Kutta first-order method (with $R_4(z)$ as stability function) is derived applying equation (2).

(4.) We utilize Richardson extrapolation to obtain the higher-order scheme. Let us suppose that $y_0 \approx y(x_0)$ is the solution previously obtained, and $\Delta t$ is the length step for the following iteration. Using the latter step, a first-order approximation is obtained, $S_{1,1} \approx y(x_0 + \Delta t)$. When we utilize $y_0$ and two steps of the first-order SERK scheme given in (3.), but with $\Delta t/2$, then we obtain $RE_{2,1}$, and so on. Finally we employ (4) to obtain the sixth-order approximation $RE_{6,6}$.

## 2.3 Parallelization of the ESERK schemes

The idea of the parallelization for ESERK schemes is very simple: it is possible to calculate at the same time $RE_{i,1} = y_{\Delta t/i}(x_{n+1})$ separately from $RE_{j,1} = y_{\Delta t/j}(x_{n+1})$ for different values of $i, j$. At the same time, we know that the computational cost of calculating $RE_{i,1}$ is proportional to the number of function evaluations necessary to calculate it: $i \times s$. Hence, when the final order of the ESERK method is even $p = 4$ or 6, we calculate $RE_{p,1}$ in one processor, $RE_{p-1,1}$ and $RE_{1,1}$ in another one, etc. And finally $RE_{p/2,1}$ in the last one.

If the final order of the ESERK method is odd, as with $p = 5$, we calculate $RE_{p,1}$ in one processor, $RE_{p-1,1}$ and $RE_{1,1}$ in another one, etc. For example, for $p = 5$, we calculate $RE_{5,1}$ in one processor, $RE_{4,1}$ and $RE_{1,1}$ in another processor, and $RE_{3,1}$ and $RE_{2,1}$ in a third one. In this way, the computational cost is proportional to $5s$ and not to $15s$ as in the sequential code.

### 2.4 Variable-step and number of stages algorithm

The step size estimation and stage number selection are similar to the ones given for other similar schemes, but it is necessary to change the formulae according to the order of the methods derived, see [5, 6] and references therein. First, we select the step size in order to control the local error and then, later we choose the minimum number of stages such that the stability properties are satisfied. The best results (for these extrapolated schemes) were obtained using techniques described in [2] for (traditional) extrapolated methods.

## 3 Numerical example and conclusions

Let us consider the following two-dimensional non-linear problem from combustion theory, see [8].

$$u_t = d\triangle u + \frac{R}{\alpha\delta}(1+\alpha-u)\mathrm{e}^{\delta(1-1/u)}, \tag{5}$$

defined on the unit square for $t \geq 0$. The problem is subjected to $u(x, y, 0) = 1$. For $t > 0$ we consider Dirichlet boundary condition $u = 1$ at $x = 1$, $y = 1$, but Neumann boundary condition at $x = 0$, $y = 0$. We used second-order schemes to approximate the Neumann condition. The parameter values in this problem are $d = 2.5$, $\alpha = 1$, $\delta = 20$, and $R = 5$. We used $N = 600$ equispaced nodes in each variable and solved first in the interval $[0, 1.45]$ and later from 1.45 to 1.48.

First, in Table 1, numerical results at $t_{end} = 1.45$ ($L_\infty$ errors) are shown in the upper part of the Table. In this interval the solution is very smooth as it is explained in [8]. We can clearly see how, for tolerances $10^{-6}$ and $10^{-8}$, and similar errors $\sim 10^{-5}, 10^{-6}$, the number of steps given by ESERK5 are smaller than with ESERK4, but the number of function evaluations and CPU times are bigger in this interval. It is more difficult to compare RKC with this table in this interval, however for moderate tolerances is faster than the other two codes.

However, we also solved this problem at $t_{end} = 1.48$, and calculated times, Nfe and number of steps in the interval $[1.45, 1.48]$. We show these differences in Table 1 (bottom part).

We show how the number of steps (with the three methods) is higher in the small interval $[1.45, 1.48]$ than previously in $[0, 1.45]$. It is caused because of the stiff solution in this interval (see [8]), and this motivates the use of variable-step algorithms with very stiff PDEs. However, the number of function evaluations, and CPU times do not grow in the same proportion; obviously, the reason is all these codes vary the number of stages to optimize the CPU times. Finally, in

| Interval | Tolerance | Method | max. err. | Time (s) | NFE | Steps |
|---|---|---|---|---|---|---|
| $[0, 1.45]$ | $10^{-6}$ | RKC | $0.5151_{-2}$ | 393.24 | 40264 | 123 |
| | | ESERK4 | $0.2598_{-4}$ | 2368.47 | 243728 | 51 |
| | | ESERK5 | $0.2208_{-4}$ | 2562.71 | 285750 | 34 |
| $[0, 1.45]$ | $10^{-8}$ | RKC | $0.2471_{-3}$ | 841.13 | 87300 | 545 |
| | | ESERK4 | $0.7085_{-6}$ | 3476.68 | 358539 | 122 |
| | | ESERK5 | $0.9726_{-7}$ | 5709.96 | 418466 | 75 |
| $[1.45, 1.48]$ | $10^{-6}$ | RKC | $0.1797_{0}$ | 57.75 | 9211 | 262 |
| | | ESERK4 | $0.9094_{-3}$ | 420.30 | 64712 | 112 |
| | | ESERK5 | $0.7583_{-3}$ | 707.97 | 74171 | 71 |
| $[1.45, 1.48]$ | $10^{-8}$ | RKC | $0.8426_{-2}$ | 212.01 | 21040 | 1259 |
| | | ESERK4 | $0.2288_{-4}$ | 668.13 | 105674 | 314 |
| | | ESERK5 | $0.3217_{-5}$ | 745.04 | 103212 | 177 |

Table 1: Maximal absolute error, CPU times, number of function evaluations, steps, and maximal steps for the methods RKC, ESERK4, and ESERK5 for $[0, 1.45]$ (up), and $[1.45, 1.48]$ (bottom).

this interval, for moderate errors $< 10^{-5}, 10^{-6}$ higher-order codes provide faster more accurate solutions. This motivates that we want to combine ESERK codes with different convergence orders into one final code.

## Acknowledgements

The authors acknowledge support from the University of Salamanca, through the grant ID2017/096.

## References

[1] A. Abdulle. Fourth order Chebyshev methods with recurrence relation. *SIAM J. Sci. Comput.*, 23(6):2041–2054, 2001.

[2] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I (2nd Revised. Ed.): Nonstiff Problems.* Springer-Verlag, New York, 1993.

[3] W. H. Hundsdorfer and J. G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations.* Springer, Berlin, 2007.

[4] D. I. Ketcheson and A. J. Ahmadia. Optimal stability polynomials for numerical integration of initial value problems. *Commun. Appl. Math. Comput. Sci.*, 7(2):247–271, 2012.

[5] J. Martín-Vaquero and A. Kleefeld. ESERK5: a fifth-order extrapolated stabilized explicit Runge-Kutta method. *Journal of Computational Physics (submitted)*, 2018.

[6] J. Martín-Vaquero and B. Kleefeld. Extrapolated stabilized explicit Runge–Kutta methods. *Journal of Computational Physics*, 326:141–155, 2016.

[7] B. Sommeijer, L. Shampine, and J. Verwer. RKC: An explicit solver for parabolic PDEs. *Journal of Computational and Applied Mathematics*, 88(2):315–326, 1997.

[8] J. G. Verwer. Explicit Runge-Kutta methods for parabolic partial differential equations. *Appl. Numer. Math.*, 22(1–3):359–379, 1996.