ESERK5: a Fifth-Order Extrapolated Stabilized Explicit Runge-Kutta Method

J. Martín-Vaquero[a,*], A. Kleefeld[b]

[a] *ETS Ingenieros industriales, Universidad de Salamanca. E37700, Bejar, Spain*
[b] *Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre. Wilhelm-Johnen-Straße, 52425 Jülich, Germany*

Abstract

A new algorithm is developed and analyzed for multi-dimensional non-linear parabolic partial differential equations (PDEs) which are semi-discretized in the spatial variables leading to a system of ordinary differential equations (ODEs). It is based on fifth-order extrapolated stabilized explicit Runge-Kutta schemes (ESERK). They are explicit methods, and therefore it is not necessary to employ complicated software for linear or non-linear system of equations. Additionally, they have extended stability regions along the negative real semi-axis, hence they can be considered to solve stiff problems coming from very common diffusion or reaction-diffusion problems.

Previously, only lower-order codes (up to fourth-order) have been constructed and made available in the scientific literature. However, at the same time, higher-order codes were demonstrated to be very efficient to solve equations where it is necessary to have a high precision or they have transient zones that are very severe, and where functions change very fast. The new schemes allow changing the step length very easily and with a very small computational cost. Thus, a variable step length, with variable number of stages algorithm is constructed and compared with good numerical results in relation to other well-known ODE solvers.

Keywords: Higher-order schemes, Multi-dimensional partial differential equations, Stabilized explicit Runge-Kutta methods, Variable-step length ODE solvers

1. Introduction

In this manuscript, we explain how to develop fifth-order extrapolated stabilized explicit Runge-Kutta methods (ESERK) to solve systems of ODEs coming from multi-dimensional non-linear partial differential equations (PDEs). More

*Corresponding author
Email addresses: jesmarva@usal.es (J. Martín-Vaquero), a.kleefeld@fz-juelich.de (A. Kleefeld)

precisely, we want to solve efficiently parabolic equations, usually with reaction and diffusion terms in the equations after semi-discretization in space. Due to reaction terms, solutions may vary fast in some regions, and therefore it is necessary to utilize algorithms with high accuracy [2, 13, 14, 15].

This work is not only an extension of the fourth-order extrapolated stabilized explicit Runge-Kutta methods (ESERK4) by [31]. The new contribution of this paper is the derivation of necessary stability conditions for a fifth-order algorithm. In sum, 49 fifth-order algorithms are constructed for different stages. For this purpose, roughly 15000 coefficients have been obtained for the first time. These algorithm are combined in a code named ESERK5 (fifth-order extrapolated stabilized explicit Runge-Kutta methods) with variable step size, which is freely available at Github (https://github.com/kleefeld80/ESERK5). Additionally, ESERK5 provides extended stability regions, including some algorithms which are stable in $[-l_s, 0]$ with l_s bigger than 10^6, which is clearly bigger than other similar methods such as DUMKA or ROCK (around 40 times bigger than with ROCK4). Therefore, the new library might be a much more stable and accurate competitive alternative to such existing libraries, since fifth-order libraries do not exist with extended stability regions. ESERK5 is also shown to provide stable and accurate numerical examples for well-known testing scenarios with a very good performance. There is a need for new higher-order algorithms because some PDEs require very accurate solutions, and it has been demonstrate that in some phases "a high order is advocated" see [20]. This is also a small step to provide a possibility to combine this library with Radau5 like it has been done in [15] with ROCK4 and Radau5, but for fifth-order schemes now. Additionally, it is clear that the new code ESERK5 is able to obtain faster more accurate solutions than the previous code ESERK4, and therefore it might be interesting developing other ESERK codes, with different order of convergence, and combine all of them in one script, able to change the order. However, for this purpose, it is necessary to continue with the work begun in [31] and the current work on ESERK5.

These types of problems are common in a large amount of applications such as atmospheric phenomena, biology, chemistry, combustion problems, financial mathematics, fluid mechanics, industrial engineering, laser modelling, malware propagation, medicine, molecular dynamics, nuclear kinetics, etc., see [3, 6, 12, 13, 14, 15, 17, 20, 30, 43], to mention a few.

They are non-linear PDEs where the non-linearity cannot be neglected because this would affect the solution of these problems. Hence, habitually these partial differential equations are transformed into systems of ordinary differential equations (ODEs), this is done through spatial discretizations. Although functions and solutions are frequently smooth, in many of these cases, equations are usually very stiff. Thus, traditional explicit methods are computationally very expensive (due to limitations of the length step). On the other hand, traditional implicit stable Runge-Kutta or BDF schemes (as [11, 17, 18]) have important difficulties to solve two- and three-dimensional PDEs, since these non-linear systems of ODEs have many unknowns.

Other types of algorithms can be considered to approximate these large

systems of ODEs such as: exponential fitting or ETD schemes [9, 19, 21, 24, 32] or implicit-explicit algorithms (for example [8, 34]) in order to obtain accurate solutions with lower computational cost. In all these cases explained above it is necessary to approximate exponentials of very large matrices or solving a large number of linear or even non-linear equations at each step. Therefore, these codes are very expensive or there might be important memory demanding problems.

However, very often, 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 parabolic equations with dominating diffusion and/or reaction terms.

In this case, Runge-Kutta-Chebyshev, stabilized explicit Runge-Kutta (SERK) or similar methods were demonstrated to be very efficient, see [2, 20, 22, 27, 33, 39, 40, 41]. They are explicit algorithms where 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. Since these methods are explicit, the computational cost is given by the number of function evaluations, it is not necessary to solve any linear or non-linear system of equations.

Therefore, they are especially well-suited for the method of lines (MOL) discretizations of parabolic non-linear multi-dimensional PDEs (where implicit methods are very expensive). However, they can be used only to solve problems where all eigenvalues of the Jacobian are negative real numbers or they are close to the negative real axis.

So far, stabilized explicit RK methods have been built only up to fourth-order. Hence, similar higher-order methods might be interesting for problems where it is necessary to obtain high precision. Additionally, they could be combined with Radau5, for example, and this is interesting to obtain high-accurate implicit-explicit methods for reaction-diffusion problems. There are some papers on fourth-order ROCK4 schemes combined with Radau5 for several problems [13, 14, 15]. Recent work in the direction of deriving higher-order codes using factorized Runge-Kutta-Chebyshev (FRKC) polynomials, or Runge-Kutta with Gegenbauer polynomials has also been done by O'Sullivan (see [35, 36].

Additionally, our final project is creating a software based on extrapolated stabilized explicit Runge-Kutta methods similar to ODEX (see [23] for a parallelized version), that allows changing the order, the number of stages, and the length step. Thus, we will need to create a series of schemes with different order and number of stages and combine them in one script.

In [31], the general theory to obtain extrapolated stabilized explicit Runge-Kutta methods was proposed, but only fourth-order schemes were derived. It was also briefly explained that, once first-order methods with some specific and necessary stability conditions are derived, then it is possible to build second-, third- and fifth-order algorithms in a similar way (p. 143). However, none of these coefficients were obtained, except for the fourth-order codes. In this manuscript, it is now explained which are these necessary stability conditions for fifth-order algorithms, and we give details about how fifth-order stabilized

explicit Runge-Kutta methods are derived in Section 3. We built algorithms for $s = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300, 350, 400, 450, 500, 600, 700, 800, 900, 1000, 1200, 1400, 1600, 1800, 2000$.

In the next section, we show how to develop extrapolated explicit schemes and also which are the stability properties that we are looking for these explicit methods. Later, the fixed step algorithms for ESERK5 are built, and we also show their properties. With these methods, in Section 4, the variable step length code is developed. It is tested in Section 5, comparing the new algorithm with other excellent ODE solvers such us RKC, ROCK2, IRKC, and PIROCK. We focus our numerical experiments on reaction-diffusion equations of the form

$$u_t = \Delta u + g(t, u),$$

where $g(t, u)$ is a non-linear reaction term. Finally, some conclusions and future goals are addressed.

2. Extrapolated explicit methods

There have been several types of stabilized explicit Runge-Kutta methods (also called Runge-Kutta-Chebyshev) in the scientific literature. And some of them have difficulties with problems where any of the eigenvalues are very large, especially the schemes with order higher than two. Actually, in [28], Lebedev and Finogenov explained that this kind of algorithms have two kind of problems: (i) internal stability and (ii) propagation of errors. They tried to solve them by finding special permutation of the roots of the polynomials. This procedure reduces this problem when the order of the polynomial is low. These permutations are also briefly explained by Hairer and Wanner [17]. On page 34, they stated that the propagation of errors should be as small as possible.

Thus, Sommeijer, Shampine, and Verwer developed their Runge-Kutta-Chebyshev (RKC) methods in [40]. These schemes show an important improvement to minimize these internal instability and propagation errors. They used a three-term recurrence procedure to build their algorithms, but their RKC schemes are only second-order in time.

Later, Abdulle used a similar three-term recurrence scheme combined with a composition procedure to build ROCK2 (second-order) and ROCK4 (fourth-order) methods. These codes are excellent for many mildly stiff ODEs [2, 4] coming from spatial discretizations of non-linear PDEs. However, they have some difficulties with the amplification of the errors in the last two (with ROCK2) or four (with ROCK4) stages of the method, as it is explained in [1], in §4.4 or in [20], in §5.2. Actually, Hundsdorfer and Verwer reported amplification factors of $O(10^9)$ for ROCK2 with 200 stages in [20].

Recently, in [35], O'Sullivan derived higher-order factorized Runge-Kutta-Chebyshev schemes. They have good internal stability properties, but he is currently working on improving the codes with order higher than two. A guide and some scripts for the second-order methods can be found on the web page

www.maths.dit.ie/frkc. As far as we know he is working on codes that are fourth- and sixth-order in time.

Some Runge-Kutta-Chebyshev methods have been obtained using Richardson extrapolation by various authors, see [7, 37, 38] and references therein, but these methods are only second-order.

Some stabilized explicit Runge-Kutta algorithms (SERK) were derived in [25, 26, 29, 30]. They are also based on three-term recurrence formulae. These SERK schemes are only second-order. However, they can be developed from polynomials whose first terms are $1+z+\frac{z^2}{2}+\frac{z^3}{3!}+\frac{z^4}{4!}+\ldots$, hence they might be useful to build higher-order schemes. In [31], SERK schemes were combined with extrapolation techniques to build a fourth-order ESERK scheme (ESERK4). So far, only fourth- or lower-order methods were constructed, to the best of our knowledge. In this manuscript, we are illustrating how to construct a method that is of fifth-order in time, in a similar way as in [31] ESERK4 was derived. Now, we will show the main ingredients of this idea, and later we will analyze the new schemes.

In Section 3, 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 h_i to obtain $y_{h_i}(x_0+h) := S_{i,1}$ from $y(x_0)$. We do these calculations with this method for various values $h_1 > h_2 > h_3 > \ldots$ (taking $h_i = h/n_i$, n_i being a positive integer). Using the Aitken-Neville algorithm:

$$S_{j,k+1} = S_{j,k} + \frac{S_{j,k} - S_{j-1,k}}{(n_j - n_{j-k}) - 1},$$

higher-order schemes might be derived.

For example, the fifth-order method can be computed as:

$$S_{5,5} = \frac{S_{1,1} - 64S_{2,1} + 486S_{3,1} - 1024S_{4,1} + 625S_{5,1}}{24} =$$

$$= \frac{625y_{h/5}(x_0+h) - 1024y_{h/4}(x_0+h)}{24} +$$

$$+ \frac{486y_{h/3}(x_0+h) - 64y_{h/2}(x_0+h) + y_h(x_0+h)}{24}.$$

Since we need s stages to obtain the first-order stabilized explicit Runge-Kutta approximation ($S_{1,1}$), the total number of function evaluations of the fifth-order scheme is $n_t = 15s$ (there are $15s$ stages per step). Since we will require special stability conditions, we will utilize the well-known idea of Chebyshev polynomials to obtain them.

2.1. Shifted Chebyshev polynomials

To obtain the first-order schemes we can use Chebyshev polynomials of the first kind of order s (s = stages) which are defined by the recursion:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_s(x) = 2xT_{s-1}(x) - T_{s-2}(x).$$

Thus, if we consider

$$R_s(z) = \frac{T_s(w_{0,s} + w_{1,s} z)}{T_s(w_{0,s})}, \quad w_{0,s} = 1 + \frac{\mu}{s^2}, \quad w_{1,s} = \frac{T_s(w_{0,s})}{T_s'(w_{0,s})}, \tag{1}$$

(s being the number of stages of the first-order method, z is a function of x depending on this values s) we obtain polynomials oscillating between $-\lambda$ and λ in a region which is $O(s^2)$, and $R_s(z) = 1 + z + O(z^2)$ (as it is explained, for example in [17]). The parameter μ is a number precisely chosen to obtain that $|R_s(z)| < \lambda$. Later, we will calculate λ as 0.277923 and $\mu = 192/100$.

As it was explained in [31], in our case, if we denote the polynomial of the fifth-order extrapolated method as $P_{5s}(z)$, then:

$$P_{5s}(z) = \frac{R_s(z) - 64(R_s(z/2))^2 + 486(R_s(z/3))^3 - 1024(R_s(z/4))^4 + 625(R_s(z/5))^5}{24}. \tag{2}$$

Since

$$|P_{5s}(z)| \leq \frac{|R_s(z)| + 64|R_s(z/2)|^2 + 486|R_s(z/3)|^3 + 1024|R_s(z/4)|^4 + 625|R_s(z/5)|^5}{24},$$

we can take $\lambda \leq 0.277923$ to guarantee that $|P_{5s}(z)| \leq 0.95$. We have taken 0.95 as usual in these Runge–Kutta–Chebyshev (or stabilized explicit Runge–Kutta methods); we might have chosen 0.99 (always < 1 for stability reasons, see [17]) in this case, because the stability regions of $P_{5s}(z)$ are wider than those of $R_s(z)$, however differences are small (in the length of the final stability regions of $P_{5s}(z)$). Later, in Section 3, we will show that both, $|R_s(z)| < 1$ and $|P_{5s}(z)| < 1$ when $z \in [-0.98s^2, 0]$.

Hence, we will write $R_s(z)$ as a combination of the shifted Chebyshev polynomials:

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

where $T_i = T_i(x) = T_i(1+100z/(49s^2))$ are these shifted Chebyshev polynomials (and for some values of q that we will discuss later). In this manuscript, we have taken $\alpha = 49/100$ (for $T_i(1 + z/(\alpha s^2))$), since we want that the internal stability regions include the interval $[-0.98s^2, 0]$, and it is well known that shifted Chebyshev polynomials obtain values smaller than 1 in $[-2\alpha s^2, 0]$.

In this way, $R_s(z)$ is expressed as a linear combination of shifted Chebyshev polynomials. In the following section, we will first derive methods that have $T_i(1 + 100z/(49s^2))$ as stability functions (hence the internal stability is large), and the linear combination of these schemes will become a first-order stabilized method with the desired $R_s(z)$ as stability function. Finally, fifth-order algorithms are obtained with $P_{5s}(z)$ using extrapolation.

3. Deriving fixed-step algorithms

For this paper, a code called ESERK5 is developed. It contains fifth-order extrapolated SERK schemes with n_t up to $30,000$ stages (s up to 2000). The way to obtain the fixed-step algorithms is explained in the following lines:

1. First of all, we obtain the polynomials $R_s(z)$ taking $\mu = 192/100$ in (1). With this value for μ, the reader can check that $|R_s(z)| < 0.277923$ for $s \geq 8$, which guarantees that $|P_{5s}(z)| \leq 0.95$ when $z \in [-0.98s^2, 0]$. For $1 \leq s \leq 7$, $|R_s(z)|$ reach values over $\lambda = 0.277923$, however $|P_{5s}(z)| \leq 0.95$ for all these s and z values.

   We need to use extra precision when s is large. We calculated these polynomials (with Mathematica) for $s = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20$ (in all these cases $m = 2$ and $q = \lfloor s/m \rfloor$), $s = 25, 30, 35, 40, 45, 50$ (in all these cases $m = 5$ and $q = s/m$), $s = 60, 70, 80, 90, 100$ (in all these cases $m = 10$ and $q = s/m$), $s = 150, 200, 250, 300, 350, 400, 450, 500$ (in all these cases $m = 50$ and $q = s/m$), $s = 600, 700, 800, 900, 1000$ (in all these cases $m = 100$ and $q = s/m$), and $s = 1200, 1400, 1600, 1800, 2000$ (in all these cases $m = 200$ and $q = s/m$).

2. Later, we write $R_s(z)$ as a combination of the modified Chebyshev polynomials, following equation (3). It is necessary to solve a linear system with a lower triangular matrix.

3. The stabilized explicit Runge-Kutta method is obtained through a three-term recurrence formula:

$$
\begin{aligned}
g_0 &= y_0, \\
g_1 &= g_0 + \alpha h f(g_0), \\
g_j &= 2g_{j-1} - g_{j-2} + 2\alpha h f(g_{j-1}) \quad j = 2, \ldots, m, \\
g_{m+1} &= g_m + \alpha h f(g_m), \\
g_j &= 2g_{j-1} - g_{j-2} + 2\alpha h f(g_{j-1}) \quad j = m+2, \ldots, 2m, \\
&\ldots \\
g_{qm+1} &= g_{qm} + \alpha h f(g_{qm}), \\
g_j &= 2g_{j-1} - g_{j-2} + 2\alpha h f(g_{j-1}) \quad j = qm+2, \ldots, s,
\end{aligned}
\tag{4}
$$

   and

$$
y_1 = \sum_{j=0}^{s} b_j g_j \tag{5}
$$

   where $\alpha = 100z/(49s^2)$ and b_j are the solutions of the linear system (3). $S_{1,1} = y_1$ for a given h, $S_{2,1} = y_2$ for $h/2$, ..., $S_{5,1} = y_5$ for $h/5$.

4. The fifth-order method can be computed as:

$$S_{5,5} = \frac{S_{1,1} - 64S_{2,1} + 486S_{3,1} - 1024S_{4,1} + 625S_{5,1}}{24}. \tag{6}$$

Example: Let us consider the case $s = m = 2$, q is therefore 1.

(1.) We first calculate $R_2(z)$ taking $\mu = 192/100$ in (1), precisely, we obtain $w_{0,2} = \frac{37}{25}$, and

$$w_{1,2} = \frac{T_2\left(\frac{37}{25}\right)}{T_s'\left(\frac{37}{25}\right)} = \frac{2113}{3700},$$

and hence

$$R_2(z) = 1 + z + \frac{2113}{10952}z^2.$$

It is not really necessary to calculate $P_{10}(z)$ to derive our algorithm, however we can do it for studying the stability regions. We only need to apply equation (2).

(2.) Now, we can write $R_2(z)$ as a combination of the modified Chebyshev polynomials:

$$R_2(z) = \frac{2077539}{13690000}\,T_0(x) + \frac{1634787}{3422500}\,T_1(x) + \frac{5073313}{13690000}\,T_2(x),$$

with $x = 1 + 100z/(49s^2)$.

(3.) The stabilized explicit Runge-Kutta first-order method (with $R_2(z)$ as stability function) is derived applying equation (4).

(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 h is the length step for the following iteration. Using the latter step, a first-order approximation is obtained, $S_{1,1} \approx y(x_0 + h)$. If we utilize y_0 and two steps of the first-order SERK scheme given in (3.) with $h/2$, then we would obtain $S_{2,1}$, and so on. Finally

$$S_{5,5} = \frac{S_{1,1} - 64S_{2,1} + 486S_{3,1} - 1024S_{4,1} + 625S_{5,1}}{24}$$

is a fifth-order approximation with $P_{10}(z)$ as stability function.

Some readers might wonder why these $q > 1$ values appear. The reason is decreasing those propagation of errors mentioned above. As it was explained, some higher-order stabilized explicit methods suffer from these problems (see [17, 20, 28]). Hundsdorfer and Verwer reported amplification factors of $O(10^9)$ for ROCK2 with 200 stages, but those values are higher for ROCK4, and DUMKA methods have similar difficulties.

In this aspect, the three-term recurrence proposed for RKC is a great advance. Hundsdorfer and Verwer included the study of this concept for RKC in

the chapter V.1.3 [20], and we can use a similar procedure to study these values for ESERK schemes. Hundsdorfer and Verwer explained in page 431 that, with the three recurrence formula over the full s-stages,

$$|Q_{sj}| \leq k(s-j-1)(1+C\varepsilon),$$

and therefore errors obtained with RKC satisfy

$$||e_{n+1}||_2 \leq ||e_n||_2 + \frac{1}{2}s(s+1)K \max_j ||r_j||_2,$$

w_{nj} defined as in Eq. (1.13) [20], $\tilde{w}_{nj}$ defined as in Eq. (1.14), Q_{jk}, e_{nj} also as they were defined in page 426.

With a similar goal, we choose m, q to make the coefficients $b_j \sim O(1)$. For us

$$e_{n+1} = R_s(\tau A)e_n + \sum_{j=1}^{s} b_j Q_{sj}(\tau A)r_j,$$

(e_{n+1} varies because we obtain our Runge-Kutta method from Eqs. (4) and (5) in our paper). In our case, the internal stability region at stage $j = vm + r$ is given by $(T_m(z))^v T_r(z)$. Therefore, in a similar way as it was demonstrated in [20]:

$$|Q_{sj}| \leq k(m-r-1)(1+C\varepsilon),$$

(because $|T_m(z)| < 1$ for $z \in [-0.98s^2, 0]$), and therefore

$$||e_{n+1}||_2 \leq ||e_n||_2 + K\sum_{j=0}^{s} b_j(m-r+1)||r_j||_2.$$

Whenever $b_j \sim O(1)$, with the procedure proposed for ESERK schemes, propagation of errors grow with $s \times m = m^2 \times q < s^2$. This is why we think that this procedure reduces slightly the propagation of errors. Actually, we think that moderate m values help to decrease propagation of errors; however, for large s values, if m is small we checked that b_j grew significantly and there is no improvement in this technique. Thus, when we developed SERK and later ESERK schemes, we have chosen m in such a way that the coefficient of z^s in $R_s(z)$ was similar (same order) to the coefficient of z^s in $(T_m(z))^q$.

In this way, it is possible to obtain fifth-order extrapolated schemes with large stability regions near the real and negative semi-axis. In Fig. 1, we show the values that $R_s(z)$ (the stability function of the first-order stabilized explicit Runge-Kutta scheme) and $P_{5s}(z)$ (the stability polynomial of the extrapolated corresponding method) reach in this semi-axis (for $s = 10$ and 20). These plots demonstrate that these new algorithms satisfy the expected properties.

Additionally, since $\mu = 192/100$, these stability regions have width different from 0 near the semi-axis. This means that the extrapolated schemes might be able to compute efficiently some parabolic equations with (small enough) advection terms. In Fig. 2, plots of some (for $s = 7$ and $s = 14$) stability regions in the complex plane are provided.

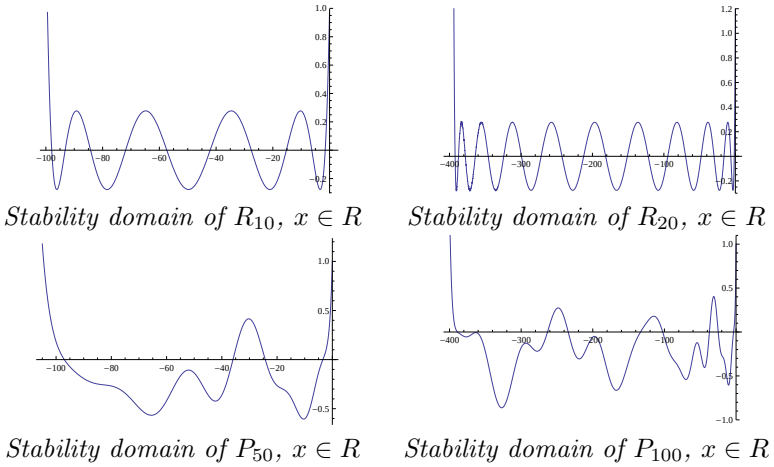

Figure 1: Comparison of the stability regions of polynomials R_{10} and its extrapolated P_{50}, and R_{20} and its extrapolated P_{100}, in the real and negative semi-axis.

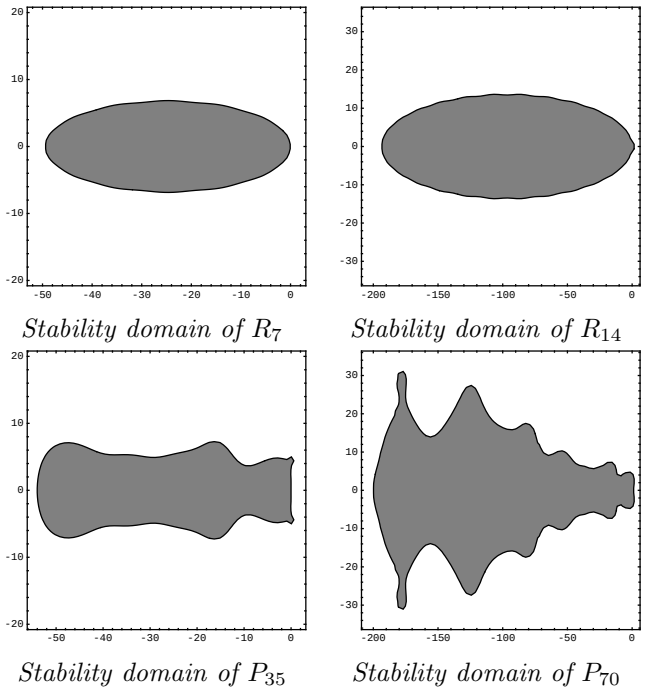

Figure 2: Comparison of the stability regions (in gray) of polynomials R_7 and its extrapolated P_{35}, and R_{14} and its extrapolated P_{70}, in the complex plane.

It is clear that $P_{5s}(z)$ polynomials have larger and (in general) wider stability regions than $R_s(z)$. We can see this in Table 1, where it is explained how the stability regions in the real and negative semi-axis grow.

Table 1 specifies the values l_s^* and $d_s^* = l_s^*/s^2$, l_s and $d_s = l_s/(15s)^2$ for which the polynomials $R_s(z)$ and $P_{5s}(z)$ are stable in $[-l_s^*, 0]$ and $[-l_s, 0]$, respectively. We did it for $s = 20, 100, 400$ and 2000 to check how d_s converges for large s values.

s	$n_t = 15s$	l_s^*	d_s^*	l_s	d_s
20	300	393.737	0.984343	398.884	0.004432
100	1500	9810.2	0.981020	9816.7	0.004363
400	6000	156942	0.980888	156948	0.004360
2000	30000	3923507	0.980877	3923513	0.004359

Table 1: Stability parameters of $R_s(z)$ and $P_{5s,2}(z)$ in the real and negative semi-axis.

As we can check in Table 1, l_s^* converges to $\sim 0.9808s^2$ (for large s values), hence $\alpha = 100z/(49s^2)$ is a good choice. l_s is slightly larger as can be checked in Table 1, $l_s \sim 0.004359n_t^2$ when n_t takes very large values. These stability bounds l_s are clearly shorter (for a given n_t^2 value) than others for some second-order schemes, hence RKC or ROCK2 are more efficient methods for large tolerances. But higher-order schemes are very interesting when small errors are required.

With all of this, we obtain the following result:

Theorem 1. *For the extrapolated stabilized explicit Runge-Kutta method derived through equations (4) and (6), we have the following stability and consistency properties:*

(i) It is stable in the interval $[-0.98s^2, 0]$ *(not only, but includes this region), with* $s \leq 2000$. *Additionally, the internal stability, at all the stages, includes this region. And there is not any propagation of errors at any stage.*

(ii) Since the value $S_{j,k}$ *represents a numerical method of order* k*, the scheme obtained with equation (6) converges with fifth-order, whenever the numerical method is stable, and the right hand term in the system of ODEs is six times continuously differentiable,* $\mathcal{C}^6$.

Proof. (i) is the result of how equation (4) was derived, and the properties of shifted Chebyshev polynomials described in this and the previous sections.

The demonstration of (ii) can be done in the same fashion as Theorems 9.1 and 9.2 in [16]. In this case, $p = 1$, i.e. $S_{1,1}$ is first-order (the proof is similar to the one given in [25], because we already demonstrated that $R_2(z) = 1+z+\ldots$). Hence, $S_{5,5}$ converges with fifth-order, whenever the numerical method is stable, and $f(t, u)$, the right hand term in the system of ODEs ($u'(t) = f(t, u)$), is $\mathcal{C}^6$. □

This theorem involves high-order convergence for both, linear and nonlinear problems. Also, we obtain a similar remark as in [16], p. 225: a great advantage

of this procedure (to create methods with extrapolation) is that it provides a sequence of embedded methods, and allows some estimates of the local error and strategies for variable order.

We already created a family of fourth-order methods in [31], and fifth-order schemes in this article, and our idea is creating more families and combine all of them in one algorithm.

4. Deriving variable-step and number of stages algorithm

The step size estimation and stage number selection are very similar to the ones obtained for the ESERK4 algorithm described in [31]. First, we select the step size in order to control the local error and then, choose the minimum number of stages such that the stability properties are satisfied.

4.1. Step size selection

The best results (for these extrapolated schemes) were obtained using techniques described in [16] for (traditional) extrapolated methods:

$$h_{new} = h_{old} \min\left(\text{facmax}, \max\left(\text{facmin}, \text{fac} \cdot (1/err)^{1/5}\right)\right), \tag{7}$$

with fac= 0.8, facmax= 10 (except after a rejection), facmin= 10^{-3}.

A comparison between the estimated error and the prescribed tolerance denoted by err is calculated as usual:

$$err = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{(S_{5,5} - S_{5,4})_i}{sc_i}\right)^2}, \tag{8}$$

with

$$sc_i = \left(Atol_i + \max\left(|y_{0,i}|, |S_{5,5,i}|\right) \cdot Rtol_i\right)/2, \tag{9}$$

$y_{0,i}$ being the i-th component of the solution at the previous step, $S_{5,5,i}$ the i-th component of the solution obtained through the extrapolation technique (sc would be a weighted mean of $Atol$ and $Rtol$).

$(S_{5,5} - S_{5,4})_i$ is the i-th component of the estimation of the error, which can be calculated as

$$\begin{aligned} S_{5,5} - S_{5,4} = \frac{S_{1,1} - 32S_{2,1} + 162S_{3,1} - 256S_{4,1} + 125S_{5,1}}{24} = \\ = 125/24 y_{h/5}(x_0 + h) - 32/3 y_{h/4}(x_0 + h) + \\ +27/4 y_{h/3}(x_0 + h) - 4/3 y_{h/2}(x_0 + h) + y_h(x_0 + h)/24. \end{aligned} \tag{10}$$

Additionally, we try to decrease the number of rejected steps when solving problems with non-smooth data. Hence, we employ a technique also used in [26]:

$$\frac{h_{n+i+1}^{(j)}}{h_{n+i}^{(k)}} \leq 1$$

for the two steps following the rejection ($i = 0, 1$) and

$$\frac{h_{n+i+1}^{(j)}}{h_{n+i}^{(k)}} \leq 2.5$$

for the three steps after that ($i = 2, 3, 4$), unless the interval where there could be jumps has passed. When the risk of rejections has decreased we allow again that

$$\frac{h_{n+i+1}}{h_{n+i}} \leq 10.$$

4.2. Stage number selection

We utilize, as usual in other Chebyshev codes (or stabilized explicit methods), a family of fifth-order methods with different numbers of stages, and we choose the best number of stages according to the stability regions of the algorithm.

In each step, we first select the step size in order to control the local error as stated previously, then we select the number of stages so that the stability property is satisfied

$$s > \sqrt{\frac{h_{new}\rho\left(\frac{\partial f}{\partial y}\right)}{0.98}}, \tag{11}$$

where $\rho\left(\frac{\partial f}{\partial y}\right)$ is a bound for the spectral radius (the largest eigenvalue in absolute value of the Jacobian of the function $f(y)$) and $0.98s^2$ is the estimate of the bound of the stability interval.

For the estimation of the spectral radius several procedures have traditionally been considered. If it is not possible to get an estimate of the spectral radius easily, then a non-linear power method (see [40], for example) is usually considered. Another way is to use the Gershgorin theorem

$$\rho\left(\frac{\partial f}{\partial y}\right) \leq \max_{i=1,\ldots,neqn}\left(-a_{ii} + \sum_{j=1,j\neq i}^{neqn} |a_{ij}|\right).$$

The new code can employ any of them depending on the considered test problem.

5. Numerical experiments and comparisons

As it was commented before, there are a few codes for these kinds of partial differential equations. In this section, we are trying to give some comparisons of the new code with other very well-known algorithms traditionally used. We employed a Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz (PC with 8 cores) to create the numerical results. We used the gfortran compiler version 4.8.1. This code and the numerical examples are available at the web page https://github.com/kleefeld80/ESERK5

We are comparing the new code with ESERK5 (the sequential algorithm) with some codes proposed for diffusion and/or reaction-diffusion as RKC, ROCK4, ESERK4, IRKC, and PIROCK, all written in Fortran (see [5] and references therein).

ROCK4 and PIROCK, including some examples are freely available at the address http://anmc.epfl.ch. RKC, with several drivers, can be freely downloaded at http://www.netlib.org/ode/.

In this manuscript, we are not trying to show that the new code ESERK5 is always better compared to the others. Actually all the solvers above are excellent choices in general. Instead, we will try to show some advantages and disadvantages of using any of the methods above, and we will try to show that, when it is necessary to obtain small errors, it might be interesting using ESERK algorithms and/or deriving new schemes with similar properties.

As usual, errors were calculated only at t_{end} with infinity norm for the system of ODEs (this is also called temporal error). As in most of articles on this topic we used second-order approximations in space for the spatial semi-discretization, because we are only trying to understand the behaviour of the codes for very large systems of ODEs. Hence, errors due to spatial discretizations might be larger than some temporal errors for small values *tol*, however these codes are able to use higher-order discretizations in space with good results as it was explained in [26].

Test 1: A 1D diffusion model

This is a simple diffusion equation taken from [10]. The problem under consideration is given by

$$u_t = u_{xx}, \quad x \in [0,1],$$

with initial condition $u(0,x) = a\sin(\sqrt{2}x) - \sin(x)$ and boundary conditions $u(t,0) = 0$, $u(t,1) = a\mathrm{e}^{-2t}\sin(\sqrt{2}) - \mathrm{e}^{-t}\sin(1)$, where $a = \cos(\sqrt{2})/(\sqrt{2}\cos(2^{-1/2}))$.

In [10], the authors considered the variables $y_i(t) = u(t, i/(N+1)), i = 1,\ldots,N$, and they spatially discretized the problem in the following way:

$$y'(t) = (N+1)^2 \begin{pmatrix} -2 & 1 & 0 & \ldots & 0 & 0 \\ 1 & -2 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -2 & 1 \\ 0 & 0 & 0 & \ldots & 1 & -2 \end{pmatrix} y(t) + (N+1)^2 \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \phi(t) \end{pmatrix}$$

with $\phi(t) = a\mathrm{e}^{-\nu t}\sin(\sqrt{2}) - \mathrm{e}^{-\mu t}\sin(1)$ and

$$\mu = 2(N+1)^2\left(1 - \cos\left(\frac{1}{N+1}\right)\right), \quad \nu = 2(N+1)^2\left(1 - \cos\left(\frac{\sqrt{2}}{N+1}\right)\right).$$

The exact solution of the discretized (in space) problem is

$$y_i(t) = a\mathrm{e}^{-\nu t}\sin\left(\frac{\sqrt{2}i}{N+1}\right) - \mathrm{e}^{-\mu t}\sin\left(\frac{i}{N+1}\right),$$

and we have the property that $\mu \to 1$ and $\nu \to 2$ as $N \to \infty$.

This is a simple example to support the theoretical results previously demonstrated in Theorem 1. Here, we have considered $N = 99$ for which μ and ν are approximately 0.9999916666947328 and 1.9999666668879534, respectively. With this value of N, $\rho\left(\frac{\partial f}{\partial y}\right)$ is bounded by 39990.1. Finally, we obtain the numerical results with all the methods derived above (for all s values) for several different $k = \Delta t$ values to check the fifth-order convergence when stability is obtained.

Whenever $39990.1k < 0.98s^2$, Theorem 1 states that the schemes are stable and should converge with fifth-order. In this experiment we chose $k_1 = 0.004$, $k_2 = 0.02$, and $k_3 = 0.001$, hence it means that all the methods with $s > 12.78$ are stable for all these k values. And any scheme with $s > 9.03$ is stable when $k \leq 0.002$. In Table 2, numerical errors are shown for $s = 10, 40, 150$ using $t = 1$ and $x = 1/2$. An estimation of the numerical convergence is obtained calculating $\log_4\left(\frac{err_{k_1}}{err_{k_3}}\right)$.

Method	$s = 10$	$s = 40$	$s = 150$
$k = 0.004$	1.09078×10^{974}	9.23506×10^{-10}	6.19622×10^{-10}
$k = 0.002$	3.37361×10^{-12}	1.15327×10^{-11}	8.16161×10^{-12}
$k = 0.001$	3.15165×10^{-13}	8.16430×10^{-13}	4.27353×10^{-13}
Numerical convergence	$--$	5.0718	5.2509

Table 2: Numerical errors at $t = 1$, $x = 1/2$ with some fifth-order ESERK schemes in Example 1. $s = 10, 40, 150$, and $k = 0.1, 0.05, 0.025$ were considered to study the stability and consistency of the numerical methods.

First of all, if k is fixed, we can check that all the errors have similar magnitudes for different s values. Logically, an exception is the case for ESERK with $s = 10$, when $k = 0.04$ (due to stability). With other s, x, and k values, other numbers for a numerical convergence rate were observed. However, they are frequently between 3.9 and 5.3, at least for small k values.

Test 2: A combustion 2D model

The second problem under consideration is a two-dimensional non-linear problem from combustion theory (see [20, 42] for example),

$$u_t = d\triangle u + \frac{R}{\alpha\delta}(1 + \alpha - u)\mathrm{e}^{\delta(1-1/u)}, \tag{12}$$

defined on the unit square for $t > 0$. The problem is subjected to the initial condition $u(x, y, 0) = 1$. For $t > 0$ we have the zero Neumann boundary condition at $x = 0$, $y = 0$ and the Dirichlet boundary condition $u = 1$ at $x = 1$, $y = 1$. 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 in the interval $[0, 1.48]$. We used second-order approximations for the Neumann conditions.

This problem models a reaction of a mixture of two chemicals with u representing the temperature of the mixture. For small times u is very smooth

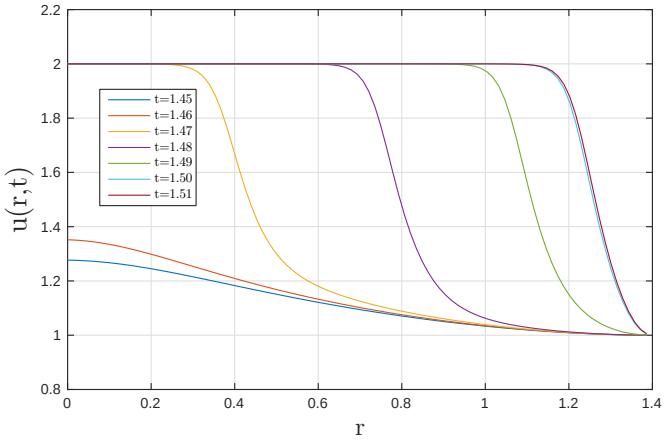

Figure 3: The traveling front solution of the combustion Test 1 along the diagonal $x = y$. At $t = 1.5$ and $t = 1.51$ the plots almost coincide due to steady-state arrival.

and large length steps can be used. Then ignition occurs, and u changes very fast from near unity to $1 + \alpha$. In this region small steps or higher-order schemes are mandatory. The reaction front reaches the boundary, near $t = 1.5$, then a steady state results. Figure 3 shows an accurate reference solution of the traveling front along the diagonal line $x = y$ at several output times $(t = 1.45, 1.46, 1.47, \ldots, 1.51)$ with $r = \sqrt{x^2 + y^2}$ on the horizontal axis.

First, in Table 3, numerical results at $t_{end} = 1.48$ (L_∞ errors) are shown to demonstrate that RKC has some difficulties to obtain an accurate solution. Additionally, readers can check that ESERK4 and ESERK5 are able to utilize much larger number of stages than ROCK4.

Tolerance	Method	max. err.	Time (s)	NFE	Steps	Max
10^{-7}	RKC	0.3910_{-1}	738.03	73184	839	474
	ROCK4	0.7637_{-6}	7441.56	830215	16959	54
	ESERK4	0.1060_{-3}	3646.07	314278	241	7000
	ESERK5	0.4943_{-4}	4075.21	456730	169	13500
10^{-9}	RKC	0.1816_{-2}	1645.25	161361	3893	224
	ROCK4	0.1617_{-6}	15155.73	1481171	51089	34
	ESERK4	0.6988_{-5}	7392.65	588785	788	4000
	ESERK5	0.2097_{-6}	5709.96	637079	388	7500

Table 3: Maximal absolute error, CPU times, number of function evaluations, steps, and maximal steps for the methods RKC, ROCK4, ESERK4, and ESERK5 using different values for the tolerances in the combustion 2D model.

As we can see in this Table 3, all methods achieve a poor accuracy compared to the prescribed tolerance (except ROCK4 when $tol = 10^{-7}$). This is due to the fact that the problem is very stiff near the endpoint. In this type of problems,

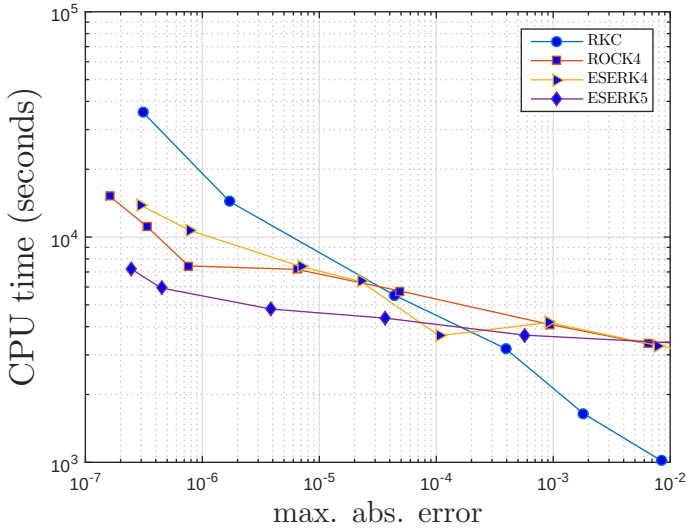

Figure 4: Maximal absolute error versus CPU times in seconds for the second example using RKC, ROCK4, ESERK4, and ESERK5.

we noticed that these integrators had similar difficulties, specially RKC. This is explained in the book by Hundsdorfer and Verwer [20]. It is related with the fact that $g'(u)$ (where $g(u)$ is the non-linear reaction term) is positive during a small interval close to (and including) the endpoint.

In Figure 4, we can also check how codes based on lower-order schemes are fast when large tolerances and errors are obtain. However, they become more expensive compared to higher-order methods when small errors are required. ESERK5 is the fastest in this numerical example when errors are in the range of $[10^{-5}, 10^{-8}]$.

Test 3: A 2D Brusselator problem with reaction

The third example considered in this paper is a two-dimensional Brusselator reaction-diffusion problem

$$\begin{aligned} \frac{\partial u}{\partial t} &= A + u^2 v - (B+1)u + \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right), \\ \frac{\partial v}{\partial t} &= Bu - u^2 v + \alpha \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right). \end{aligned} \tag{13}$$

We solve this problem for $0 \leq x \leq 1$, $0 \leq y \leq 1$, $0 \leq t \leq t_{end} = 1$, using $A = 1.3$, $B = 2 \times 10^6$, and $\alpha = 0.1$, with periodic boundary conditions $u(x+1, y, t) = u(x, y, t) = u(x, y+1, t)$ and the initial condition chosen as in [5]:

$$u(x, y, 0) = 22y(1-y)^{3/2}, \quad v(x, y, 0) = 27x(1-x)^{3/2}.$$

We discretized u, v in space with two $N \times N$ uniform meshes, where $N = 400$.

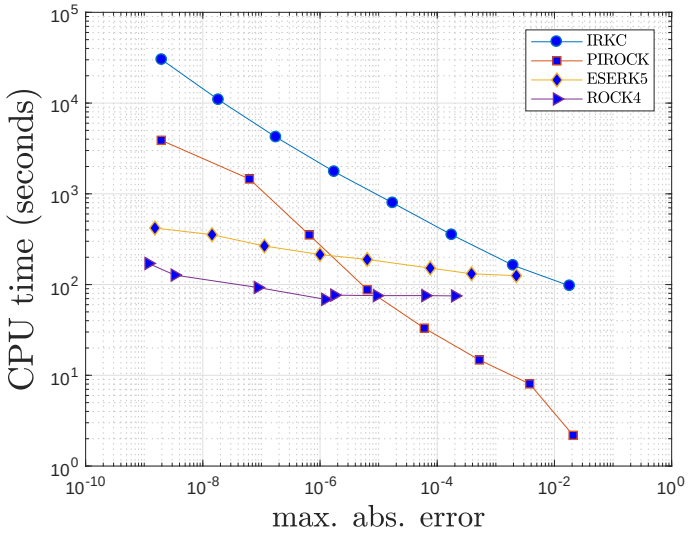

Figure 5: Maximal absolute error versus CPU times in seconds for the third example using IRKC, PIROCK and ESERK5.

Thus, $\rho \sim 2.4 \times 10^6$ and $\rho_D \sim 8\alpha N^2 = 1.28 \times 10^5$ (the spectral radius of the diffusion term).

In Figure 5, we are comparing the results obtained with ESERK5, ROCK4, IRKC, and PIROCK. The latter two are two excellent codes for these types of problems with diffusion plus very stiff reaction. As expected, these lower-order schemes are very fast when large tolerances are employed. However, they become more expensive compared to ESERK5 when small errors are required.

ROCK4 obtains excellent results in this numerical example, including when small errors are necessary. However, ROCK4 utilizes the maximum step in time for some of the tolerances. For longer t_{end} values, the solution gets smoother and ROCK4 employs the maximum length step in time. The advantage of ESERK5 is that this maximum length step in time is more than 40 times bigger than for ROCK4. Thus, when t_{end} is bigger (or B and N are larger) is able to obtain accurate solutions faster.

Although the reaction term makes the use of implicit-explicit algorithms very reasonable, ESERK schemes are able to approximate efficiently these types of problems if the reaction term is not extremely large. This is due to the large number of stages that they are able to employ without suffering from propagation of errors or internal stability.

6. Conclusions and future goals

In this article, we derive fifth-order stabilized explicit Runge-Kutta methods with large number of stages and good internal stability. These schemes are

very useful to solve non-linear parabolic PDEs in several dimensions, specially if small tolerances are required. We checked that they are also very efficient when the eigenvalues of the large system of ODEs are large in absolute value (over $O(10^5)$).

The technique employed in this paper, and also in [31] can be utilized to obtain large families of extrapolated SERK algorithms, and with different orders of convergence. Hence, it would be interesting to combine them in one code that allows changing the step size, the number of stages, and the order of convergence as it was done with extrapolated traditional explicit Runge–Kutta methods for ODEX. This might be studied in the future. Additionally, these methods can be combined with other implicit A-stable Runge-Kutta methods to solve other types of problems with advection and/or reaction as in [5, 13].

Additionally, we will consider a parallelization technique to reduce the CPU time of these codes. As with any other extrapolated explicit Runge-Kutta method, all the values $T_{j,1}$ can be computed independently of each other. In our case, the computational time employed by ESERK4 would decrease from a theoretical point of view by a factor of 2.5, and in the case of ESERK5, the CPU time would decrease almost by a factor of 3.

Acknowledgements

The authors would like to thank Assyr Abdulle, Britta Kleefeld, Lawrence Shampine, and Gilles Vilmart for providing us some codes, and their explanations about these algorithms.

[1] A. Abdulle. *Chebyshev methods based on orthogonal polynomials*. PhD thesis, University of Geneva, 2001.

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

[3] A. Abdulle and S. Cirilli. S-ROCK: Chebyshev methods for stiff stochastic differential equations. *SIAM J. Scientific Computing*, 30(2):997–1014, 2008.

[4] A. Abdulle and A. A. Medovikov. Second order Chebyshev methods based on orthogonal polynomials. *Numerische Mathematik*, 90(1):1–18, 2001.

[5] A. Abdulle and G. Vilmart. Pirock: A swiss-knife partitioned implicit–explicit orthogonal Runge–Kutta–Chebyshev integrator for stiff diffusio–advection–reaction problems with or without noise. *Journal of Computational Physics*, 242:869–888, 2013.

[6] R. C. Aiken. *Stiff Computation*. Oxford University Press, Inc., New York, NY, USA, 1985.

[7] V. Alexiades, G. Amiez, and P.-A. Gremaud. Super-time-stepping acceleration of explicit schemes for parabolic problems. *Communications in Numerical Methods in Engineering*, 12(1):31–42, 1996.

[8] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Appl. Numer. Math.*, 25(2-3):151–167, 1997.

[9] H. P. Bhatt and A. Q. M. Khaliq. The locally extrapolated exponential time differencing LOD scheme for multidimensional reaction–diffusion systems. *Journal of Computational and Applied Mathematics*, 285:256–278, 2015.

[10] J. C. Butcher and N. Rattenbury. ARK methods for stiff problems. *Applied Numerical Mathematics*, 53(2):165–181, 2005.

[11] J. Cash. The integration of stiff initial value problems in ODEs using modified extended backward differentiation formulae. *Computers & Mathematics with Applications*, 9(5):645–657, 1983.

[12] M. Cubillos-Moraga. *General-domain compressible Navier-Stokes solvers exhibiting quasi-unconditional stability and high-order accuracy in space and time*. PhD thesis, California Institute of Technology, 2015.

[13] M. Duarte, Z. Bonaventura, M. Massot, A. Bourdon, S. Descombes, and T. Dumont. A new numerical strategy with space-time adaptivity and error control for multi-scale streamer discharge simulations. *Journal of Computational Physics*, 231(3):1002–1019, 2012.

[14] M. Duarte, S. Descombes, C. Tenaud, S. Candel, and M. Massot. Timespace adaptive numerical methods for the simulation of combustion fronts. *Combustion and Flame*, 160(6):1083–1101, 2013.

[15] T. Dumont, M. Duarte, S. Descombes, M.-A. Dronne, M. Massot, and V. Louvet. Simulation of human ischemic stroke in realistic 3D geometry. *Communications in Nonlinear Science and Numerical Simulation*, 18(6):1539–1557, 2013.

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

[17] E. Hairer and G. Wanner. *Solving ordinary differential equations. II: Stiff and differential-algebraic problems*. Springer, Berlin, 1996.

[18] A. C. Hindmarsh. LSODE and LSODI, two new initial value ordinary differential equation solvers. *ACM SIGNUM Newsletter*, 15(4):10–11, 1980.

[19] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comp.*, 19(5):1552–1574, 1998.

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

[21] L. Ixaru, G. V. Berghe, and H. D. Meyer. Frequency evaluation in exponential fitting multistep algorithms for ODEs. *Journal of Computational and Applied Mathematics*, 140(1-2):423–434, 2002.

[22] 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.

[23] D. I. Ketcheson and U. bin Waheed. A comparison of high order explicit Runge-Kutta, extrapolation, and deferred correction methods in serial and parallel. *CAMCoS*, 9(2):175–200, 2014.

[24] A. Q. M. Khaliq, J. Martín-Vaquero, B. Wade, and M. Yousuf. Smoothing schemes for reaction-diffusion systems with nonsmooth data. *Journal of Computational and Applied Mathematics*, 223(1):374–386, 2009.

[25] B. Kleefeld and J. Martín-Vaquero. Serk2v2: A new second-order stabilized explicit Runge-Kutta method for stiff problems. *Numerical Methods for Partial Differential Equations*, 29(1):170–185, 2013.

[26] B. Kleefeld and J. Martín-Vaquero. Serk2v3: solving mildly stiff nonlinear partial differential equations. *Journal of Computationald & Applied Mathematics*, 299:194–206, 2016.

[27] V. I. Lebedev. A new method for determining the roots of polynomials of least deviation on a segment with weight and subject to additional conditions. part II. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 8(5):397–426, 1993.

[28] V. I. Lebedev and S. A. Finogenov. Solution of the parameter ordering problem in chebyshev iterative methods. *USSR Computational Mathematics and Mathematical Physics*, 13(1):21–41, 1974.

[29] J. Martín-Vaquero and B. Janssen. Second-order stabilized explicit Runge-Kutta methods for stiff problems. *Computer Physics Communications*, 180(10):1802–1810, 2009.

[30] J. Martín-Vaquero, A. Q. M. Khaliq, and B. Kleefeld. Stabilized explicit Runge-Kutta methods for multi-asset American options. *Computers & Mathematics with Applications*, 67(6):1293–1308, 2014.

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

[32] J. Martín-Vaquero and J. Vigo-Aguiar. Exponential fitting BDF algorithms: Explicit and implicit 0-stable methods. *Journal of Computational and Applied Mathematics*, 192(1):100–113, 2006.

[33] A. A. Medovikov. High order explicit methods for parabolic equations. *BIT Numerical Mathematics*, 38(2):372–390, 1998.

[34] L. Noels, L. Stainier, and J.-P. Ponthot. Combined implicit/explicit time-integration algorithms for the numerical simulation of sheet metal forming. *Journal of Computational and Applied Mathematics*, 168(1-2):331–339, 2004.

[35] S. O'Sullivan. A class of high-order Runge–Kutta–Chebyshev stability polynomials. *Journal of Computational Physics*, 300:665–678, 2015.

[36] S. O'Sullivan. Runge-Kutta-Gegenbauer methods for advection-diffusion problems. *ArXiv e-prints*, Dec. 2017.

[37] S. O'Sullivan and T. P. Downes. A three-dimensional numerical method for modelling weakly ionized plasmas. *Monthly Notices of the Royal Astronomical Society*, 376(4):1648–1658, 2007.

[38] S. O'Sullivan and C. O'Sullivan. On the acceleration of explicit finite difference methods for option pricing. *Quantitative Finance*, 11(8):1177–1191, 2011.

[39] L. Skvortsov. Explicit stabilized Runge-Kutta methods. *Computational Mathematics and Mathematical Physics*, 51(7):1153–1166, 2011.

[40] 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.

[41] M. Torrilhon and R. Jeltsch. Essentially optimal explicit Runge-Kutta methods with application to hyperbolic-parabolic equations. *Numerische Mathematik*, 106(2):303–334, 2007.

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

[43] Z. Zlatev. *Computer Treatment of Large Air Pollution Models*. Environmental Science and Technology Library. Springer Netherlands, 2012.

Appendix:

In this section we provide some of the coefficients developed for this work. Unfortunately all the new schemes required over 15000 new coefficients, this is time-consuming, but we are not able to provide all of them in this manuscript. However, they can be found in https://github.com/kleefeld80/ESERK5, lines 37 to 15638.

For $s = 1$ the final b_i coefficients are:

$$b_0 = 0.51, b_1 = 0.49.$$

For $s = 2$ the final b_i coefficients are:

$$b_0 = -0.2128171597633136, b_1 = 0.9637562130177514,$$

$$b_2 = 0.2490609467455621.$$

For $s = 3$, b_i coefficients are:

$$b_0 = 0.1712922718556347, b_1 = -0.1423943632649187,$$

$$b_2 = 0.3031160937815012, b_3 = 0.6679859976277827.$$

For $s = 4$, final coefficients are:

$$b_0 = 0.1412541319644020, b_1 = -0.8685875774123184, b_2 = 0.4776631958662505,$$

$$b_3 = 1.0664757342353369 2, b_4 = 0.18319451534632894.$$

For $s = 5$, b_i coefficients are:

$$b_0 = -0.07055272331742087, b_1 = -0.28314320544924802,$$

$$b_2 = 0.19785090029766025, b_3 = -0.42177903459881749,$$

$$b_4 = 0.34798179035774081, b_5 = 1.22964227271008532.$$

For $s = 6$, b_i coefficients are:

$$b_0 = -0.0233121086266971, b_1 = -0.1335872416064247, b_2 = -0.8078161877275098,$$

$$b_3 = 0.0354768957832420, b_4 = 0.1496077877880309,$$

$$b_5 = 0.5749883471254331, b_6 = 1.2046425072639256.$$

For $s = 7$, b_i coefficients are:

$$b_0 = -0.0618325593695405, b_1 = 0.2458501093889481,$$

$$b_2 = -0.2227872829351750, b_3 = -1.1522803607019805,$$

$$b_4 = 0.2724834533245227, b_5 = -0.9466350971213604,$$

$$b_6 = 0.4910735130318972, b_7 = 2.3741282243826884.$$

For $s = 8$, final b_i coefficients are:

$$b_0 = 0.2676299331370952, b_1 = -0.0061415869134074, b_2 = -0.0754137605301323,$$

$$b_3 = -0.4137326085817093, b_4 = -2.2197559533564896,$$

$$b_5 = 0.0418316007174432, b_6 = 0.1982519692491985,$$

$$b_7 = 0.8594290797003330, b_8 = 2.3479013265776686.$$

For $s = 9$, final coefficients are:

$$b_0 = 0.0433423349672831, b_1 = 0.2831471867437382,$$

$$b_2 = -0.1969117505913101, b_3 = 1.0143522195178293,$$

$$b_4 = -0.5445644897846778, b_5 = -3.4422730603162499,$$

$$b_6 = 0.4161605174583901, b_7 = -1.9943115249036625,$$

$$b_8 = 0.7661598440442524, b_9 = 4.6548987228644070.$$

For $s = 10$, b_i coefficients are:

$$b_0 = 0.0118598151160970, b_1 = 0.0831381079859492, b_2 = 1.3652430804804162,$$

$$b_3 = -0.0182747403058947, b_4 = -0.1762850131883753,$$

$$b_5 = -1.0169885262162042, b_6 = -5.5937613871887220,$$

$$b_7 = 0.0531212686641022, b_8 = 0.2837514939268146,$$

$$b_9 = 1.3863138835464458, b_{10} = 4.6218820171793711.$$

For $s = 11$, b_i coefficients are:

$$b_0 = 0.0396998143921759, b_1 = -0.2612331415070279,$$

$$b_2 = 0.2237774628808507, b_3 = 1.6966675622686337,$$

$$b_4 = -0.4913194960095769, b_5 = 3.1080707614085650,$$

$$b_6 = -1.2286488173060305, b_7 = -9.1168936653993763,$$

$$b_8 = 0.6778510530177390, b_9 = -4.1063172037290536,$$

$$b_{10} = 1.2693754026307855, b_{11} = 9.1889702673523152.$$

For $s = 12$, final coefficients are:

$$b_0 = -0.2721732816746095, b_1 = 0.0023242317945640, b_2 = 0.0583515040445118,$$

$$b_3 = 0.4288342559984806, b_4 = 4.9590570916533354,$$

$$b_5 = -0.0397726675163696, b_6 = -0.3657178767124540,$$

$$b_7 = -2.3156825268742213, b_8 = -13.4392531313636734,$$

$$b_9 = 0.0707761623683441, b_{10} = 0.4228299401098627,$$

$$b_{11} = 2.3479638334679581, b_{12} = 9.1424624647042709.$$

For $s = 13$, b_i coefficients are:

$$b_0 = -0.0323291168689784, b_1 = -0.2804146299695864,$$

$$b_2 = 0.2072403341391079, b_3 = -1.5857902313554160,$$

$$b_4 = 0.7892241538284155, b_5 = 6.7610317353997789,$$

$$b_6 = -1.1280644757467384, b_7 = 8.4128930401780590,$$

$$b_8 = -2.6810327635279448, b_9 = -22.6441139374225888,$$

$$b_{10} = 1.1533853320305336, b_{11} = -8.3664929427388960,$$

$$b_{12} = 2.1899914542080842, b_{13} = 18.2044720478461697.$$

For $s = 14$, b_i coefficients are:

$$b_0 = -0.0066746193320942, b_1 = -0.0627341688276580, b_2 = -1.9186006564469559,$$

$$b_3 = 0.0108195027269256, b_4 = 0.1862028740070028,$$

$$b_5 = 1.5168549709345810, b_6 = 15.4846722522406338,$$

$$b_7 = -0.0762943554961964, b_8 = -0.7099428072976171,$$

$$b_9 = -5.0978612128849602, b_{10} = -31.3136629305042665,$$

$$b_{11} = 0.0966892528144702, b_{12} = 0.6416072939932105,$$

$$b_{13} = 4.1151576626715777, b_{14} = 18.1337669414013465.$$

For $s = 15$, b_i coefficients are:

$$b_0 = -0.0303749487922495, b_1 = 0.2676948265526194,$$

$$b_2 = -0.2342348247235180, b_3 = -2.2369116983358381,$$

$$b_4 = 0.7382581599821140, b_5 = -6.4025911985291914,$$

$$b_6 = 2.3691160375685935, b_7 = 22.4404831319297990,$$

$$b_8 = -2.4948213905392395, b_9 = 21.2665447333875015,$$

$$b_{10} = -5.7573754412653146, b_{11} = -54.0325218520891823,$$

$$b_{12} = 2.0255889138384524, b_{13} = -16.9513244714478901,$$

$$b_{14} = 3.8909179921272460, b_{15} = 36.1415520303361006.$$