Networks and Heterogeneous Media

NHM, 18(3): 1059–1082.
DOI: 10.3934/nhm.2023046
Received: 04 January 2023
Revised: 26 February 2023
Accepted: 02 March 2023
Published: 29 March 2023

http://www.aimspress.com/journal/NHM

Research article

Adaptive step size controllers based on Runge-Kutta and linear-neighbor methods for solving the non-stationary heat conduction equation

Mahmoud Saleh[1], Endre Kovács[1,*] and Nagaraja Kallur[2]

[1] University of Miskolc, Institute of Physics and Electrical Engineering, Hungary
[2] Department of Mathematics, Amrita School of Engineering, Bengaluru, Amrita Vishwa Vidyapeetham, India

* **Correspondence:** Email: kendre01@gmail.com; Tel: +36-46-565-156.

Abstract: We systematically test families of explicit adaptive step size controllers for solving the diffusion or heat equation. After discretizing the space variables as in the conventional method of lines, we are left with a system of ordinary differential equations (ODEs). Different methods for estimating the local error and techniques for changing the step size when solving a system of ODEs were suggested previously by researchers. In this paper, those local error estimators and techniques are used to generate different types of adaptive step size controllers. Those controllers are applied to a system of ODEs resulting from discretizing diffusion equations. The performances of the controllers were compared in the cases of three different experiments. The first and the second system are heat conduction in homogeneous and inhomogeneous media, while the third one contains a moving heat source that can correspond to a welding process.

Keywords: diffusion equation; explicit adaptive step controller; moving heat source; Runge-Kutta methods; linear-neighbor method

1. Introduction

1.1. The studied problems

We consider a linear second order partial differential equation (PDE), the so-called heat equation [1]:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + q \tag{1.1}$$

Or more generally,

$$c\rho \frac{\partial u}{\partial t} = \nabla \left(k \nabla u \right) + c\rho q \tag{1.2}$$

where $u = u(t,x)$ is the temperature, $\alpha = k/(c\rho) > 0$ is the thermal diffusivity, and q, k, c and ρ are the intensity of the heat source, heat conductivity, specific heat and (mass) density, respectively. To solve Eq (1.1) numerically, we construct an equidistant grid of N nodes. For simplicity let us consider a one-dimensional equation in a homogeneous medium. Now, if we make a second order central difference approximation for the space derivative, then we get a system of ordinary differential equations in the following form:

$$\frac{du_i}{dt} = \alpha \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} + q_i, \; i = 1, \ldots, N$$

which can be written in matrix form as

$$\frac{d\vec{u}}{dt} = M\vec{u} + \vec{q} \, . \tag{1.3}$$

$M_{N,N}$ is a square matrix with $m_{ii} = -\frac{2\alpha}{\Delta x^2}$, $m_{i,i+1} = m_{i,i-1} = \frac{\alpha}{\Delta x^2} (1 < i < N)$. If the material parameters, such as c, are not spatially homogeneous, and/or the mesh is not equidistant, we use a resistance-capacitance model, which can be derived from PDE (1.2) using the central difference formula. For more details, see [2,3], where we have done it in the case of arbitrary space dimensions. In the general case, the system of ODEs can be written as follows:

$$\frac{du_i}{dt} = \sum_{j \neq i} \frac{u_j - u_i}{R_{i,j} C_i} + q_i \tag{1.4}$$

where C_i, $R_{i,j}$ are the heat capacity and the thermal resistance of the cell labeled by i. Eq (1.4) can be written into the same matrix form as in Eq (1.3), and since the capacities and resistances are nonnegative quantities, the eigenvalues $\lambda_i \; i = 1, \ldots, N$ of the matrix M remain non-positive. The stiffness ratio now can be defined as $max\{|\lambda_i|\} / min\{|\lambda_i|\}$, while the CFL (Courant–Friedrichs–Lewy) limit for the classical first order explicit method can be given as $2/\max\{|\lambda_i|\}$, and this limit is just slightly larger for the fourth order Runge-Kutta (RK) methods. It is well known that explicit Runge-

Kutta methods are only conditionally stable, which means that if the time step size is larger than the CFL limit, the solution is expected to start to oscillate with exponentially increasing amplitude.

1.2. Brief literature review and the scope of the paper

Many explicit numerical methods are available for solving a parabolic PDE, such as Eq (1.1) or Eq (1.2). A typical approach is the so-called method of lines (MOL), that is, to spatially discretize them and then solve the obtained system of ODEs by well-established ODE solvers. In fact, the three most common schemes, i.e., the standard explicit, implicit and Crank-Nicolson finite difference schemes, can be perceived as members of the MOL family, since after spatial discretization, they apply the explicit and implicit Euler time-discretization and the implicit trapezoidal rule, which are actually two first order and one second order RK methods, respectively. Although the family of the RK methods has been well known for a long time, it is still challenging to outperform them. For example, Savovic et al. [4] tested the so-called UPFD (unconditionally positive finite difference) scheme, constructed in the last decade, against the simplest standard explicit finite difference scheme. However, they obtained that the classical scheme is more accurate than the UPFD due to the extra truncation error terms of the UPFD scheme. Our current study will be restricted to two fundamentally different types of explicit methods, the multi-stage RK methods and the recently published LNe3 method [2].

In general, the explicit numerical methods for solving a system of ordinary differential equations use fixed time steps. This kind of approach can perform poorly if the solution changes rapidly in some parts of the integration interval and slowly in other ones. Using a small constant time-step where the solution changes rapidly can help to circumvent the problem of poor performance, but this small constant time-step may cause unnecessary computational cost where the solution changes slowly. Using adaptive methods based on automatic time-step selection can be the remedy of the expensive numerical computations [5]. Adapting the time-step size during the integration process is not just a matter of improving the performance of the integrator; it makes the solution of difficult problems practical [6]. There are at least three crucial factors when it comes to designing adaptive step-size integrators: the method of calculating the solution at the end of the actual time step, the method of estimating the local error in each time step and the approach for changing the time step [7].

Estimating the local error in Runge-Kutta methods when applied to ordinary differential equations was studied intensively in the literature of numerical analysis [5–14]. The methods of estimating the local error can be classified into two types: the methods which use the information from only a single step and those which use the information from successive steps. The methods of the second type are out of the scope of this paper. The most well-known method of the first type for estimating the local error is to calculate the dependent variable u first by using a full time step h and to recalculate it using two halved time steps $h/2$. The difference between the two values of u represents the local error LE. Another common method is the pseudo-iterative formula, which uses an RK formula of order p and then an RK formula of order $p + 1$, which uses the already calculated quantities of the lower order RK formula to save time, which is why these algorithms are called embedded methods [13]. In his early work [8], Merson derived a formula that gives a plausible estimation for the local error which is valid only if the ordinary differential equation is linear. Later, Scraton [14] suggested a new formula for estimating the local error but without any restrictions on its validity. The particular schemes of most

interest are given in Eqs 7–9 of his paper. However, the formula suggested by Scraton can be applied only to a single differential equation, not to a system of ODEs [12]. To overcome this shortcoming, England [12] introduced a process that can be valid even when it is applied to a general system of ODEs. Also, Shampine [7] proposed a new formula in his work and compared the performances of different error estimators.

The approach for changing the step size in the case of ODEs can be done using the elementary controller

$$h_{new} = s\left(\frac{TOL}{LE}\right)^{\frac{1}{p}} h_{present}, \tag{1.5}$$

where $s < 1$ is the safety factor, TOL is the tolerance specified by the user, h is the time step, LE is the estimated local error, and p is the order of the method. The elementary controller changes the step size based on the current estimation of the local error. This elementary controller generally shows good performance, but there are some exceptions. For instance, the time step size can be limited by the stability properties of the used method, which in turn causes an oscillation in the step size sequences. More details about the shortcomings of the elementary controller can be found in [15]. Based on control theory, Gustafsson [16,17] introduced the so-called PI controller to overcome the problem of oscillating step size. Its adaptivity algorithm is

$$h_{new} = \left(\frac{TOL}{LE_n}\right)^{K_I}\left(\frac{LE_{n-1}}{LE_n}\right)^{K_P} h_{present}, \tag{1.6}$$

where K_I and K_P are constants, LE_n is the estimated local error at the current step size, LE_{n-1} is the estimated local error at the previous step size, and h is the step size. Unlike the elementary controller, the PI controller changes the step size based on the past history of the local error estimation. Later, Soderlind [18–20] investigated this type of controller. He developed new strategies for adaptive step size based on digital control theory.

In the previously mentioned references, the authors tested the methods for estimating the local error and the approaches for changing the step size only in the case of small systems of ODEs. Those algorithms might be efficient when they are applied to a single ordinary differential equation or a system of ODEs including a limited number of equations. The objective of this paper is to design and extensively test adaptive step size controllers based on the previously mentioned studies and then apply those algorithms to an equation system (1.3), where the size of matrix M is big.

1.3. The outline of the paper

In Section 2, the I and PI types of step size controllers are introduced in more detail. In Section 3, we show the steps of designing adaptive controllers based on the Runge-Kutta and LNe3 methods. Then, in Section 4, we present three numerical experiments for 2D systems. The first case study is a spatially uniform system which is not stiff, and then the second one is a very stiff system. Finally, the third one includes a moving heat source, which can be considered as simulating a welding process.

2. The I and PI step-size controllers

Suppose that we applied an explicit numerical method of order P, with step size $h_{present}$, to equation system (1.3) in order to get an approximated solution u_i^{n+1}. Also, assume that we used some method for estimating the local error, and the local error estimation, based on that method, is denoted by LE_i. The norm of the error estimation is [21]

$$err^{n+1} = \max_{1\le i\le N}\left\{\frac{|LE_i|}{AbsTol_i + \left|u_i^{n+1}\right| \mathrm{Re}lTol}\right\}, \tag{2.1}$$

where *AbsTol* and *RelTol* are the relative and absolute tolerances that can be defined by the user.

We note that in Eq (17) in [5] and Eq (4.10) in [22], the authors used a different formula for calculating the norm of the error estimation. In their formula, they did not only consider the value of u_i^{n+1}, but they also considered the value of u_i^n. Based on our numerical experiments, we do believe that including the value of u_i^n in the calculation does not have a significant effect, and it only causes extra cost.

Now, after calculating err^{n+1} we can change the step size using the following formula [5]:

$$h_{new} = \min\left(f_{\max}, \max\left(f_{\min}, f_s \beta^{n+1}\right)\right) h_{present}, \tag{2.2}$$

where β^{n+1} is a function of err^{n+1}, and it depends on the type of the step size controller. That function will be defined for each type of controller individually, as we will see later. The nonnegative number f_s is a safety factor, and it is used to increase the probability of accepting the step size in the next iteration. The factors f_{min} and f_{max} are used to prevent the step size from decreasing or increasing too rapidly. In our code, we set the following values for the factors: f_s = 0.9, f_{min} =0.1, f_{max} = 5. If $err^{n+1} \le 1$, the step size is accepted, the solution is advanced with u_i^{n+1}, and the step size will be modified by Eq (2.2). If $err^{n+1} > 1$, the step size and the solution u_i^{n+1} are rejected, and the calculations are repeated with a new time step calculated by Eq (2.2). It is worth mentioning here that some researchers suggested setting a new, smaller value for f_{max} in computing the approximate solution directly after a step rejection [23]. The main goal of this procedure is to prevent an infinite loop that can occur. In our numerical experiment, we did not encounter such infinite loops. More information about the rejected time-step size will be given in Subsection 4.1.

In the elementary controller I (asymptotic) the value of the function β^{n+1} depends on the currently estimated error as follows:

$$\beta^{n+1} = \left(err^{n+1}\right)^{\frac{-1}{p}}. \tag{2.3}$$

In the PI controller the value of the function β^{n+1} depends on the current and previous estimated errors as follows:

$$\beta^{n+1} = \left(err^{n+1}\right)^{\frac{-k_1}{p}} \left(err^{n}\right)^{\frac{k_2}{p}}, \tag{2.4}$$

Here, the values $k_1 = 0.8$, $k_2 = 0.31$ are taken from [5]. For the first step we considered that $err^n = 1$. To be systematic, we run all our code for adaptive controller algorithms considering the following decreasing series for the tolerance: $AbsTol = RelTol = 2^{-1}, 2^{-2}, \ldots$.

3. Description of the schemes

For the sake of simplicity, all the schemes resulting from Runge-Kutta methods and adaptive Runge-Kutta methods are described when they are applied to a single ordinary differential equation. Nevertheless, these schemes can be straightforwardly expanded to solve the system in Eq (1.3).

For the initial value problem (IVP) of the form

$$\left.\begin{aligned} \frac{du}{dt} &= f\left(t,u\right) \\ u\left(t^0\right) &= u^0 \end{aligned}\right\},$$

the general s-stage Runge-Kutta method can be written as follows [24]:

$$\left.\begin{aligned} u^{n+1} &= u^n + h\sum_{i=1}^{s} b_i k_i \\ k_i &= f\left(t^n + c_i h, u^n + h\sum_{j=1}^{s} a_{ij} k_j\right), i = 1:s. \end{aligned}\right\}, \tag{3.1}$$

where a_{ij}, b_i and c_i are constants and can be defined using the Butcher tableau.

3.1. Group A: Dormand-Prince fifth-order Runge-Kutta method

In the Dormand-Prince method, the quantities k_i are evaluated as follows [15]:

$$\left.\begin{aligned}
k_1 &= f\left(t^n, u^n\right)\\
k_2 &= f\left(t^n + \frac{1}{5}h, u^n + \frac{1}{5}hk_1\right)\\
k_3 &= f\left(t^n + \frac{3}{10}h, u^n + \frac{3}{40}hk_1 + \frac{9}{40}hk_2\right)\\
k_4 &= f\left(t^n + \frac{4}{5}h, u^n + \frac{44}{45}hk_1 - \frac{56}{15}hk_2 + \frac{32}{9}hk_3\right)\\
k_5 &= f\left(t^n + \frac{8}{9}h, u^n + \frac{19372}{6561}hk_1 - \frac{25360}{2187}hk_2 + \frac{64448}{6561}hk_3 - \frac{212}{729}hk_4\right)\\
k_6 &= f\left(t^n + h, u^n + \frac{9017}{3168}hk_1 - \frac{355}{33}hk_2 + \frac{46732}{5247}hk_3 + \frac{49}{176}hk_4 - \frac{5103}{18656}hk_5\right)
\end{aligned}\right\}. \tag{3.2}$$

The fifth-order Runge-Kutta formula is

$$u^{n+1} = u^n + h\left(\frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6\right). \tag{3.3}$$

Estimating the local error requires another formula. To do that, Dormand considered an extra evaluation:

$$k_7 = f\left(t^n + h, \frac{35}{384}hk_1 + \frac{500}{1113}hk_3 + \frac{125}{192}hk_4 - \frac{2187}{6784}hk_5 + \frac{11}{84}hk_6\right). \tag{3.4}$$

The embedded formula is

$$\hat{u}^{n+1} = u^n + h\left(\frac{5179}{57600}k_1 + \frac{7571}{16695}k_3 + \frac{393}{640}k_4 - \frac{92697}{339200}k_5 + \frac{187}{2100}k_6 + \frac{1}{40}k_7\right). \tag{3.5}$$

The coefficients a_{7j} in Eq (3.4) are chosen to be the same as the coefficients b_i in Eq (3.3). This means that Eq (3.4) is equivalent to

$$k_7 = f\left(t^n + h, u^{n+1}\right).$$

Now, for the next step (when $err^{n+1} \le 1$), we set $k_1 = k_7$. This trick is called FSAL (first-same-as-last). This means that, in the case of acceptance, the evaluation of the function k_7 can be reused in the following step as k_1, which in turn reduces the cost of computations. The local error estimation can be calculated:

$$LE = \left|\hat{u}^{n+1} - u^{n+1}\right|. \tag{3.6}$$

Our notations hide the fact that the functions LE and k are vectors if the method is applied to a system of ODEs. Since the method is applied to a single differential equation, as we mentioned previously, the i index has been dropped from all the formulas of those functions. Also, for the functions of the subsequent sections, except Subsection 3.4, the i index has been dropped as well.

Substituting Eqs (3.5)–(3.7) and considering Eq (2.2) and Eq (2.3) result in the adaptive step size scheme of the Dormand-Prince method with I controller type, which will be denoted as "DPRK5(4) #I". Considering Eq (2.4) instead of Eq (2.3) leads us to the adaptive step size scheme of the Dormand-Prince method but with PI controller type, which will be denoted as "DPRK5(4) #PI". For the sake of comparison, we will also test the scheme of Eq (3.3), which is a fifth-order Runge-Kutta method with a fixed step size. This method will be denoted as "non-adaptive DPRK5(4)".

Based on Eq (3.2), we can design adaptive controllers which use the doubling step size technique. First, we take a single step of size h, and we use the four stages of Eq (3.2) in order to calculate the solution u^{n+1} using Eq (3.3). Second, we take two steps of size $\frac{\mathrm{h}}{2}$ to recalculate the solution, denoted again by $\hat{u}^{n+1}$, using Eqs (3.2) and (3.3) two times. The local error can be simply estimated as in Eq (3.6) and then substituted into Eq (2.1) to obtain

$$err^{n+1} = \max\left\{\frac{|LE|}{AbsTol + \left|u^{n+1}\right| \mathrm{Re}lTol}\right\}. \tag{3.7}$$

Note that instead of $\hat{u}^{n+1}$, now there is u^{n+1} in the denominator. If the error norm is tolerable, the step size is accepted, and there are three possibilities to advance the solution. The first possibility is to accept the solution u^{n+1} resulting from taking a single step. With that possibility, we can design two controllers based on Eqs (2.2)–(2.4). Those adaptive controllers will be denoted as "RKduplicate 1#I" and "Rkduplicate 1 # PI". The second possibility is to accept the more accurate solution $\hat{u}^{n+1}$, and we will be left again with "RKduplicate 2 # I" and "RKduplicate 2 # PI". The third one is Richardson extrapolation (see Eq (4.5) from [22]), which combines the solutions u^{n+1} and $\hat{u}^{n+1}$ to produce another, more accurate solution as follows:

$$\tilde{u}^{n+1} = \hat{u}^{n+1} + \left(\frac{\hat{u}^{n+1} - u^{n+1}}{2^p - 1}\right), \tag{3.8}$$

where p is the order of the method, which is four in this scheme. Based on the last formula, we can design two adaptive controllers, and they will be denoted as "RKduplicate 3 # I" and "RKduplicate 3 # PI".

3.2. Group B: Scraton's fourth-order Runge-Kutta method

In his work [14], Scraton introduced a Runge-Kutta method with five stages,

$$\left.\begin{aligned}
k_1 &= f\left(t^n, u^n\right)\\
k_2 &= f\left(t^n + \frac{2}{9}h, u^n + \frac{2}{9}hk_1\right)\\
k_3 &= f\left(t^n + \frac{1}{3}h, u^n + \frac{1}{12}hk_1 + \frac{1}{4}hk_2\right)\\
k_4 &= f\left(t^n + \frac{3}{4}h, u^n + \frac{3}{128}\left(23hk_1 - 81hk_2 + 90hk_3\right)\right)\\
k_5 &= f\left(t^n + \frac{9}{10}h, u^n + \frac{9}{10000}\left(-345hk_1 + 2025hk_2 - 1224hk_3 + 544hk_4\right)\right)
\end{aligned}\right\}, \tag{3.9}$$

and the fourth-order scheme is

$$u^{n+1} = u^n + h\left(\frac{17}{162}k_1 + \frac{81}{170}k_3 + \frac{32}{135}k_4 + \frac{250}{1377}k_5\right). \tag{3.10}$$

To estimate the local error, Scraton evaluated the following functions:

$$\begin{aligned}
q &= -\frac{1}{18}k_1 + \frac{27}{170}k_3 - \frac{4}{15}k_4 + \frac{25}{153}k_5\\
r &= \frac{19}{24}k_1 - \frac{27}{8}k_2 + \frac{57}{20}k_3 - \frac{4}{15}k_4\\
s &= k_3 - k_1
\end{aligned}$$

Then, the local error estimation is given by the following nonlinear formula:

$$LE = -\frac{qr}{s}. \tag{3.11}$$

Scraton stated that subtracting the local error calculated in Eq (3.11) from Eq (3.10) will increase the order of the scheme to five:

$$u^{n+1} = u^n + h\left(\frac{17}{162}k_1 + \frac{81}{170}k_3 + \frac{32}{135}k_4 + \frac{250}{1377}k_5\right) + \frac{qr}{s}. \tag{3.12}$$

For easy recognition, we refer to scheme (3.10) as "non-adaptive RKSc 1" and Eq (3.12) as "non-adaptive RKSc 2". To design an adaptive step size controller based on Scraton's error estimation, we substitute Eqs (3.10) and (3.11) into Eq (2.1), and then we use Eqs (2.2) and (2.3). This is an adaptive-Scraton scheme with an I controller type. We have two possibilities to advance the solution when the step size is accepted: First, use Eq (3.10), and this type will be referred to as "RKSc 1 # I". Second, use Eq (3.12), and this type will be referred to as "RKSc 2 # I". If we repeat the previous steps but use Eq (2.4) instead of Eq (2.3), we will get another two types of adaptive step size controllers, which are "RKSc 1 # PI" and "RKSc 2 # PI".

3.3. Group C: England fourth-order Runge-Kutta method

England used the four-stage Runge-Kutta method in the first step [7,12]:

$$\begin{aligned} k_1 &= f\left(t^n, u^n\right) \\ k_2 &= f\left(t^n + \frac{1}{2}h, u^n + \frac{1}{2}hk_1\right) \\ k_3 &= f\left(t^n + \frac{1}{2}h, u^n + \frac{1}{4}\left(hk_1 + hk_2\right)\right) \\ k_4 &= f\left(t^n + h, u^n - hk_2 + 2hk_3\right) \end{aligned} \tag{3.13}$$

The fourth-order formula is

$$u^{n+1} = u^n + \frac{h}{6}\left(k_1 + 4k_3 + k_4\right) \tag{3.14}$$

Unlike in the case of other methods, England started the second time step before estimating the local error, and he used the same stage formulas as in the first step:

$$\begin{aligned} k_5 &= f\left(t^n + h, u^{n+1}\right) \\ k_6 &= f\left(t^n + \frac{3}{2}h, u^{n+1} + \frac{1}{2}hk_5\right) \\ k_7 &= f\left(t^n + \frac{3}{2}h, u^{n+1} + \frac{1}{4}\left(hk_5 + hk_6\right)\right) \end{aligned}$$

Before completing the second step, an extra evaluation is made which enables us to estimate the local error accumulated in the two steps:

$$k_{extra} = f\left(t^n + 2h, u^n + \frac{h}{6}\left(-k_1 - 96k_2 + 92k_3 - 121k_4 + 144k_5 + 6k_6 - 12k_7\right)\right)$$

The local error estimation according to England is

$$LE_{En} = \frac{h}{90}\left(-k_1 + 4k_3 + 17k_4 - 23k_5 + 4k_7 - k_{extra}\right). \tag{3.15}$$

Now, we substitute Eqs (3.14) and (3.15) into Eq (2.1):

$$err^{n+2} = \max\left\{\frac{\left|LE_{En}\right|}{AbsTol + \left|u^{n+1}\right| \mathrm{Re}lTol}\right\}. \tag{3.16}$$

If $err^{n+2} \le 1$, the evaluation of the last function in the second step can now proceed,

$$k_8 = f\left(t^n + 2h, u^{n+1} - hk_6 + 2hk_7\right),$$

and we accept the following numerical solution:

$$u^{n+2} = u^{n+1} + \frac{h}{6}\left(k_5 + 4k_7 + k_8\right) \tag{3.17}$$

If $err^{n+2} \leq 1$, we reject the step size without evaluating the function k_8 and repeat the first step with a new step size. In this case, we lose seven function evaluations. When the time step is acceptable, nine function evaluations will be performed if we consider the calculation of k_{extra} as well. In other words, we make only $4\frac{1}{2}$ function evaluations for each step, while it requires five function evaluations per step if we use the logic of the classical adaptive Runge-Kutta such as the Fehlberg method. This means that we saved 1/2 of a function evaluation for each step. One might argue that saving only a half function evaluation per step cannot compensate for the expensive cost of the probability of losing seven function evaluations when the step size is rejected. According to the experience, for a well-designed adaptive controller, the probability of a rejected step size is low, and the majority of the steps are accepted. It is worth it here to recall that there are more effective tricks that enable us to make only four function evaluations per step: for instance, the so called FSAL trick previously described and the local extrapolation technique [15, p. 915].

Nevertheless, in case of either a rejected or an accepted step, we should repeat or proceed with the calculations with the new step size. If we use Eq (3.16) along with Eqs (2.2) and (2.3), an adaptive controller of type I is applied, and it will be denoted as "RKEn #I". If we use Eq (3.16) along with Eq (2.4), a new adaptive controller is applied but of type "PI", and it will be denoted as "RKEn #PI". The simple 4th-order RK scheme using only Eq (3.14) will be referred to as "non-adaptive RKEn".

Shampine [7] used the same function evaluations that England used but with a new local error estimator of the form

$$LE_{Sh} = \frac{h}{180}\left(k_1 - 4k_3 - 17k_4 + 23k_5 - 4k_7 + k_{extra}\right) = -\frac{1}{2}LE_{En}\,. \tag{3.18}$$

Again, based on Shampine's formula for estimating the local error and using Eqs (2.1)–(2.4), we can design another two types of step size controllers, which are "RKSh #I" and "RKSh #PI".

Since England used two steps to calculate the numerical solution, it is fair to compare the "RKEn #" types with that adaptive controller which depends on doubling the step size. First, we take a single step of size h and use the four stages of Eq (3.13) in order to calculate the solution u^{n+1} using Eq (3.14). Second, we take two steps of size h/2 to recalculate the solution $\hat{u}^{n+1}$ using Eqs (3.13) and (3.14) two times. The local error can be simply estimated now as in Eq (3.6), and then Eq (3.7) is used to calculate err^{n+1}. If the error norm is tolerable, then the step size is accepted, and we again have the same three possibilities as in Subsection 3.1 to calculate the new solution. The first possibility is to accept the solution u^{n+1} resulting from taking a single step. In this case, we can design two controllers based on Eqs (2.2)–(2.4). Those adaptive controllers will be denoted as "RKdoubling 1 # I" and "Rkdoubling 1 # PI". The second possibility is to accept the solution $\hat{u}^{n+1}$, and this will be denoted again with "Rkdoubling 2 # I" and "Rkdoubling 2 # PI". The third one is to use Richardson extrapolation:

$$\tilde{u}^{n+1} = \hat{u}^{n+1} + \left(\frac{\hat{u}^{n+1} - u^{n+1}}{2^p - 1}\right), \tag{3.19}$$

Where p is the order of the method, which is four in this scheme. Based on the last formula, we can design two adaptive controllers, and they will be denoted as "RKdoubling 3 # I" and "RKdoubling 3 # PI".

3.4. Group D: Second-order LNe3 method

In [2] we introduced a new family of explicit methods, dealing with the spatially discretized heat equation, or generally, any system of first order linear ODEs. In this Subsection, we explain the core ideas of the LNe3 method, which allows us to introduce the necessary schemes for designing an adaptive step size controller, but a more detailed explanation can be found in [2] and [3]. Unlike the previous subsections, this method will be explained when it is applied to a system of ODEs instead of the single differential equation.

For the system in Eq (1.4), the LNe3 method can be summarized in three stages:

Stage 1: We take a time step of size h and calculate the variable u_i^{n+1}, assuming that the neighbors u_j are not changing during that time step. This means that $u_j^n = u_j^{n+1}$ during the time $\Delta t = t^{n+1} - t^n = h$, while $j \neq i$, and the index j refers to the neighbors of the cell with index i. That assumption decouples the system (1.4), converting it to an uncoupled system of ODEs of the following form:

$$\frac{du_i}{dt} = a_i - \frac{u_i}{\tau_i} \tag{3.20}$$

where

$$a_i = \sum_{j \neq i} \frac{u_j^n}{R_{ij} C_i} + q_i \quad \text{and} \quad \tau_i = \frac{C_i}{\sum_{j \neq i} \frac{1}{R_{ij}}}. \tag{3.21}$$

Eq (3.20) can be solved analytically, and we can use the solution as a predictor value for the next stage. The solution is

$$u_i^{n+1,predict} = u_i^n e^{-\frac{h}{\tau_i}} + a_i t_i \left(1 - e^{-\frac{h}{\tau_i}} \right). \tag{3.22}$$

Stage 2: Here, we assume that the neighboring variable u_j, of the actual variable u_i, is changing linearly during the time $\Delta t = h$. For each cell of index i, we introduce the effective slope:

$$s_i = \frac{a_i^{\text{predict}} - a_i}{h}, \tag{3.23}$$

where

$$a_i^{\text{predict}} = \sum_{j \neq i} \frac{u_j^{n+1,\text{predict}}}{R_{ij} C_i} + q_i \,. \tag{3.24}$$

Using the new assumption, we get a new system of uncoupled ODEs:

$$\frac{du_i}{dt} = s_i t + a_i - \frac{u_i}{\tau_i} \,. \tag{3.25}$$

Again, Eq (3.25) can be solved analytically, and the solution is

$$u_i^{n+1} = u_i^n e^{-\frac{h}{\tau_i}} + \left(a_i t_i - s_i \tau_i^2\right)\left(1 - e^{-\frac{h}{\tau_i}}\right) + s_i \tau_i h \,. \tag{3.26}$$

The scheme using Eq (3.26) has second order temporal convergence and is called LNe2.

Stage 3: We use the values of Eq (3.26) as predictors for this stage. We substitute these values in Eq (3.24) and then in (3.23) to calculate the new values $a_i^{predict}$ and s_i, respectively. Now, we can use Eq (3.26) with the new values to obtain an algorithm which will be denoted as "non-adaptive LNe3". This scheme is still of the second order but usually more accurate, as one can see from the truncation error, which is published in [26]. Since the stages serve as estimations of the solutions at the end of the time step, it is straightforward to organize an adaptive version of the LNe3 method. If the solutions produced in the second and third stages are denoted by u_i^{n+1} and $\hat{u}_i^{n+1}$, the local error can be estimated as before:

$$LE_i = \left|\hat{u}_i^{n+1} - u_i^{n+1}\right| . \tag{3.27}$$

Substituting Eqs (3.26) and (3.27) into (2.1) and considering Eqs (2.2)–(2.4), adaptive controllers will be applied, and they will be denoted as "ALNe3 #I" and "ALNe3 #PI".

We must note that when Fehlberg introduced his adaptive step size controller, many practitioners questioned the robustness of the method at that time. They thought that it was risky to estimate the local error using the same evaluation points. Later, experiments showed that this concern was not a problem in practice [25, p. 716]. Since the embedded LNe3 method is new, one might have the same concern. As we can see later, our experiments showed a very stable performance for that method.

4. Numerical experiments

The numerical solution and the reference solution are compared only at t_{fin}, which is the final time of the simulation and will be specified later. We measure the accuracy using the global L_∞ error, which is the maximum of the absolute difference between the reference temperature $u_{\text{j}}^{\text{ref}}$ and the temperature $u_{\text{j}}^{\text{num}}$ calculated by our numerical methods at the final time:

$$\text{Error}(L_\infty) = \max_{1 \le \text{j} \le N} \left| u_{\text{j}}^{\text{ref}}(t_{\text{fin}}) - u_{\text{j}}^{\text{num}}(t_{\text{fin}}) \right| . \tag{3.28}$$

In the first experiment, we will test the previously described methods in the case of a linear diffusion equation in the absence of the heat source, which yields a non-stiff system of ODEs after spatial discretization. In the second experiment, a linear diffusion equation in inhomogeneous media will be tested. The third experiment will treat the problem of a moving heat source.

The simulations are conducted using MATLAB R2020b software on a desktop computer with an Intel Core (TM) i11-11700F. Since the analytical solution does not exist for the systems we examine, the reference solution was calculated by the implicit ode15s solver setting very stringent error tolerance ('RelTol' and 'AbsTol' were both 10^{-14}).

4.1. Experiment 1: Non-stiff linear diffusion equation

In this experiment, we consider Eq (1.4) in 2 space dimensions $(x,y,t) \in [0, 1] \times [0, 1] \times [0, 2 \times 10^{-3}]$, subjected to zero Neumann boundary conditions, and in the absence of the heat source, $q = 0$. The space domain was divided into $N = N_x \times N_y = 50 \times 50$, and thus we have 2500 cells. The initial conditions were generated randomly using the built-in function 'rand' in MATLAB, $u_i(0) = rand$. The 'rand' function generates random numbers uniformly distributed in the interval [0,1]. The resistances and capacities were set as $Rx_i = Rz_i = 1$, $C_i = 10^{-3}$. The stiffness ratio of the introduced system is roughly 4×10^4, and the CFL limit for the explicit Euler scheme is 2.5×10^{-4}.

For all the groups of methods, Figures 1–4 show that the adaptive controllers of type (I) achieve approximately the same or slightly better performance compared to the controllers of type (PI) when both use the same method for estimating the local error. For example, as we can see from Figure 2, the curves of the controllers "RKdoubling 1 # I" and "RKdoubling 1 # PI" are almost identical. From Figures 1 and 3, we can clearly see that using Eq (3.19), which was suggested in Theorem 4.1 in [22], improved the performance of the algorithms based on the step doubling technique. However, the embedded Runge-Kutta-Dormand-Prince adaptive controller showed better performance than all the types of adaptive controllers based on the step doubling technique, as we can see in Figure 1. That is not a surprising result and was clearly stated in [15, p. 911]. Another important observation is related to the England and Shampine methods of group C. Although England stated in his work [12] that his method is valid "*when applied to a general system of differential equations*," our numerical experiments appear to contradict his claims. As we can see in Figure 3, the adaptive controllers based on England and Shampine showed poor performance when they were applied to our suggested system (1.4). Scraton suggested a new scheme (3.12), Eq (9) in his work [14], to increase the order of the method. As we can see in Figure 2, the non-adaptive scheme (3.12), which is the red color curve, showed unstable behavior. However, the adaptive schemes, which are RKSc 2 # I and RKSc 2 # PI, showed a stable performance but without improving the accuracy, if they are compared to "RKSc 1 # I" and "RKSc 1 # PI".

For each group of methods, we can see that the non-adaptive scheme is faster than the adaptive controller. Here, a question arises: Why do we use the adaptive controller if the non-adaptive scheme is faster? The subsequent discussion will show that the non-adaptive Runge-Kutta scheme is vulnerable, and the stability can be easily violated when small changes in the conditions or parameters of the experiment take place. To illustrate that, the time domain of the experiment, 2×10^{-3}, was replaced by 2×10^{-1} while all other settings and conditions remained the same. Figure 5 shows the performance of the methods of group A. We can see that after we made a small change in the time domain, the behavior of "non-adaptive DPRK5(4)" became unstable, and its curve blew up and became discontinuous. Despite the fact that the behavior of the adaptive controllers changed after we changed

the time domain, their performance remained stable. It does indeed look like the running time, in the case of adaptive controllers, is relatively independent of the required accuracy. This point is out of the scope of this study and one can see a good explanation in Appendix D of [27]. So, the advantage of using the adaptive controller is not always about reducing the computational time, but it is sometimes more about providing reliable results when the non-adaptive scheme fails.

The "non-adaptive LNe3" method is excluded from the last discussion, and in the second experiment we will show that this method remains stable regardless of the conditions of the system. It was proved mathematically and verified by numerical experiments that the method is unconditionally stable [2].

It is true that non-adaptive DPRK5 is faster than adaptive schemes when the time interval is $[0, 2 \times 10^{-3}]$, and this is advantageous from the point of view of the computational cost. However, in practice, the adaptive controllers still have an advantage even when they are slower than non-adaptive controllers. Suppose that the user needs a specific accuracy for the solution. Actually, in the case of a non-adaptive scheme, the user should choose an appropriate step size to achieve that accuracy, which requires information about the stiffness ratio of the system and the CFL limit, etc. Generally, the ordinary users of the simulation software are not mathematicians and cannot calculate the appropriate step size for the required accuracy. They are usually engineers who are interested in the output of the software but not in the input. Moreover, for some methods, it is so difficult, if not impossible, to calculate the CFL limit, even for mathematicians. In this case, the adaptive controller will allow the user to define a specific tolerance that can guarantee the required accuracy without being involved in complicated calculations.

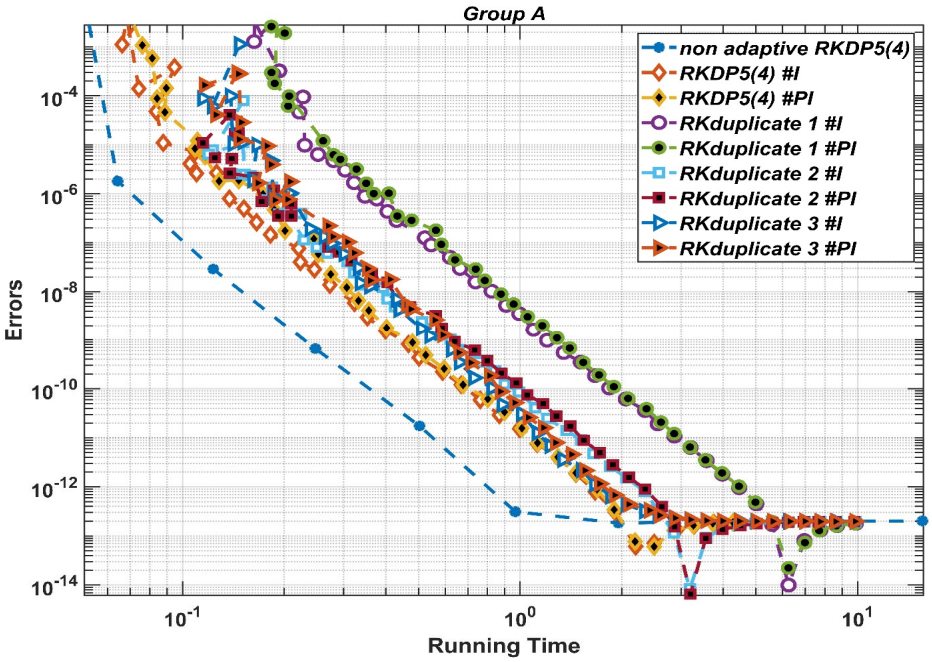

Figure 1. Experiment 1: L_∞ errors as a function of the running time for group A.

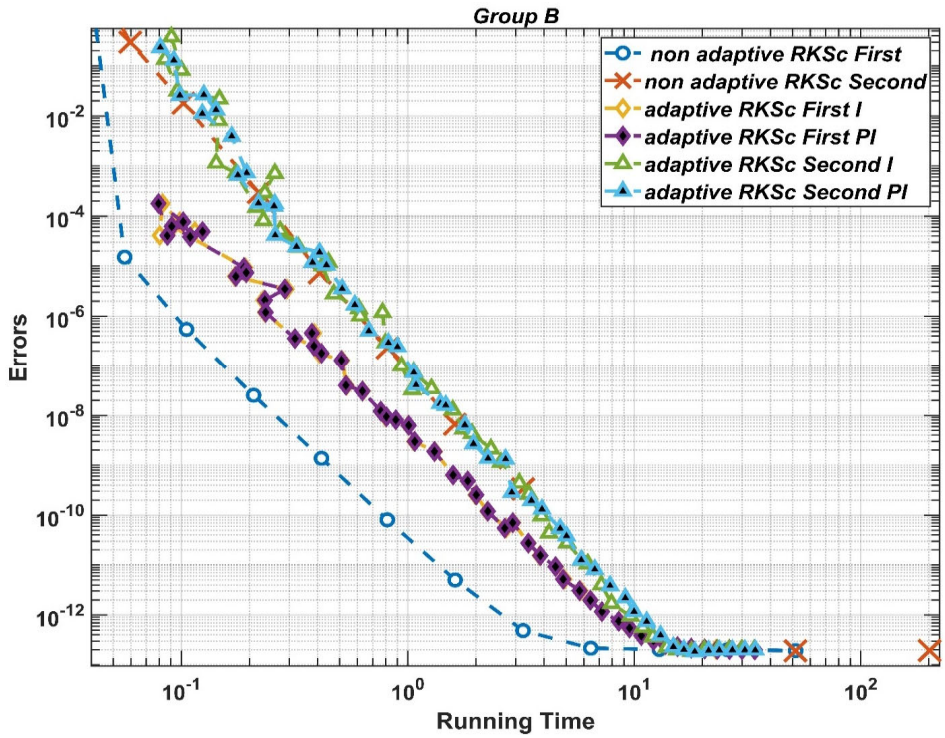

Figure 2. Experiment 1: L_∞ errors as a function of the running time for group B.

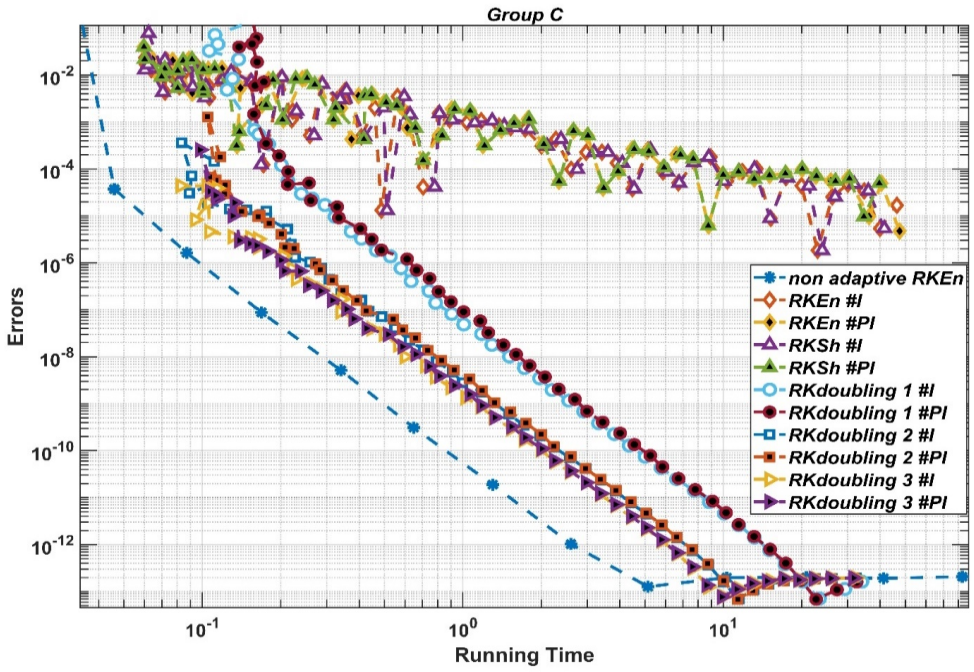

Figure 3. Experiment 1: L_∞ errors as a function of the running time for group C.

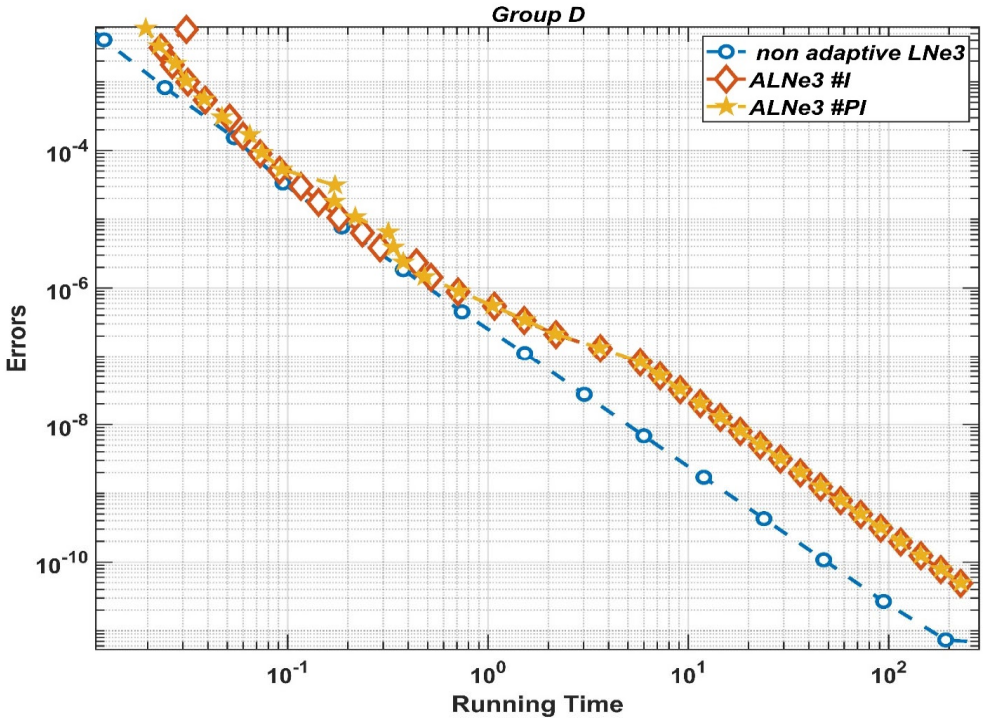

Figure 4. Experiment 1: L_∞ errors as a function of the running time for group D.

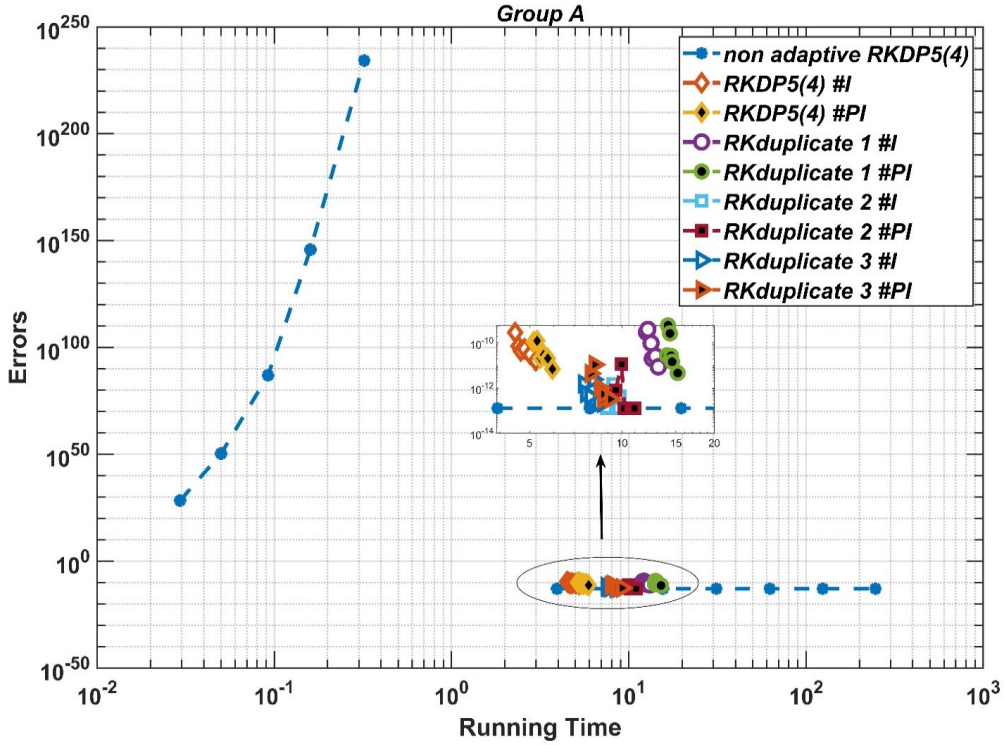

Figure 5. Experiment 1: L_∞ errors as a function of the running time for group A.

As we mentioned previously, we did not experience infinite loops in any of our numerical experiments. For example, in the case of DPRK5(4) #I for the current experiment, for high tolerance,

which guarantees error of order 10^{-13}, the observed number of rejected time-steps was 10, while the number of accepted time-steps was 1495. The largest number of successive rejected time-steps, out of the 10 rejected time-steps, was 5. On the other hand, when the tolerance guarantees an error of order 10^{-2}, the number of accepted time-steps was found to be 483 while the number of rejected time-steps was 20. The largest number of successive rejected time-steps, out of the 20 rejected time-steps, was 5. Table 1 shows some data for some methods as an illustrative example.

Table 1. The number of rejected and accepted time-steps for different methods at specific tolerances.

The method	TOL	Max error	Number of accepted time-steps	Number of rejected time-steps	Largest number of successive rejected time-steps
DPRK5(4) #I	1.25×10^{-1}	4.9×10^{-2}	483	20	5
DPRK5(4) #I	7.8×10^{-3}	10^{-3}	484	28	4
DPRK5(4) #I	9.09×10^{-13}	7.8×10^{-13}	941	22	5
ALNe3 #I	2.3×10^{-7}	3.8×10^{-5}	3003	5	5
RKSc 2 # I	1.2×10^{-4}	3.2×10^{-5}	555	98	4

4.2. Experiment 2: Stiff linear diffusion equation

In this experiment we consider Eq (1.4) in 2 space dimensions $(x,y,t) \in [0,1]\times[0,1]\times[0,\ 2\times10^{-4}]$, subjected to zero Neumann boundary conditions, and in the absence of the heat source, $q = 0$. The space domain was divided into $\mathrm{N} = N_x \times N_y = 20 \times 20$, and thus we have 400 nodes. The capacity and the resistances obeyed the following formula:

$$C = \left(\left(10^{-4}-1\right)x+1\right)10^{-4}, R_x = \left(10^{-4}-1\right)x+1, R_y = y+1$$

The stiffness ratio of the resulting system is 1.05×10^9, while the CFL limit is 9×10^{-10}. For such a relatively stiff system, all the non-adaptive schemes based on Runge-Kutta showed a poor performance. They can provide a reliable result only when the time step is very small, which increases the cost of the computations. The adaptive controllers which used England or Shampine methods for estimating the local error also showed poor performance when they are compared to those adaptive controllers which used the step doubling technique. As we can see in Figure 6, the highest accuracy that the England or Shampine formulas could reach was of the order 10^{-8}, while it is of the order 10^{-13} if the step doubling technique is used. Also, the performance of the controllers of type (I) was identical to and sometimes even better than the performance of those of type (PI). For the sake of comparison, in Figure 7 we selected the most accurate methods of groups A, B and C, as well as the methods of group D.

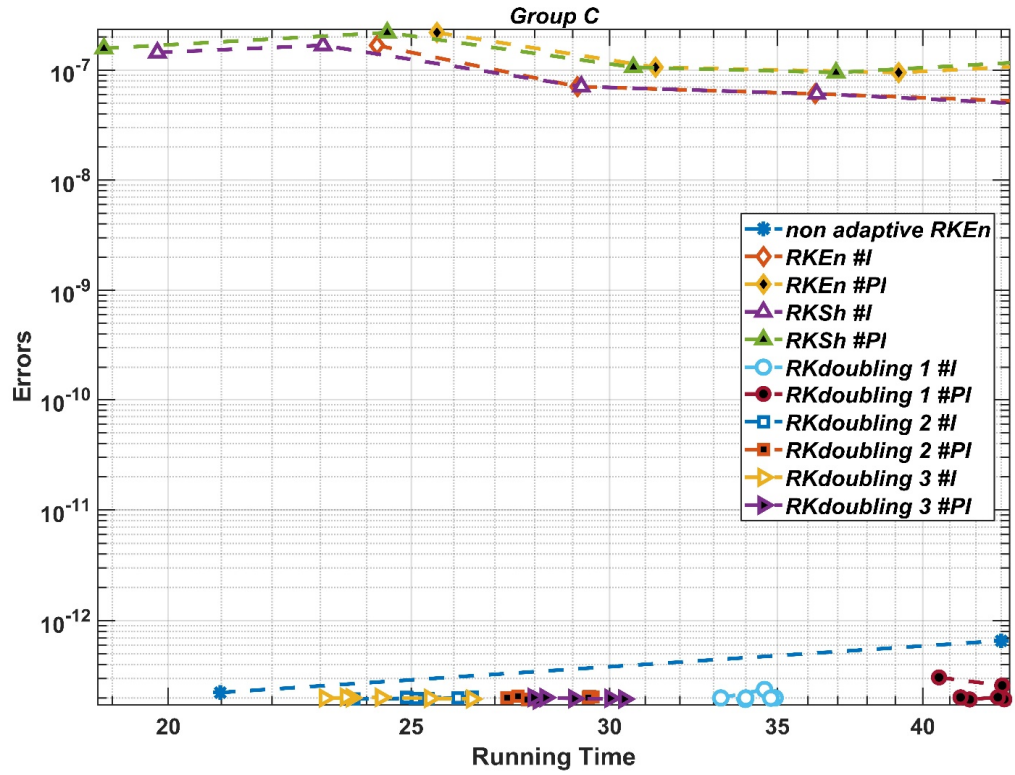

Figure 6. Experiment 2: L_∞ errors as a function of the running time for group C.

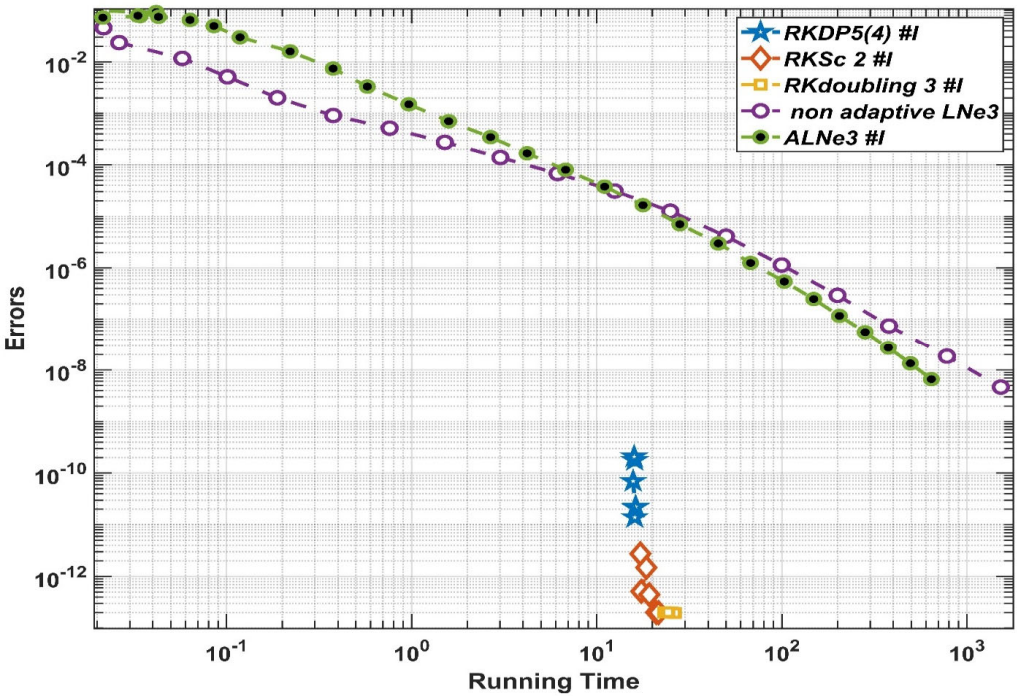

Figure 7. Experiment 2: L_∞ errors as a function of the running time for the selected methods.

4.3. Experiment 3: Stiff linear diffusion equation with a moving heat source

In this experiment, we consider Eq (1.4) in 2 space dimensions $(x,y,t) \in [0,1]\times[0,1]\times[0,\ 2\times10^{-5}]$, subjected to zero Neumann boundary conditions. The space domain was divided into $N = N_x \times N_z = 30 \times 30$, and thus we have 900 cells. The initial conditions are generated randomly using the built-in function 'rand' in MATLAB $u_i(0) = rand$. The capacity and the resistance obeyed the following form:

$$C = \left(\left(10^{-6}-1\right)x+1\right)10^{-4},\ R_x = \left(10^{-6}-1\right)x+1,\ R_y = y+1$$

The stiffness ratio of the resulting system is roughly 1.5×10^{11}, while the CFL limit is 6.4×10^{-12}. Here, we consider a moving Gaussian point heat source which takes the formula

$$q = q_{\max}\, \mathrm{e}^{-\left(\frac{(z+z_0-v_z t)+(x+x_0-v_x t)}{r^2}\right)},$$

where q_{max} is the maximum heat flux at the center of the heat source, r is the effective heating radius of the heat source, v_x and v_z are the velocities of the heat source in x and z directions, respectively, and (x_0,z_0) is the initial position of the heat source. The parameters of the heat source are set as

$$\left.\begin{array}{l} q_{\max} = 10^6 \\ v_x = 25\times10^3, v_z = 0 \\ (x_0, z_0) = (0,-0.5) \end{array}\right\}$$

This means that the heat source will move with a constant speed along the positive direction of the x axis. The effective heating radius is chosen to be $r = 5\Delta x = 5\Delta z$ to ensure that there are at least four nodes inside the effective heating diameter. Figure 8 shows the contour of the temperature distribution at the end of the time interval, and the trace of the heating process refers to the trajectory of the heat source.

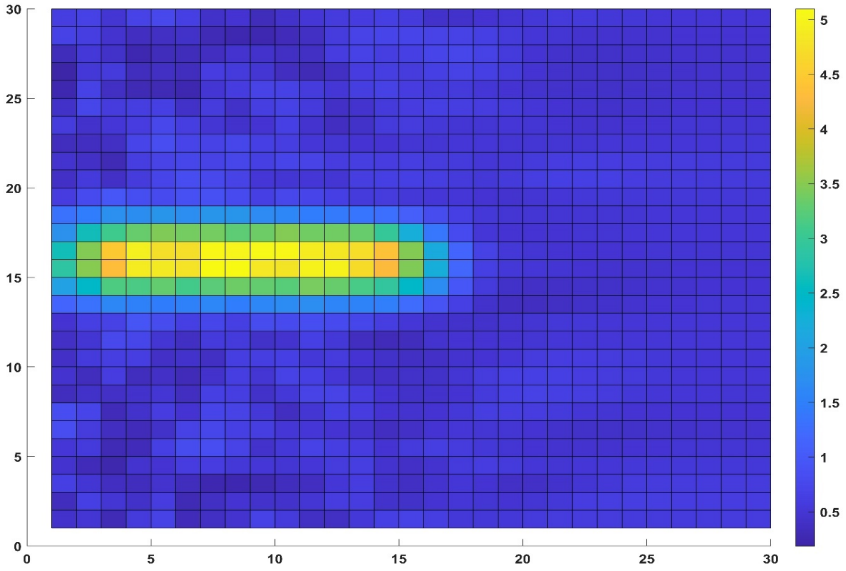

Figure 8. Experiment 3: The contour of the temperature distribution at the end of the time interval.

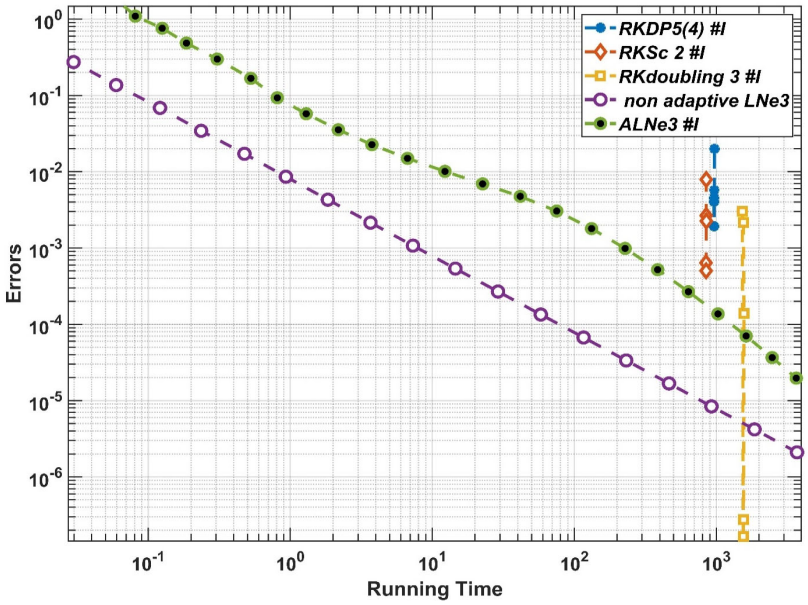

Figure 9. Experiment 3: L_∞ errors as a function of the running time.

We can clearly see from Figure 9 that the LNe3 and its adaptive scheme are much faster than the adaptive schemes of Runge-Kutta when high accuracy is not required, while adaptive Runge-Kutta schemes are more applicable when the desired accuracy goes beyond a certain level (which is of order in our experiment).

5. Conclusions

1) For solving a system of ODEs, such as the one in Eq (1.4), adaptive schemes using (PI) controller do not have any advantage compared to the same schemes using (I) controller. Moreover, (I) type is relatively faster than (PI) type in some cases, as the second experiment shows.
2) In his work [12], England aimed to overcome the shortcoming of Scraton's formula and to create a formula valid for a general system of ODEs. The scheme that England suggested is less accurate than the one suggested by Scraton when they are used to solve our system.
3) Scraton suggested his formula (3.12) that increases the order of the method when solving a single ordinary differential equation. That formula showed poor performance when it was applied to our system (1.4). However, his formula for estimating the local error is valid, and it is still better and less complicated than England's formula when it is used to solve Eq (1.4).
4) Our first numerical experiment shows that the method of duplicating time steps yields higher accuracy than the procedure of England when both use the same approach for changing the step size. Since both methods use the same approach for selecting the step size, this difference in accuracy can be explained by the inefficiency of England's formula for estimating the local error. So, we are skeptical about the result that Shampine and Watts came up with in their paper [7]. They claimed that the methods of the duplicating and England's method could be compared since they estimate the local error per two steps, and their *"accuracies are identical"*.
5) When solving our system, Richardson extrapolation increased the accuracy of the technique of time step duplication.

6) Embedded methods, such as the Dormand-Prince Runge-Kutta method, are still more efficient than the technique of duplicating the steps even after we consider Richardson extrapolation.
7) The LNe3 method remained perfectly stable even if the system had a high stiffness, as well as in the presence of a moving heat source.
8) According to our experience in numerical methods, the non-adaptive scheme can be sometimes, if not often, faster than its corresponding adaptive schemes when it is applied to Eq (1.4). Despite that fact, the adaptive schemes still have the advantage of enabling the user to define a desired level of accuracy, while the non-adaptive ones do not provide such a facility.
9) Some authors recommended integrating the parabolic partial differential equations by explicit methods even if they are only conditionally stable [28]. However, the Runge-Kutta methods are only conditionally stable, and the non-adaptive versions are not reliable. A small change in the parameters, e.g., increasing the final time of the simulation, can make them unstable. On the other hand, the adaptive schemes of Runge-Kutta may automatically select the proper step size, which can reach the stability region. This is typical for strict tolerances, but in the case of loose tolerances, the adaptive RK solvers can also produce huge errors.
10) Unlike the Runge-Kutta methods, the LNe3 method is unconditionally stable. As a result, the adaptive scheme of LNe3 will sometimes lose its advantage, and the time step selector can be redundant. However, it can be used to help the user to reach the desired accuracy. However, we do not say that the adaptive scheme of LNe3 is restricted to the previous role because in some conditions it was faster than the non-adaptive Lne3, as we saw in the second experiment.
11) It is clear that if the accuracy requirement is only moderate, but the speed is important, unconditionally stable explicit schemes should be used instead of explicit RK methods. If very high accuracy is required, then either the RKDP5(4) or, in stiff cases, the Lne3 method can be recommended. The explicit Lne3 method, which is unconditionally stable, can be used as a reliable integrator of the parabolic equations even in very extreme conditions.

Acknowledgments

This research was supported by the ÚNKP-21-3 new national excellence program of the ministry for innovation and technology from the source of the national research, development and innovation fund for M. S., and by the EU and the Hungarian state, co-financed by the ERDF in the framework of the GINOP-2.3.4-15-2016-00004 project for E. K.

Conceptualization, methodology, and supervision: E. K. Investigation and the first draft of the manuscript: M. S. Supervision and project administration: N. K. V. All authors commented on previous versions of the manuscript and approved the final manuscript.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. E. Kovács, Á. Nagy, M. Saleh, A set of new stable, explicit, second order schemes for the non-stationary heat conduction equation, *Mathematics*, **9** (2021), 2284. https://doi.org/10.3390/math9182284
2. E. Kovács, A class of new stable, explicit methods to solve the non-stationary heat equation, *Numer Methods Partial Differ Equ*, **37** (2021), 2469–2489. https://doi.org/10.1002/num.22730
3. E. Kovács, Á. Nagy, M. Saleh, A new stable, explicit, third-order method for diffusion-type problems, *Adv Theory Simul*, **5** (2022), 2100600. https://doi.org/10.1002/adts.202100600
4. S. Savović, B. Drljača, A. Djordjevich, A comparative study of two different finite difference methods for solving advection–diffusion reaction equation for modeling exponential traveling wave in heat and mass transfer processes, *Ricerche di Matematica*, **71** (2022), 245–252. https://doi.org/10.1007/s11587-021-00665-2
5. S. Conde, I. Fekete, J. N. Shadid, *Embedded error estimation and adaptive step-size control for optimal explicit strong stability preserving Runge--Kutta methods*, [Preprint], (2018) [cited 2023 Mar 29]. Available from: https://doi.org/10.48550/arXiv.1806.08693.
6. L. F. Shampine, Error estimation and control for ODEs, *J Sci Comput*, **25** (2005), 3–16. https://doi.org/ 10.1007/bf02728979
7. L. F. Shampine, H. A. Watts, Comparing error estimators for Runge-Kutta methods, **25** (1971), 445–455.
8. R. H. Merson, An operational methods for study of integration processes, *Weapon Research Establishment Conference on Data Processing*, **1** (1957), 110–125.
9. L. F. Shampine, Local extrapolation in the solution of ordinary differential equations, *Math Comput*, **27** (1973), 91. https://doi.org/ 10.2307/2005249
10. J. C. Butcher, P. B. Johnston, Estimating local truncation errors for Runge-Kutta methods, *J Comput Appl Math*, **45** (1993), 203–212. https://doi.org/ 10.1016/0377-0427(93)90275-G
11. J. H. Verner, Explicit Runge–Kutta methods with estimates of the local truncation error, *SIAM J Numer Anal*, **15** (1978), 772–790. https://doi.org/ 10.1137/0715051
12. R. England, Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations, *Comput J*, **12** (1969), 166–170. https://doi.org/ 10.1093/comjnl/12.2.166
13. A. S. Chai, Error estimate of a fourth-order Runge-Kutta method with only one initial derivative evaluation, *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, 1968, 467. https://doi.org/ 10.1145/1468075.1468144
14. R. E. Scraton, Estimation of the truncation error in Runge-Kutta and allied processes, *Comput J*, **7** (1964), 246–248. https://doi.org/ 10.1093/comjnl/7.3.246
15. W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes The Art of Scientific Computing*, Cambridge: Cambridge University Press, 2007.
16. K. Gustafsson, M. Lundh, G. Söderlind, API stepsize control for the numerical solution of ordinary differential equations, *BIT*, **28** (1988), 270–287.
17. K. Gustafsson, Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods, *ACM Trans Math Softw*, **17** (1991), 533–554.

18. G. Söderlind, Automatic control and adaptive time-stepping, *Numer Algorithms*, **31** (2002), 281–310.
19. G. Söderlind, Digital filters in adaptive time-stepping, *ACM Trans Math Softw*, 29 (2003), 1–26.
20. G. Söderlind, L. Wang, Adaptive time-stepping and computational stability, *J Comput Appl Math*, **185** (2006), 225–243. https://doi.org/ 10.1016/j.cam.2005.03.008
21. T. Ritschel, Numerical Methods For Solution of Differential Equations, (Denmark), Doctoral Thesis of Technical University of Denmark, Lyngby, 2013.
22. E. Hairer, S. P. Nørsett, G. Wanner, *Solving Ordinary Differential Equations I*, Berlin: Springer, 1993. https://doi.org/ 10.1007/978-3-540-78862-1
23. I. Fekete, S. Conde, J. N. Shadid, Embedded pairs for optimal explicit strong stability preserving Runge–Kutta methods, *J Comput Appl Math*, **412** (2022), 114325. https://doi.org/ 10.1016/j.cam.2022.114325
24. David F. Griffiths and Desmond J. Higham, *Numerical methods for ordinary differential equations: initial value problems*, London: Springer, 2010.
25. W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical recipes in C : the art of scientific computing*, Cambridge: Cambridge University Press, 1992.
26. Á. Nagy, J. Majár, E. Kovács, Consistency and convergence properties of 20 recent and old numerical schemes for the diffusion equation, *Algorithms*, **15** (2022), 425. https://doi.org/ 10.3390/a15110425.
27. J. Feldman, A. Rechnitzer, E. Yeager, D.3: Variable Step Size Methods, In: *CLP-2 Integral Calculus*, (2021), 91843. Available from: https://math.libretexts.org/@go/page/91843.pdf.
28. S. Essongue, Y. Ledoux, A. Ballu, Speeding up mesoscale thermal simulations of powder bed additive manufacturing thanks to the forward Euler time-integration scheme: A critical assessment, *Finite Elem Anal Des*, **211** (2022), 103825. https://doi.org/ 10.1016/j.finel.2022.103825

AIMS Press