Journal of Scientific Computing (2024) 98:39
https://doi.org/10.1007/s10915-023-02426-9

Unconditionally Positive, Explicit, Fourth Order Method for the Diffusion- and Nagumo-Type Diffusion–Reaction Equations

Endre Kovács[1] · **János Majár**[1] · **Mahmoud Saleh**[1]

Received: 27 September 2022 / Revised: 7 October 2023 / Accepted: 28 November 2023 /
Published online: 3 January 2024

Abstract
We present a family of novel explicit numerical methods for the diffusion or heat equation with Fisher, Huxley and Nagumo-type reaction terms. After discretizing the space variables as in conventional method of lines, our methods do not apply a finite difference approximation for the time derivatives, they instead combine constant- linear- and quadratic-neighbour approximations, which decouple the ordinary differential equations. In the obtained methods, the time step size appears in exponential form in the final expression with negative coefficients. In the case of the pure heat equation, the new values of the variable are convex combinations of the old values, which guarantees unconditional positivity and stability. We analytically prove that the convergence of the methods is fourth order in the time step size for linear ODE systems. We also prove that the concentration values in the case of Fisher's and Nagumo's equations lie within the unit interval regardless of the time step size. We construct an adaptive time step size time integrator with an extremely cheap embedded error control method. Several numerical examples are provided to demonstrate that the proposed methods work for nonlinear equations in stiff cases as well. According to the comparisons with other solvers, the new methods can have a significant advantage.

Keywords Diffusion equation · Explicit time-integration · Stiff equations · Nagumo's equation · Unconditional stability

1 Introduction

1.1 The Studied Problems

Diffusive transport of particles or heat has a fundamental role in several scientific and engineering applications [1–3]. It is well known that the so-called heat or diffusion equation, which describes diffusion and conductive heat transfer, is the following linear parabolic

✉ Endre Kovács
kendre01@gmail.com

[1] Institute of Physics and Electrical Engineering, University of Miskolc, Miskolc, Hungary

partial differential equation (PDE),

$$c\rho\frac{\partial u}{\partial t} = \nabla\cdot(k\nabla u) + c\rho q. \tag{1.1}$$

here $u : \mathbb{R}^3 \times \mathbb{R} \mapsto \mathbb{R}$; $(\vec{r}, t) \mapsto u(\vec{r}, t)$ is the unknown function, $q : \mathbb{R}^3 \times \mathbb{R} \mapsto \mathbb{R}$; $(\vec{r}, t) \mapsto q(\vec{r}, t)$ and $u(\vec{r}, t = 0) = u^0(\vec{r})$ are given functions, while $c = c(\vec{r}, t)$, $\rho = \rho(\vec{r}, t)$, and $k = k(\vec{r}, t)$ are known nonnegative functions. The boundary conditions will be specified at the concrete numerical examples.

In one space dimension, provided that k is independent of the space variable, Eq. (1.1) can be written into the simple form

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2} + q, \tag{1.2}$$

where $u : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$; $(x, t) \mapsto u(x, t)$, $\alpha > 0$ is a constant.

In the case of diffusive mass transfer, u is the concentration of the particles. In the case of heat conduction, u denotes the temperature, $\alpha = k/(c\rho)$ is the thermal diffusivity, k, ρ, and c are the heat conductivity, the specific heat and the mass density, while q is the intensity of the heat sources (due to electromagnetic radiation, electric currents, etc.), respectively.

The diffusion equation and its generalizations, such as the advection–diffusion-reaction equation, can model mass transport in countless physical, chemical, and biological systems, for example charge carriers in semiconductors [4], atoms in carbon nanotubes [5], and proteins in embryos [6]. Furthermore, very similar equations or systems of equations are used to simulate fluid flow through porous media, such as moisture [7], ground water, or crude oil in underground reservoirs [8]. We are going to deal here with two nonlinear reaction–diffusion equations. The first one is Fisher's equation, also called Fisher-Kolmogorov-Petrovsky-Piskunov equation [9], which contains an additional nonlinear logistic reaction term besides the diffusion term:

$$\frac{\partial u}{\partial t} = \alpha\nabla^2 u + \beta u(1-u). \tag{1.3}$$

This equation is used to model the spreading of gene-variants in space, as well as the growth and spreading of misfolded proteins in neurophysiology [10], and the propagation of fronts in combustion processes [11]. The second nonlinear equation we use here contains a source term which can be a large and non-integer power of the variable u, and has the following form:

$$\frac{\partial u}{\partial t} = \alpha\nabla^2 u + \beta u\left(1-u^{\delta}\right)\left(u^{\delta}-\gamma\right), \tag{1.4}$$

where β, γ, and δ are nonnegative real numbers. This equation has applications not only in biology [12], but in chemical reactions [13] and nuclear reactor theory [14] as well. It is usually called the Nagumo, FitzHugh-Nagumo or the generalized Huxley equation [15]. In fact, Huxley's equation can be obtained from the Nagumo equation if one sets $\gamma = 0,\ \delta = 1$ as follows

$$\frac{\partial u}{\partial t} = \alpha\nabla^2 u + \beta u^2(1-u). \tag{1.5}$$

It was shown [16] that this equation is superior to Fisher's equation when the change in frequency of a new, advantageous recessive allele in a sexually reproducing population has to be modelled. Equation (1.5) is a special case of the Burgers-Huxley equation [17], which is used to model nonlinear wave phenomena.

1.2 On the Solution Methods

It is well known that several numerical methods have been developed and applied to solve the equations in question. Nevertheless, most of these are elaborated and tested under circumstances where the coefficients in the equations, such as the diffusivity α, do not depend on the space variable. In real applications, however, there are systems where the physical properties can be extremely different at neighbouring points, let us just think about a microprocessor. This means that the coefficients, and therefore the eigenvalues of the system matrix, may have a range of several orders of magnitude, thus the problem can be severely stiff. Although when combined with method of lines, the traditional explicit methods (such as the fourth-order Runge–Kutta method) can be very accurate [18], very small time step sizes have to be applied in stiff cases regardless of the measurement errors of the input data and the requirements on the accuracy of the output. This is because they are stable only if the time step size does not exceed a certain threshold number, the mesh Fourier number (also called Courant or CFL limit). Even the adaptive time step size solvers like ode23 and ode45 of MATLAB can run into instability [19] if the tolerance is not very small.

On the other hand, implicit methods have much better stability properties. That is why they are typically proposed to solve these equations [20–22]. For example, Ramos [23] successfully modified the standard explicit method to reach higher accuracy in the case of the one-dimensional Huxley's equation. However, the stability was not improved by the modification, and the implicit methods, especially the different versions of Crank-Nicolson schemes are still outperform the modified explicit method as well. He also applied different linearized-implicit techniques for one-dimensional [24] and multi-dimensional [25] diffusion-reaction problems. Manaa and Sabawi solved [26] $\delta = 1$ the version of Eq. (1.4) using Explicit (Euler) and Crank- Nicolson methods, and obtained that albeit the explicit method is faster, it is less stable and accurate than the Crank-Nicolson scheme. Kadioglu and Knoll solved coupled hydrodynamics and nonlinear heat conduction problems by treating the hydrodynamics explicitly and the heat conduction part implicitly [27]. They explain that this strategy, which is called IMEX, is typical for these kinds of problems. Its counterpart in the field of reservoir-simulation is the IMplicit Pressure Explicit Saturation (IMPES) approach [28], where the pressure-equation (mathematically similar to the diffusion equation) is solved implicitly, and the saturation is calculated explicitly. However, fully implicit methods are also proposed [29] for this problem.

The most serious problem with the implicit methods is that the solution of a system of algebraic equations is required at each time step, which cannot be straightforwardly parallelized. In one space dimension, when the number of nodes or cells is small and the matrix is tridiagonal, these calculations can be very fast and implicit methods are hard to compete with. However, the solution can be extremely time-consuming in the opposite case, and we note that the number of cells has already reached one trillion in reservoir simulations. Since the clock frequencies of CPUs halted in the last decades, which reinforced the tendency towards increasing parallelism in high-performance computing [30, 31], we believe that the attention towards the easily parallelizable explicit algorithms is going to increase.

The second problem with most of the methods, either explicit or implicit, is that they can lead to qualitatively unacceptable solutions, such as unphysical oscillations or negative values of the otherwise non-negative variables. These variables can be concentrations, densities, or temperatures measured in Kelvin, and the numerical methods should preserve their positivity. That is why Chen-Charpentier and others constructed and studied [32–34] the fully explicit and unconditionally positive finite difference (UPFD) scheme for linear advection–diffusion reaction equations. Then Kolev et al. [35] considered a model of cancer

migration and invasion, which consists of two PDEs with diffusion terms and an ordinary differential equation (ODE). They discretized one of the PDEs and the ODE implicitly while the remaining PDE was solved by an explicit scheme similar to the UPFD method. However, their method, as well as the original UPFD scheme, has only first order temporal accuracy. Chertock and Kurganov developed a positivity preserving scheme for a system of advection-reaction–diffusion equations describing chemotaxis/haptotaxis models, but their method is positivity preserving only if the time step size is below the rather low mesh Fourier number [36]. The situation is similar to the nonstandard finite difference schemes (NSFD), applied to cross-diffusion equations by Chapwanya et al. [37], and Songolo [38], and to the Fisher and the Nagumo equation by Agbavon et al. [12, 39]. The schemes of Agbavon will be tested in this paper as well.

There are explicit algorithms which are at the same time unconditionally stable, at least in the case of the linear heat equation. For example, the Alternating Direction Explicit (ADE) scheme [40] and the odd–even hopscotch algorithm [41, 42] have second order temporal accuracy. They are often quite accurate indeed, but not positivity preserving. Actually, for stiff systems, the odd–even hopscotch method can produce very large errors [43]. Moreover, both of them build on the regularity of the mesh, thus they lose their explicit nature otherwise.

Nonlinear equations of type (1.3) or (1.4) are treated by operation splitting approaches as well, such as Strang-splitting. The diffusion part is solved by an implicit method as if the nonlinear term does not exist. The nonlinear term is treated locally, solving the ODE, either analytically or via linearization. In this way, one can spare the Newton iterations due to the nonlinearity, but most of the above-mentioned problems remain. Moreover, the splitting implies larger errors and even the decoupling of the reaction and the diffusion processes in the case of large time steps [18].

To summarize, we do not know any explicit and unconditionally stable, let alone unconditionally positive method above second order, even for the linear diffusion equation. That is why we started to construct novel explicit methods based on a fundamentally new way of thinking. The obtained lower order members (the CNe, LNe, CCL and CLL methods, see [44–46], respectively) have already been published, and now we are going to present the fourth order members, for which we need the quadratic-neighbour approximation.

1.3 The Outline of the Paper

In Sect. 2, only the linear diffusion equation is considered. First we briefly recall the CNe and LNe schemes, because they make the first two stages of the new methods. In Sect. 2.2, we introduce the new algorithms for the linear diffusion equation, first for the simplest case (one dimensional, equidistant mesh), then in Sect. 2.3, for a general, arbitrary mesh as well. Then we start to analyse the properties of these new methods. The unconditional positivity and stability are proved in Sect. 2.4, then the truncation error is examined to determine the conditions for consistency and the order of convergence in 2.5. In Sect. 3, we show how to adapt the algorithms to the nonlinear cases. In Sect. 4, five numerical tests are presented in 1D cases where analytical solutions of the equations are used as reference solutions. In Sect. 5, we present the adaptive stepsize controller and then perform three numerical experiments for two space-dimensional stiff systems. Finally, we summarise our conclusions and write about our future research goals.

2 Description and Properties of the New Method for the Linear Diffusion Equation

We begin with the construction of the new schemes for the one dimensional Eq. (1.2). First, the space variable is discretized by creating $N \in \mathbb{N}^+$ equidistant nodes: $x_i = i\Delta x\,,\ \ i = 1,\ ...,\ N$. As in the most standard method of lines, the next step is the discretization of the space derivatives by the second order central difference formula:

$$\frac{\partial^2}{\partial x^2} f(x_i,\, t) \approx \frac{\frac{f(x_{i+1},t)-f(x_i,t)}{\Delta x} + \frac{f(x_{i-1},t)-f(x_i,t)}{\Delta x}}{\Delta x}. \tag{2.1}$$

Using (2.1) we obtain an ordinary differential equation (ODE) for each node $i = 1,\ ...,\ N$:

$$\frac{du_i}{dt} = \alpha \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} + q_i = \alpha \frac{u_{i\pm1} - 2u_i}{\Delta x^2} + q_i,$$

where $u_i : \mathbb{R} \mapsto \mathbb{R};\ t \mapsto u_i(t)$ are functions of the time variable, which is still continuous, while q_i is the value of the source term at node i. The $u_{i\pm1} = u_{i+1} + u_{i-1}$ notation will be used throughout the whole paper for the sake of brevity. This system of Eq. (2.2) has the following matrix-form:

$$\frac{d\vec{u}}{dt} = M\vec{u} + \vec{q}. \tag{2.2}$$

here M is a tridiagonal matrix with the usual elements: $m_{ii} = -\frac{2\alpha}{\Delta x^2}\,,\ \ m_{i,i+1} = m_{i,i-1} = \frac{\alpha}{\Delta x^2}\ \ (1 < i < N)$, while the first and last row is determined by the boundary condition. The ith element of the vector $\vec{u} : \{1,\ ...,\ N\} \times \mathbb{R} \mapsto \mathbb{R}^N$ is the function $u_i(t)$. If one calculates the eigenvalues of the matrix M, then the (nonzero) eigenvalues with the largest and smallest absolute value (let us say $\lambda_{\max}$ and $\lambda_{\min}$) can determine the stiffness ratio, which is $\lambda_{\max}/\lambda_{\min}$. Moreover, the maximum time step size above which the explicit Euler time integration will be unstable can be given as $h^{\mathrm{EE}}_{\mathrm{MAX}} = -2/\lambda_{\max}$. Similar but slightly larger mesh Fourier numbers hold for all explicit Runge–Kutta methods.

We define the usual mesh ratio as $r = \frac{\alpha h}{\Delta x^2} = -\frac{m_{ii}}{2}h\,,\ \ 1 < i < N$, with which we can write

$$\frac{du_i}{dt} = -2\frac{r}{h}u_i + \underbrace{\frac{r}{h}u_{i\pm1} + q_i}_{(*)} \tag{2.3}$$

At this point the time variable is also discretized, $t^n = nh\,,\ \ n \in \{0,\ 1,\ 2,\ ...,\ T\}$, and u_i^n is the numerical solution at x_i and t^n. We use h instead of Δt or k for the time step size because of the similarity of our methods to the method of lines. We emphasize here that the time derivatives will not be approximated by finite difference formulas as in the Finite Difference Methods (FDM). Instead, the ODE system (2.2) or (2.3) will be solved analytically assuming the time-variable to be continuous and the initial value at t^n is u_i^n. The constructed solution of ODE (2.3) is used to approximate the real $u\left(t^{n+1}\right)$ value. Due to this, the new methods are not always considered as FDMs.

2.1 The Already Known Constant- and Linear-Neighbour Type Approximations

Now the simplest member of our methods, the CNe algorithm [47] is briefly repeated via the following points.

(1a) When the new value u_i^{n+1} of the unknown function is calculated, we neglect that other variables, most importantly the neighbours u_{i-1}^n and u_{i+1}^n are also changing during the time step. This crude approximation means that u_j ($j \neq i$), and therefore the (*) term in (2.3) are considered to be constants (this is where the name of the method comes from). After this a set of *uncoupled*, linear ODEs remains:

$$\frac{du_i}{dt} = a_i - \frac{2r}{h}u_i, \tag{2.4}$$

where the unknown u_i is still a function of the (continuous) time in each equation with initial values u_i^n, while h is a parameter, but r/h is not a function of time. The quantity a_i has been introduced to condense information about the neighbours of cell i and the source term

$$a_i = \sum_{j\in\{i-1,\ i+1\}} m_{ij}u_j^n + q_i = \alpha\frac{u_{i\pm1}^n}{\Delta x^2} + q_i = \frac{r}{h}u_{i\pm1}^n + q_i. \tag{2.5}$$

We note again that u_{i-1}^n and u_{i+1}^n are the values of the neighbours at the beginning of the actual time step, thus the a_i values are considered as constants during the time step.

(1b) The Eq. (2.4) are very simple and it is straightforward to use their *analytical solution* at time $t = h$ to obtain the values of u at the end of the time step:

$$u_i^{n+1} = u_i^n \cdot e^{-2r} + \frac{ha_i}{2r}\left(1 - e^{-2r}\right). \tag{2.6}$$

Thus, for a one-dimensional homogeneous system with an equidistant mesh, the following one-stage method was introduced.

Algorithm 1, CNe method

$$u_i^{n+1} = u_i^n \cdot e^{-2r} + \left(\frac{u_{i\pm1}^n}{2} + \frac{h}{2r}q_i\right)\left(1 - e^{-2r}\right). \tag{2.7}$$

The CNe method has first order convergence in time [44]. If one uses a half time step to obtain predictor values, and then a full time step for the corrector values, both with the CNe formula, one will have the so-called CpC algorithm:

(2a) Instead of the constant-neighbour approximation, now we suppose that the neighbours of the node, and therefore the (*) term in (2.3) are changing *linearly*. It means that we have to solve the following uncoupled ODE system:

$$\frac{du_i}{dt} = s_i t + a_i - \frac{2r}{h}u_i, \quad (i = 1 \ldots N) \tag{2.8}$$

To determine the s_i values (the aggregated or effective slopes), we need a set of predictor values u_i^{pred}. These are obtained by the CNe algorithm and are valid at the end of the examined time step. So the slope for the neighbours can be calculated as

$$s_i = \frac{a_i^{\mathrm{pred}} - a_i}{h}. \tag{2.9}$$

Here

$$a_i^{\mathrm{pred}} = \sum_{\mathrm{j}\in\{i-1,\,i+1\}} m_{ij}u_j^{n+1,\mathrm{pred}} + q_\mathrm{i} = \frac{\alpha}{\Delta x^2}u_{i\pm1}^{\mathrm{n+1,pred}} + q_i = \frac{r}{h}u_{i\pm1}^{\mathrm{n+1,pred}} + q_\mathrm{i} \tag{2.10}$$

contains the predictor values of all the neighbours. The source terms are considered to be independent of time for the sake of simplicity.

(2b) The equation system (2.8) is a set of independent linear ODEs again, with the simple analytical solution:

$$u_i(t) = u_i^n e^{-2\frac{r}{h}t} - \left(\frac{h^2}{4r^2}s_i - \frac{h}{2r}a_i\right)\left(1 - e^{-2\frac{r}{h}t}\right) - \frac{h}{2r}s_i t.$$

This solution is used at the end of the time step $t = h$ to provide us with the corrector values. Steps 1a–b and then 2a–b make a two-stage (predictor–corrector) method, which is called "*linear-neighbour* method" [44] and abbreviated by LNe2 or LNe. After some rearrangement, we have the following scheme.

Algorithm 2, LNe method

Stage 1 The same as Algorithm 1.

$$Stage\,2 \quad u_i^{\mathrm{L},n+1} = u_i^n e^{-2r} + \frac{h}{2r}\left(\frac{1}{2r}\left(a_i - a_i^{\mathrm{pred}}\right) + a_i\right)\left(1 - e^{-2r}\right) - \frac{h}{2r}\left(a_i^{\mathrm{pred}} - a_i\right), \tag{2.11}$$

where a_i and a_i^{pred} are defined in (2.5) and (2.10).

This algorithm has second order convergence [44], and it is an important building block of our multistage methods.

2.2 The New Quadratic-Neighbour Approximation

Now we approximate the aggregate change of the neighbours, more precisely, the (*) term in (2.3), by a second order polynomial, which yields the following ODE for the node-variable:

$$\frac{du_i}{dt} = -\frac{2r}{h}u_i + w_i t^2 + s_i t + a_i \tag{2.13}$$

where $u(t = 0) = u^0$ and the values of w, s, and a contain information about the neighbours and the source term. The analytical solution of the initial value problem (2.13) at $t = h$ is

$$u(t) = e^{-2r}u^0 + \left(1 - e^{-2r}\right)\left(2w_i\left(\frac{2r}{h}\right)^3 - s_i\left(\frac{2r}{h}\right)^2 + a_i\frac{2r}{h}\right) + 2r\left(w_i h - 4w_i\frac{r}{h} + s_i\right) \tag{2.14}$$

In the case of the LNe method, we needed values of $\vec{u}$ at two different time-points to determine the "slope" coefficients s_i. Now, for each node, we need three values, f_0, f_1 and f_2, to determine the coefficients w, s and a. The quantity a_i is obviously $a_i = f_0 = {}^{\mathrm{r}}/_{\mathrm{h}}u_{i\pm1}^n + q_i$. We choose to obtain an estimation of the other two values at the middle and at the end of the actual time step by the LNe scheme described above, so the two missing function values are:

$$f_{1,i} = {}^{\mathrm{r}}/_{\mathrm{h}}u_{i\pm1}^{\mathrm{L,n+1/2}}u_i^n + q_i, \quad f_{2,i} = {}^{\mathrm{r}}/_{\mathrm{h}}u_{i\pm1}^{\mathrm{L,n+1}}u_i^n + q_i$$

Now we have the following simple system of equations:

$$w\frac{h}{4}^2 + s\frac{h}{2} + a = f_1 \text{ and } wh^2 + sh + a = f_2$$

The solution of this gives the unknown parameters for each node:

$$s_i = \frac{4f_{1,i} - f_{2,i} - 3a_i}{h} \text{ and } w_i = 2\frac{f_{2,i} - 2f_{1,i} + a_i}{h^2},$$

and with these, the new values of the solution can be calculated using (2.14). For the sake of simplicity and higher speed of the calculations, we introduce the following new quantities:

$$F_{1,2} = f_{1,2}\frac{h}{2r},\ A_i = a_i\frac{h}{2r},\ S_i = s_i\frac{h^2}{2r},\ W_i = w_i\frac{h^3}{2r}$$

Now, if someone uses quantities with dimensions, then the new quantities A, S and W have the same dimension as u. With these new notations, we have

$$\begin{aligned} u_i^Q &= e^{-2r}u_i^{\mathrm{n}} + \left(1-e^{-2r}\right)\left(2W_i\frac{2r}{h^3}\frac{h^3}{8r^3} - S_i\frac{2r}{h^2}\frac{h^2}{4r^2} + A_i\frac{2r}{h}\frac{h}{2r}\right) \\ &\quad + h\frac{h}{2r}\left(W_i\frac{2r}{h^3}h - 2W_i\frac{2r}{h^3}\frac{h}{2r} + S_i\frac{2r}{h^2}\right) \\ &= e^{-2r}u_i^{\mathrm{n}} + \left(1-e^{-2r}\right)\left(\frac{W_i}{2r^2} - \frac{S_i}{2r} + A_i\right) + W_i - \frac{1}{r}W_i + S_i \end{aligned}$$

We can summarize the obtained algorithms as follows:

Algorithm 3: three-stage constant-linear-quadratic neighbour (CLQ) method

Stage 1 Calculate $A_i = \frac{u_{i\pm1}^{\mathrm{n}}}{2} + \frac{h}{2r}q_i$, then take a full time step with the CNe method

$$u_i^{\mathrm{C}} = u_i^n e^{-2r} + A_i\left(1-e^{-2r}\right)$$

Stage 2 Calculate $A_i^{\mathrm{pred}} = \frac{u_{i\pm1}^{\mathrm{C}}}{2} + \frac{h}{2r}q_i$, then calculate a half as well as a full time step with the LNe method

$$\begin{aligned} u_i^{\mathrm{L}1/2} &= u_i^n e^{-r} + A_i\left(1-e^{-r}\right) + \frac{A_i^{\mathrm{pred}} - A_i}{2}\left(1 - \frac{1-e^{-r}}{r}\right) \\ u_i^{\mathrm{L}} &= u_i^n e^{-2r} + A_i\left(1-e^{-2r}\right) + \left(A_i^{\mathrm{pred}} - A_i\right)\left(1 - \frac{1-e^{-2r}}{2r}\right) \end{aligned}$$

Stage 3 Calculate $F_{1,i} = \frac{u_{i\pm1}^{L1/2}}{2} + \frac{h}{2r}q_i$ and $F_{2,i} = \frac{u_{i\pm1}^{L}}{2} + \frac{h}{2r}q_i$, then $S_i = 4F_{1,i} - F_{2,i} - 3A_i$ and $W_i = 2\left(F_{2,i} - 2F_{1,i} + A_i\right)$, and finally

$$u_i^{\mathrm{Q}} = e^{-2r}u_i^{\mathrm{n}} + \left(1-e^{-2r}\right)\left(\frac{W_i}{2r^2} - \frac{S_i}{2r} + A_i\right) + W_i\left(1 - \frac{1}{r}\right) + S_i. \tag{2.15}$$

One can use the CLQ result obtained at Stage 3 to add one more stage, which will be the CLQ2 method. For this, the midpoint values must be calculated first as follows:

$$u_i^{Q1/2} = e^{-r}u_i^n + \left(1-e^{-r}\right)\left(\frac{W_i}{2r^2} - \frac{S_i}{2r} + A_i\right) + \frac{W_i}{4} - \frac{W_i}{2r} + \frac{S_i}{2} \tag{2.16}$$

Algorithm 4: four-stage CLQ2 method

Stage 1–2–3 The same as in Algorithm 6.

Stage 4 Use formula (2.15) again to obtain the final values of the four-stage CLQ2 method, but now $u_i^{Q1/2}$ is obtained by (2.16), $F_{1,i} = \frac{1}{2}u_{i\pm1}^{\mathrm{Q}1/2} + \frac{h}{2r}q_i$, $F_{2,i} = \frac{1}{2}u_{i\pm1}^{\mathrm{Q}} + \frac{h}{2r}q_i$, while S_i and W_i must also be recalculated. This iteration stage can be further repeated (in the very same way as Stage 4) to get the CLQ3 method (5 stages altogether), the CLQ4 method (6 stages altogether), etc. We will show that although this iteration CLQn does not converge to the true solution as $n \to \infty$, it becomes more and more accurate due to the decreasing truncation error coefficients.

We have expressed the new values u_i^{n+1} with the old values u_j^n for the new algorithm using the Mathematica software, but for the sake of brevity we present it only for the CLQ algorithm, because for the other schemes they are much longer expressions.

$$
\begin{aligned}
u_i^{n+1} = {} & \frac{e^{-6r}}{8r^3}\left\{-2-r+e^r\left(4(1+r)+e^r\left(\begin{matrix}2+r(-1+6r)+4e^{3r}\left(1+r-2r^2\right)-8e^r\left(1+r+r^2\right)\\ +e^{4r}\left(-2-3r+6r^2\right)+e^{2r}\left(2+r\left(5+4r+8r^2\right)\right)\end{matrix}\right)\right)\right\}u_i^n \\
& + \frac{e^{-6r}}{32r^3}\left\{\begin{matrix}3(2+r)-12e^r(1+r)+4e^{5r}(r-1)(3+8r)-e^{2r}\left(6+r+20r^2\right)\\ +8e^{3r}(3+4r(1+r))-3e^{4r}(2+r(5+4r(2+r)))+e^{6r}(6+r(13+4r(3r-5)))\end{matrix}\right\}u_{i\pm1}^n \\
& + \frac{e^{-6r}}{16r^3}\left(e^r-1\right)^2\left(-2+4e^{2r}-2e^{4r}-r+2e^r r+4e^{2r}r-2e^{3r}r-3e^{4r}r+6e^{2r}r^2+4e^{3r}r^2+6e^{4r}r^2\right)u_{i\pm2}^n \\
& + \frac{e^{-6r}}{32r^3}\left(e^{2r}-1\right)\left(-2+4e^r-4e^{3r}+2e^{4r}-r+4e^r r-6e^{2r}r+4e^{3r}r-e^{4r}r+4e^{2r}r^2-4e^{4r}r^2+4e^{4r}r^3\right)u_{i\pm3}^n
\end{aligned}
\tag{2.17}
$$

2.3 Extension to a General Grid

It is possible to extend the methods to more general cases where the geometrical and material properties of the simulated system depend on the position, and the discretization reflects this fact. To take a step towards a resistance–capacitance-type model [48] of heat conduction, we switch to cell variables, where u_i, c_i, ρ_i, and q_i are the average temperature, specific heat, density and heat source intensity of cell i, respectively, which will be approximated by their values at the cell centres. The quantity $k_{i,\,i+1}$ is the heat conductivity between cell i and its right neighbour, which can be approximated as the value of k at the cell borders. The heat capacity of the cells are $C_i = c_i \rho_i V_i$, while the thermal resistance between the cells can be estimated as $R_{i,i+1} \approx d_{i,i+1}/\left(k_{i,i+1}A_{i,i+1}\right)$, where $A_{i,i+1}$ is the interface area between the neighbouring cells. Using these quantities and approximations, one can obtain the ODE system for the time derivative of each cell-variable for a general (e.g. unstructured) grid in any space-dimensions, independently of any coordinate-system as follows:

$$
\frac{du_i}{dt} = \sum_{j\neq i} \frac{u_j - u_i}{R_{i,\,j}C_i} + q_i . \tag{2.18}
$$

Equation system (2.18) can be written into the same matrix-form as (2.2). The off-diagonal elements of the matrix M are $m_{ij} = 1/\left(R_{ij}C_i\right)$. If the cells i and j are not neighbours, i.e. there is no direct heat conduction between them, then the corresponding element is zero. The diagonal elements of the matrix are the negative sums of the off-diagonal elements: $m_{ii} = -\sum_{j\neq i}\left(R_{i,\,j}C_i\right)^{-1}$. Let us introduce two notations, namely $r_i = -hm_{ii}$ and $\tau_i = -\frac{1}{m_{ii}} = \frac{h}{r_i}$. The first one can be perceived as the generalization of r, while τ_i is the characteristic time or time constant of cell i. For the simplest one-dimensional case, we obtain $r_i = h\frac{2\alpha}{\Delta x^2} = 2r$ and $\tau_i = \frac{h}{2r}$ $(1 < i < N)$.

Algorithm 3b: generalized three-stage CLQ method

Stage 1 Calculate $a_i = \sum_{j\neq i} \frac{u_j^n}{R_{i,j}C_i} + q_i$, then take a full time step with the general form of the CNe method:

$$u_i^{\mathrm{C}} = u_i^n e^{-r_i} + \tau_i a_i \left(1 - e^{-r_i}\right)$$

Stage 2 Calculate $a_i^{\mathrm{C}} = \sum_{j\neq i} \frac{u_j^{\mathrm{C}}}{R_{i,j}C_i} + q_i$, then calculate a half and a full time step with the LNe method:

$$u_i^{\mathrm{L}1/2} = u_i^n e^{-r_i/2} + \tau_i a_i \left(1 - e^{-r_i/2}\right) + \tau_i \left(a_i^{\mathrm{C}} - a_i\right)\left[\frac{1}{2} - \frac{\tau_i}{h}\left(1 - e^{-r_i/2}\right)\right]$$

$$u_i^{\mathrm{L}} = u_i^n e^{-r_i} + \tau_i a_i \left(1 - e^{-r_i}\right) + \tau_i \left(a_i^{\mathrm{C}} - a_i\right)\left[1 - \frac{\tau_i}{h}\left(1 - e^{-r_i}\right)\right]$$

Stage 3 Calculate
$f_{1,i} = \sum_{j\neq i} \frac{u_j^{\mathrm{L}1/2}}{R_{i,j}C_i} + q_i$ and $f_{2,i} = \sum_{j\neq i} \frac{u_j^{\mathrm{L}}}{R_{i,j}C_i} + q_i$, then $s_i = \frac{4f_{1,i} - f_{2,i} - 3a_i}{h}$, and $w_i = 2\frac{f_{2,i} - 2f_{1,i} + a_i}{h^2}$.

After this take a full (and if necessary, a half) time step as follows

$$u_i^{\mathrm{Q}} = e^{-r_i} u_i^n + \left(1 - e^{-r_i}\right)\left(2w_i\tau_i^3 - s_i\tau_i^2 + a_i\tau_i\right) + \tau_i h(w_i h - 2w_i\tau_i + s_i)$$

$$u_i^{\mathrm{Q}1/2} = e^{-r_i/2} u_i^n + \left(1 - e^{-r_i/2}\right)\left(2w_i\tau_i^3 - s_i\tau_i^2 + a_i\tau_i\right) + \tau_i \frac{h}{2}\left(w_i \frac{h}{2} - 2w_i\tau_i + s_i\right)$$

Further stages can be constructed straightforwardly by recalculating $f_{1,i} = \sum_{\mathrm{j}\neq \mathrm{i}} \frac{u_{\mathrm{j}}^{\mathrm{Q}1/2}}{R_{\mathrm{i,j}}C_{\mathrm{i}}} + q_i$ and $f_{2,i} = \sum_{\mathrm{j}\neq \mathrm{i}} \frac{u_{\mathrm{j}}^{\mathrm{Q}}}{R_{\mathrm{i,j}}C_{\mathrm{i}}} + q_i$, then using these for the recalculation of s_i, w_i, $u_i^{Q1/2}$, and u_i^Q according to the formulas in Stage 3. (At the last stage calculating $u_i^{Q1/2}$ is redundant.) The generalized CNe and LNe methods are presented in [44] and also can be extracted from the generalized version of the CLQ method. These generalized algorithms can still be implemented with little coding effort, and once the quantities C_i and R_{ij} are given, the methods are easy to use.

Remark 2.1 Note that in Algorithm 3b, the C and R quantities are not directly present, only through the matrix elements. Thus, if one defines $a_i = \sum_{j\neq i} m_{i,j} u_j^n + q_i$, etc., then the algorithms can be applied for any ODE system which has the form (2.2), independently of the physical content of the variables. Since exponential terms are present in the new algorithms, they may look similar to the so-called exponential integrators [49]. However, matrix exponentials are calculated in the case of those methods, while all of our algorithms are fully *explicit*: solving equation systems or performing operations on matrices is not needed. The necessary operations are clearly proportional to the first power of the number of spatial nodes as well as the total number of timesteps, thus the algorithms scale like $O(N \cdot T)$. This is confirmed by all of our numerical experiments where we measured the running time for sufficiently long runs. Moreover, the calculations are suitable for parallelization on GPUs as well.

2.4 Unconditional Positivity and Stability of the CLQn Method

First, we examine the one-dimensional linear diffusion equation with an equidistant mesh. We recall the following simple lemma, the associativity of convex combinations [43, p. 28].

Lemma A convex combination $x = \sum a_i x_i$ of convex combinations $x_i = \sum b_{ij} y_{ij}$ is again a convex combination:

$$x = \sum \sum \left(a_i b_{ij}\right) y_{ij}$$

for any $y_{ij} \in \mathbb{R}^n$.

Theorem 2.1 In case of the linear diffusion Eq. (1.2), if the q_i source terms are zero, the new values u_i^{C}, u_i^{L}, u_i^{CLQ}, $u_i^{\mathrm{CLQQ}} u_i^{\mathrm{CLQ3}}$ and u_i^{CLQ4} are the convex combinations of the initial values $u_j(0), \quad j = 1, \ldots, N$.

Proof Using the Mathematica software, we have analytically calculated the coefficients of the old values u_j^n in the expression of the new value of a general element u_i^{n+1} (such as Eq. (2.17)) as functions of the mesh ratio $r > 0$, and examined their properties. The sum of the coefficients is always exactly one. We have to prove that each of the coefficients is nonnegative. In the case of the CNe and LNe methods, we have 3 and 5 nonzero coefficients (first and second neighbours), respectively. These had been examined in our previous papers [44, 47] by hand calculations, but we repeated them by the software for verification. In the case of the CLQn method, we have $2(n + 2) + 1$ nonzero coefficients, which are more and more complicated as n increases, but still continuous functions of the variable r. First, the limit values are calculated at 0 and ∞. Then, the functions are differentiated with respect to r to obtain the extrema (extreme points). In some cases, there are one or two extreme points, but the extreme values are always in the unit interval. The values are tabulated in Table 1, where the last column gives the second derivative of the coefficient with respect to r at the extrema to let the readers know if the extreme value is minimum or maximum. In Fig. 1 we exemplify the coefficients of the $u_{i\pm j}^n$ terms as a function of r for the CLQ4 method, where i is a general mesh point while $j \in \{0, \ldots, 6\}$. The last two coefficients are quite small, and their curves are hard to distinguish with the naked eye.

From the analysis above, it is clear that in the open interval $r \in]0, \infty[$, all coefficients are strictly positive. The Lemma now implies the theorem. □

Remark 2.2 In our previous paper [44], we analytically proved the convex combination property for the CNe and LNe methods in the case of an arbitrary mesh and space dimensions. For the CLQn method, however, we managed to do this only for the simplest case of Eq. (1.2) with an equidistant mesh, but all of the numerical experiments suggest that it holds in more general cases as well. Theorem 2.1 about convex combination property states that (for heat conduction) the temperature of each cell tends to the weighted average of the temperatures of the surrounding cells. It implies that errors cannot be amplified during the calculation. The solution obeys the maximum and minimum principles [51, p. 87], i.e. the extremal values of the function u occur among the initial or the (prescribed) boundary values. It also implies that starting from nonnegative initial values and assuming no heat sinks are present, the function u always remains positive. This convex-combination property is a much stronger property than unconditional stability.

In case of linear or linearized ODE systems of $u' = Mu$ type, the widely used definition of A-stability is that the numerical solution is bounded for arbitrarily large step sizes for

Table 1 Analysis of the coefficient-functions

Method	Term	$\lim_{r\to 0}$	$\lim_{r\to\infty}$	Extreme point(s)	Extreme value(s)	2nd der. value
CLQ	$u^n_{i\pm 3}$	0	1/8	–	–	–
	$u^n_{i\pm 2}$	0	0	1.7474	0.1079	– 0.03664
	$u^n_{i\pm 1}$	0	3/8	0.8312 and 1.7355	0.2211 and 0.2123	– 0.1259 and 0.03215
	u^n_i	1	0	–	–	–
CLQ2	$u^n_{i\pm 4}$	0	1/16	n.a	n.a	n.a
	$u^n_{i\pm 3}$	0	0	2.7532	0.0561	– 0.009631
	$u^n_{i\pm 2}$	0	1/4	–	–	–
	$u^n_{i\pm 1}$	0	0	0.7712	0.2189	– 0.1997
	u^n_i	1	3/8	2.5863	0.2313	0.02488
CLQ3	$u^n_{i\pm 5}$	0	1/32	–	–	–
	$u^n_{i\pm 4}$	0	0	3.6621	0.028377	– 0.002978
	$u^n_{i\pm 3}$	0	5/32	–	–	–
	$u^n_{i\pm 2}$	0	0	2.40565	0.134179	– 0.020307
	$u^n_{i\pm 1}$	0	5/16	0.7218 and 2.9838	0.2161 and 0.1496	– 0.2695 and 0.02029
	u^n_i	1	0	–	–	–
CLQ4	$u^n_{i\pm 6}$	0	1/64	–	–	–
	$u^n_{i\pm 5}$	0	0	4.54791	0.0142048	– 0.00097952
	$u^n_{i\pm 4}$	0	3/32	–	–	–
	$u^n_{i\pm 3}$	0	0	3.59707	0.0806553	– 0.0076002
	$u^n_{i\pm 2}$	0	15/64	–	–	–
	$u^n_{i\pm 1}$	0	0	0.745453	0.217207	– 0.2263
	u^n_i	1	5/16	4.00917	0.176982	0.0103322

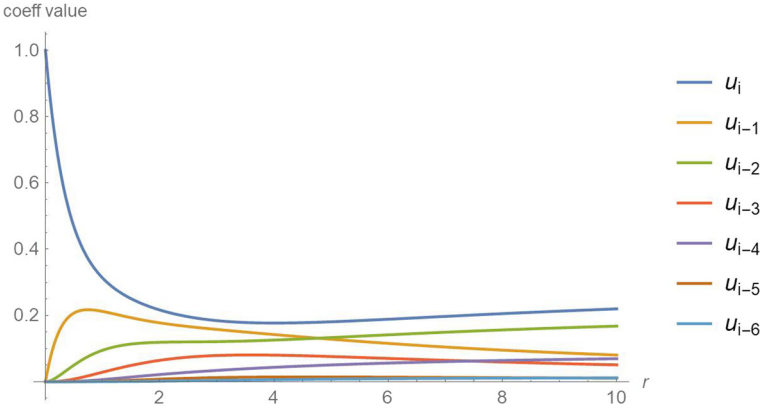

Fig. 1 The coefficients of the $u^n_{i\pm j}$ terms for the CLQ4 method as a function of the mesh ratio

Dahlquist's famous test equation, i.e. for each (non-positive) eigenvalue of M. Since the work of Dahlquist we know that no explicit Runge–Kutta or multistep Adams–Bashforth method can be A-stable (see [52] and the references therein), due to the polynomial expression of the time step size h. Now, unlike in the case of those methods, in our cases the step size h appears not in a polynomial, but in an exponential form with negative exponents (for the linear heat equation) in the expression for the new values of the variable u. This means that the proposed formulas contain h up to infinite order, which gives the entirely different stability properties. However, this will have some consequences on the truncation errors, as we will see soon.

Remark 2.3 In the case of the CLQ5 method, the coefficient of $u_{i\pm7}^n$ can be negative with very small (less than 10^{-7}) absolute value. However, this is already enough for not only the violation of the positivity preserving property, but for occasional unstable behaviour as well. That is why we propose and examine the methods only up to 4 quadratic stages. We have also tried to omit the linear stage and examined the obtained CQ, CQQ, etc. methods. These are typically slightly more accurate than their counterpart with the same number of stages. The problem is that they are only conditionally stable, albeit with much larger mesh Fourier numbers than for the RK methods. Still, we stopped their investigation, because we think that losing the unconditional stability is too big a price for a slight increase in accuracy.

2.5 Consistency and Convergence

First, the consistency and convergence will be discussed considering the schemes as direct PDE solvers as usual [53] (Sect. 4.2), [54]. Subscripts containing x or t denote differentiation with respect to the space or time variables, respectively. We give the truncation errors for the proposed methods below.

Theorem 2.2 When applied to the homogeneous PDE $u_t = \alpha u_{xx}$, the CLQn methods (Algorithms 6 and 7) are conditionally consistent, i.e. if the time and space step sizes tend to zero $h \to 0$, $\Delta x \to 0$, such that $\frac{h}{\Delta x^2} \to 0$, then the numerical solution tends to the analytical solution of the PDE.

Proof We calculate the local truncation error for each method by symbolic computer calculation. Assuming that the analytical solution u of Eq. (1.2) is sufficiently smooth, one can use the following temporal and spatial Taylor-series expansions of this u:

$$u_i^{n+1} = u_i^n + \sum_s \frac{h^s}{s!} u_{(st)}, \text{ and } u_{i\pm j}^n = 2\left(u_i^n + \sum_s \frac{j^{2s}\Delta x^{2s}}{(2s)!} u_{(2sx)}\right) \text{for} j,\ s \in \mathbb{N}^+.$$

Using the Taylor series expansion of the exponential function, we also expand the r-dependent coefficients, such as e^{-2r}. In the case of the CLQ method, we are going up at least to $u_{(4t)}$, $u_{(8x)}$, and r^4, and even further if there are more stages. We substitute these into the final expressions of the algorithms, for example, Eq. (2.17), in the case of the CLQ scheme. Since u is the true solution, we use the substitutions $u_t = \alpha u_{xx}$, $u_{tt} = (\alpha u_{xx})_t = \alpha (u_t)_{xx} = \alpha^2 u_{(4x)}$, $u_{ttt} = \alpha^3 u_{(6x)}$ and so on. After simplifications, we obtain the local truncation errors of the methods. Now the global error can be estimated by dividing the local truncation error by h. For the sake of brevity, we introduce the notations

$$\varepsilon_0 = -\frac{\alpha\Delta x^2\big(90\big(1680u_{(4x)} + 56\Delta x^2 u_{(6x)} + \Delta x^4 u_{(8x)}\big) + \Delta x^6 u_{(10x)}\big)}{1814400},$$

$$\varepsilon_1 = -\frac{\alpha^2\Delta x^2\left(5040u_{(6x)} + 378\Delta x^2 u_{(8x)} + 17\Delta x^4 u_{(10x)}\right)}{60480}h,$$
$$\varepsilon_2 = -\frac{\alpha^3\Delta x^2\left(60u_{(8x)} + 7\Delta x^2 u_{(10x)}\right)}{1440}h^2$$

where ε_0 belongs to the central difference formula (2.1). The global errors without some higher order terms are as follows:

$$\begin{aligned}
\varepsilon_{\mathrm{CLQ}} &= \varepsilon_0 + \varepsilon_1 + \varepsilon_2 \\
&+ \frac{\alpha^4\left(720\left(315u_{(2x)} + 420\Delta x^2 u_{(4x)} + 224\Delta x^4 u_{(6x)} + 64\Delta x^6 u_{(8x)}\right) + 3467\Delta x^8 u_{(10x)}\right)}{340200\Delta x^6}h^3 \\
\varepsilon_{\mathrm{CLQQ}} &= \varepsilon_0 + \varepsilon_1 + \varepsilon_2 - \frac{1}{72}\alpha^4\Delta x^2 u_{(10x)}h^3 + \\
&+ \frac{\alpha^5\left(80\left(5040u_{(2x)} + 8820\Delta x^2 u_{(4x)} + 6475\Delta x^4 u_{(6x)} + 2683\Delta x^6 u_{(8x)}\right) + 57267\Delta x^8 u_{(10x)}\right)}{1209600\Delta x^8}h^4 \\
\varepsilon_{\mathrm{CLQ3}} &= \varepsilon_0 + \varepsilon_1 + \varepsilon_2 - \frac{1}{72}\alpha^4\Delta x^2 u_{(10x)}h^3 + \frac{\alpha^5\left(80u_{(6x)} + 220\Delta x^2 u_{(8x)} + 149\Delta x^4 u_{(10x)}\right)}{57600\Delta x^4}h^4 \\
&+ \frac{\alpha^6}{2155507200\Delta x^{10}} \\
&\cdot \left(132\left(90\left(56\left(360u_{(2x)} + 750\Delta x^2 u_{(4x)} + 661\Delta x^4 u_{(6x)}\right) + 18397\Delta x^6 u_{(8x)}\right) + 561481\Delta x^8 u_{(10x)}\right)\right. \\
&\left. + 19595929\Delta x^{10} u_{(12x)}\right)h^5 \\
\varepsilon_{\mathrm{CLQ4}} &= \varepsilon_0 + \varepsilon_1 + \varepsilon_2 - \frac{1}{72}\alpha^4\Delta x^2 u_{(10x)}h^3 + \frac{\alpha^5\left(80u_{(6x)} + 220\Delta x^2 u_{(8x)} + 149\Delta x^4 u_{(10x)}\right)}{57600\Delta x^4}h^4 \\
&+ \frac{\alpha^6\left(180u_{(8x)} + 510\Delta x^2 u_{(10x)} + 377\Delta x^4 u_{(12x)}\right)}{129600\Delta x^4}h^5
\end{aligned}$$

The global errors have the following orders: $\varepsilon_{CLQ} = O\left(\Delta x^2,\ \Delta x^2 h,\ \Delta x^2 h^2,\ h^3,\ \frac{h^3}{\Delta x^6}\right)$, $\varepsilon_{\mathrm{CLQQ}} = O\left(\Delta x^2,\ \Delta x^2 h,\ \Delta x^2 h^2,\ \Delta x^2 h^3,\ h^4,\ \frac{h^4}{\Delta x^8}\right)$ and $\varepsilon_{\mathrm{CLQn}} = O\left(\Delta x^2,\ \Delta x^2 h,\ \Delta x^2 h^2,\ \Delta x^2 h^3,\ h^4,\ \frac{h^4}{\Delta x^4}\right)$, $n = 3,\ 4$. One can see that there are several mixed terms containing both the space and the time step size, which are not present in the case of traditional methods such as the Runge Kutta schemes. The most problematic is that the global error contains powers of $\frac{h}{\Delta x^2}$ and $\frac{h}{\Delta x}$ in all cases, which means that the consistency is only conditional. This fact and its negative consequences will be explained in the next remark. □

Remark 2.4 As we showed, these methods are only conditionally consistent, and therefore conditionally convergent, if one considers them as time integrators for the original PDE. The first consequence of the $\frac{h}{\Delta x^2}$ terms in the global error is that when one has to solve the PDE with higher accuracy, it is not enough to make the size of the spatial cells Δx tend to zero, because if h is kept constant, the global error actually does not decrease (but stability remains, unlike in the case of the conditionally stable explicit methods). The numerical solution converges to the true solution of the original PDE if *both* the space step size Δx and the time step size h go to zero, but h goes faster than Δx^2. If this condition is not fulfilled, the errors tend to non-negligible constant values due to the convex combination property. The second consequence is the slow convergence for larger time step sizes. As we will see, when we start to decrease the time step size, the errors are first decreasing very slowly, and they approach the theoretical order only for medium step sizes. It means that for larger time steps, the order of temporal convergence cannot be determined using the formula $\varepsilon \sim h^p$. We note that this conditional consistency is the price one has to pay for unconditional stability. It is

common among the stable explicit methods, for example in the case of the first order UPFD method. This is the main weak point of these kinds of algorithms. Nevertheless, we are going to show that they are very competitive in several cases.

As we already mentioned in Remark 2.2, the new CLQn methods (and the methods given in Sect. 2.1) can be used to solve general ODE systems. Now the order of accuracy is examined from this point of view.

Theorem 2.3 The order of convergence is three for the three-stage CLQ algorithm and four for the 4, 5 and 6-stage CLQn algorithms, $n \in \{2,\ 3,\ 4\}$, if they are applied to the general.

$$\frac{d\vec{u}}{dt} = M\vec{u} + \vec{Q}, \quad \vec{u}(\mathrm{t} = 0) = \vec{u}^0.$$

linear ODE system, where $\vec{u} : \{1, ..., N\} \times \mathbb{R} \mapsto \mathbb{R}^N$ is the unknown vector-function, $M \in \mathbb{R}^{N\times N}$ is an arbitrary constant matrix, while $\vec{Q},\ \vec{u}^0 \in \mathbb{R}^N$ are arbitrary vectors.

Proof We applied the Mathematica software for symbolic computer-calculation. The linear initial value problem

$$\frac{d}{dt}\begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} m_{11} & \cdots & m_{1N} \\ \vdots & \vdots & \vdots \\ m_{N1} & \cdots & m_{NN} \end{pmatrix}\begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix} + \begin{pmatrix} Q_1 \\ \vdots \\ Q_N \end{pmatrix}, \quad \vec{u}(t=0) = \begin{pmatrix} u_1^0 \\ \vdots \\ u_N^0 \end{pmatrix}$$

has been examined, where the general elements m_{ij} are kept undetermined. It is well known that this ODE system has the following analytical solution:

$$\vec{u}(t) = e^{Mt}\vec{u}^0 + \left(e^{Mt} - 1\right)M^{-1}\vec{Q}.$$

Substituting h instead of t, i.e. considering this function at the end of the time step, the exact solution up to fourth order in h can be written as

$$\vec{u}^{n+1} = \left(1 + Mh + M^2\frac{h^2}{2} + M^3\frac{h^3}{3!} + M^4\frac{h^4}{4!} + \cdots\right)\vec{u}^n + \left(h + M\frac{h^2}{2} + M^2\frac{h^3}{3!} + M^3\frac{h^4}{4!}\cdots\right)\vec{Q}.$$

The software calculated this solution as well as the numerical solution by symbolic matrix operations. After this, the coefficients of h from h^0 up to h^4 have been sorted out for both the analytical and the numerical solution of the ODE system. It is observed that the appropriate coefficients are exactly the same in the exact solution and in the CLQn results. It follows that the local error of the time integrator CLQn is fourth order in h if $n = 1$ and fifth order if $n = 2$, 3 or 4, which is equivalent to the statement of the theorem about global errors. □

Corollary 2.5 If we apply the CLQn methods as time integrators for the ODE system (2.18) obtained from the PDE (1.1) by spatial discretization using *any* fixed spatial grid, the order of the convergence of the method is the same as we stated for the equidistant grid. It means that these new methods can be used in any space dimensions, in case of unstructured grids or inhomogeneous medium. We underline again that some other stable and explicit methods, such as the odd–even hopscotch and ADE methods, lose their favourable properties when they are applied to these more general meshes.

3 The Adaptation of the Proposed Methods for the Nonlinear Cases

3.1 Fisher's equation

We apply a kind of operator splitting, which means that first we handle the effect of the diffusion term and then separately the effect of the nonlinear reaction term in Eq. (1.3). The diffusion term is treated exactly as in Sect. 2. For example, first Algorithm 4 is applied, which yields u_i^{pred}, the value in which the diffusion term is fully taken into account. These u_i^{pred} values are now used as starting points for the calculation of the reaction term, which has the form $\beta u(1-u) = \beta u - \beta uu$. The negative term can affect the positivity condition, thus we treat the terms in a "semi-implicit" or "pseudo-implicit" way as follows. We substitute the first power of the new, unknown value u_i^{n+1} into the expression of the nonlinear term, thus this new value appears on the right-hand side of the equation as well. Everywhere else u_i^{pred} remains, thus we have

$$u_i^{n+1} = u_i^{\text{pred}} + \beta u_i^{\text{pred}} h - \beta u_i^{\text{pred}} u_i^{n+1} h.$$

This can be straightforwardly rearranged into a fully explicit form:

$$u_{\text{i}}^{\text{n}+1} = \frac{1+\beta h}{1+\beta h u_{\text{i}}^{\text{pred}}} u_{\text{i}}^{\text{pred}}. \tag{3.1}$$

The operation (3.1) can be performed at the last stage (inside the "for" loop going through the nodes) immediately after the new values u_i^{CLQn} (u_i^{C} and u_i^{CL} in the case of the CNe and LNe method, respectively) are calculated, because these serve as u_i^{pred}. We note that a similar treatment has been applied by Appadu et al. [55] to the generalized Burgers-Huxley equation in the framework of the NSFD methods, but they constructed one-stage first order methods.

Another possibility of treating the Fisher-term arises if we solve the

$$\frac{du}{dt} = \beta u(1-u) \tag{3.2}$$

ODE analytically. More precisely, we solve the corresponding initial value problem, where the values u_i^{pred} serve as initial values at the initial time, which is the beginning of the actual time step. This yields:

$$u_i^{n+1} = \frac{1}{u_i^{\text{pred}} + \left(1 - u_i^{\text{pred}}\right) e^{-\beta h}} u_i^{\text{pred}}. \tag{3.3}$$

Apart from the operator splitting itself, this solution avoids any approximation or linearization.

Remark 3.1 Assume that $0 \le u_i^{\text{pred}} \le 1$. Now if one considers the coefficient of u_i^{pred}, one can easily see that the numerators and the denominators in both (3.1) and (3.3) are positive and the denominators cannot be larger than the numerators. Moreover, the right-hand sides cannot be larger than one. From these, it immediately follows that the new value u_i^{n+1} cannot be smaller than the u_i^{pred} value, and it cannot be negative, thus $0 \le u_i^{\text{pred}} \le u_i^{n+1} \le 1$ holds.

Corollary 3.1 (dynamical consistency of the methods). If we use Algorithm 1–4 with Eqs. (3.1) or (3.3) to solve Fisher's Eq. (1.3) and if the initial values are in the unit interval $u_i^0 \in [0,\ 1]$, then the values of u remain in this interval for arbitrary nonnegative β and time step size h during the whole calculation. This is similar to the maximum and minimum

principles, and automatically implies unconditional stability. Moreover, the appearance of the Fisher term (increasing β from zero to a positive value) cannot imply the decrease, only the increase of the u values, which also reflect a dynamical property of the original PDE (1.3).

Proof Theorem 2.1 and Remark 3.1 immediately imply the statement.

3.2 Nagumo's Equation

Now the reaction term in Eq. (1.4) has the form $\beta u\left(1-u^{\delta}\right)\left(u^{\delta}-\gamma\right)=\beta u^{\delta+1}-\beta\gamma u-\beta u u^{2\delta}+\beta\gamma u^{\delta+1}$. The ODE containing the nonlinear Nagumo term cannot be solved analytically in general, so we follow the logic which leads us to Formula (3.1) above. This implies

$$u_i^{n+1}=u_i^{\text{pred}}+\beta h\left(u_i^{\text{pred}}\right)^{\delta+1}-\beta h\gamma u_i^{n+1}-\beta h\left(u_i^{\text{pred}}\right)^{2\delta}u_i^{n+1}+\beta h\gamma\left(u_i^{\text{pred}}\right)^{\delta+1},$$

The explicit form is easily obtained:

$$u_i^{n+1}=\frac{1+\beta h(1+\gamma)\left(u_i^{\text{pred}}\right)^{\delta}}{1+\beta h\left(\gamma+\left(u_i^{\text{pred}}\right)^{2\delta}\right)}u_i^{\text{pred}}. \tag{3.4}$$

Theorem 3.1 (dynamical consistency). If Algorithm 6 or 7 and then formula (3.4) is applied to solve Eq. (1.4), then the values of the solution lie in the unit interval [0, 1] provided that $0\le u_i^0\le 1,\ 0\le\gamma\le 1,\ \delta>0,\ \beta>0$ for arbitrary time step size h if.

$$\delta\le\frac{(1+\beta h)^2+\beta h\gamma(2+2\beta h+\beta h\gamma)}{\beta h\left[(1+\beta h)-\gamma(1+\beta h\gamma)\right]}. \tag{3.5}$$

Since $1+\beta h>\gamma(1+\beta h\gamma)$, u_i^{n+1} is obviously nonnegative, thus we have to prove that it cannot be larger than 1. Let us introduce the temporary notations $u=u_i^{n+1}$ $p=u_i^{\text{pred}}$ and $b=\beta h$ for brevity. We prove that

$$u=u(p)=\frac{p+b(1+\gamma)p^{\delta+1}}{1+b(p^{2\delta}+\gamma)}\le 1$$

provided that $0\le p\le 1,\quad 0\le\gamma\le 1,\quad \delta>0,\quad b>0.$

The examined $u(p)$ function is continuous and sufficiently smooth in $p\in[0,\ 1]$ and at the boundaries of the examined interval its values are $u(p=0)=0$ and $u(p=1)=1$. Due to the extreme value theorem, if u has no extremal value in the interior of the unit interval, then it cannot be larger than 1. Let us differentiate the $u(p)$ function:

$$\frac{du}{dp}=\frac{1+b\gamma+b(1-2\delta)p^{2\delta}+b^2(1+\gamma)(1-\delta)p^{3\delta}+b(1+\gamma)(1+b\gamma)(1+\delta)p^{\delta}}{\left(1+bp^{2\delta}+b\gamma\right)^2}. \tag{3.6}$$

The denominator of (3.6) is always positive, thus to seek for extremal values, we need to examine only the numerator. The value of the numerator is nonnegative for $p=0$. The coefficients of the two largest power of p, namely $p^{2\delta}$ and $p^{3\delta}$ can be negative, which can cause a non-positive numerator. Note that since $p\in[0,\ 1]$ and the exponents are positive, larger powers of p are smaller and increasing with p. This implies that if the numerator is negative for a particular $p^*\in]0,\ 1[$, then it is also negative for $p=1$. Thus, if the numerator

of (3.6) is positive for $p = 1$, then the $u(p)$ function is monotonously increasing from 0 to 1 and the statement of the theorem is true. The numerator for $p = 1$ can be written as

$$\begin{aligned} &1 + b\gamma + b(1 - 2\delta) + b^2(1 + \gamma)(1 - \delta) + b(1 + \gamma)(1 + b\gamma)(1 + \delta) \\ &= (1 + b)^2 + b\gamma(2 + 2b + b\gamma) + b(\gamma(1 + b\gamma) - (1 + b))\delta \end{aligned} \tag{3.7}$$

and the requirement of the non-negativity of (3.7) gives the condition (3.5). □

Remark 3.2 If condition (3.5) is reorganized into the form

$$1 + (2 - \delta)b + (1 - \delta)b^2 + (2 + \delta)b\gamma + 2b^2\gamma + (1 + \delta)b^2\gamma^2 \geq 0$$

Then it is easy to see that if $\delta \leq 1$ then (3.5) holds for arbitrary non-negative $b = \beta h$, and γ. Moreover, by examining the right-hand side of (3.5) one can obtain that it approaches the value 1 only if b tends to infinity and γ tend to zero. For example, for $\gamma \to 0$ it takes $(1 + b)/b$, so the maximum value of δ is at least two if $h \leq 1/\beta$. On the other hand, if b tends to zero and/or γ tends to one, it strictly monotonously tends to infinity. The values of the maximum allowed δ are plotted in Fig. 2.

Remark 3.3 In the case of Huxley's equation, the $\gamma = 0$, $\delta = 1$ substitution into (3.5) yields the

$$1 \leq \frac{(1 + \beta h)^2}{\beta h(1 + \beta h)} = \frac{1 + 2\beta h + (\beta h)^2}{\beta h + (\beta h)^2}$$

condition, which trivially holds for any nonnegative β for arbitrary time step size, thus the statement of the theorem is automatically true in this case.

We note that the investigation of the consistency of the methods in the case of nonlinear equations is out of the scope of this paper, and is planned to be published in the future.

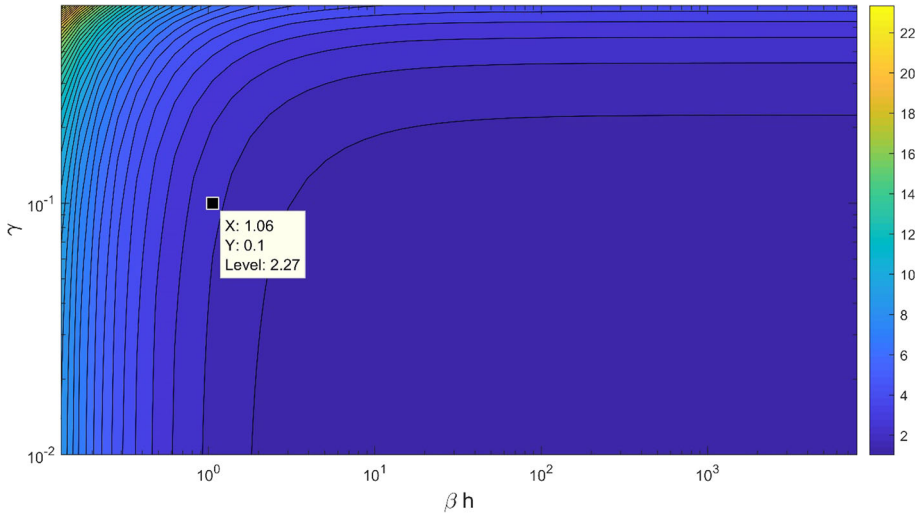

Fig. 2 The maximum allowed values of the parameter δ as a function of βh and γ for Eq. (1.4) under which dynamical consistency is guaranteed

4 Numerical Experiments in 1D Using Analytical Reference Solutions

The numerical solution is compared with the reference solution only at final time t_{fin} which 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:

$$\text{Error}(L_\infty) = \max_{1\le \text{j}\le N} \left| u_{\text{j}}^{\text{ref}}(t_{\text{fin}}) - u_{\text{j}}^{\text{num}}(t_{\text{fin}}) \right|. \tag{4.1}$$

The numerical order of convergence can be calculated using two values of the errors belonging to two subsequent time step sizes h_1 and h_2, where $h_1 > h_2$, using the formula

$$Error \sim h^p \quad \Rightarrow \quad p = \frac{\log \frac{Error(h_1)}{Error(h_2)}}{\log(h_1/h_2)}. \tag{4.2}$$

4.1 Experiment 1: Nonlinear Fisher's Equation

We are going to reproduce the following analytical solution [39, 56] of PDE (1.3), valid for $\alpha = 1$:

$$u^{\text{exact}}(x,\, t) = \left(1 + e^{\sqrt{\frac{\beta}{6}}x - \frac{5}{6}\beta t}\right)^{-2}.$$

The initial condition is obtained simply by substituting the initial time into that:

$$u(x,\, t=0) = \left(1 + e^{\sqrt{\frac{\beta}{6}}x - \frac{5}{6}\beta t^0}\right)^{-2}.$$

The appropriate Dirichlet boundary conditions are prescribed at the two ends of the interval:

$$u(x = x_0,\, t) = \left(1 + e^{\sqrt{\frac{\beta}{6}}x_0 - \frac{5}{6}\beta t}\right)^{-2},$$

$$u(x = x_N,\, t) = \left(1 + e^{\sqrt{\frac{\beta}{6}}x_N - \frac{5}{6}\beta t}\right)^{-2}.$$

We examine the numerical solution at the interval $x \in [0,\ 2]$, thus $x_0 = 0\,,\quad x_N = 2$, which is discretized by dividing it into 500 equal parts: $x_j = x_0 + j\Delta x\,,\quad j = 0,\, ...,\, 500\,,$ $\Delta x = 0.004$. The initial and the final time are $t^0 = 0$ and $t^{\text{fin}} = 0.1$. The nonlinear coefficient is quite large, $\beta = 22$.

For comparison purposes, we coded the NSFD scheme mentioned in the paper of Agbavon et al. [39], which has the formula for the $\alpha = 1$ case:

$$\text{NSFD}: u_i^{n+1} = \frac{(1 - h\beta - 2r)u_i^n + r u_{i\pm1}^n}{1 + h\beta\left(u_{i\pm1}^n + u_{i-1}^n\right)/3},$$

First, for verification of this NSFD part of our code, we reproduced some error-values from that publication (Table 6 in [39]). Then, in our experiment, we calculated the error as a function of the time step size h for all the examined methods. The results are presented in Fig. 3. We note that similar results have been obtained for other parameter values of β, t^{fin}, etc. as well. The proposed CLQ, CLQ2, etc. algorithms are smoothly converging without the

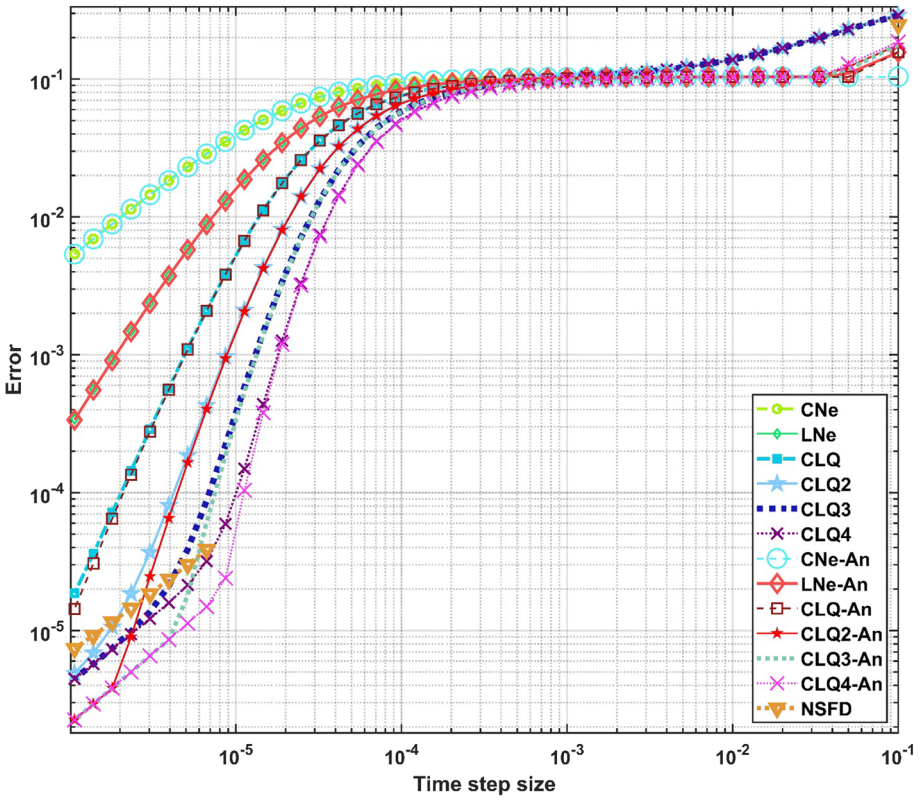

Fig. 3 L_∞ errors as a function of the time step size h for Experiment 1

slightest sign of instability. However, this convergence is very slow for larger time step sizes due to the inconsistent truncation error terms, as we explained in Remark 2.4. In the case of all members of the CNe-CLQ4 family, not only Eq. (3.1) but Eq. (3.3) is used to test, where the nonlinear term is treated analytically. For very large and very small time step sizes, this treatment (denoted by "An" in the Figure) can be more accurate, but for medium time step sizes, which are the most interesting for us, the accuracy is the same for the two treatments. The NSFD method yields an error around 10^8 for large and medium time step sizes, but below those, it is quite accurate.

In Table 2 we present the maximum of the numerical orders calculated by the (4.2) formula, with the appropriate larger and smaller time step sizes h_1 and h_2, and the belonging errors. The numerical orders are not very far from their theoretical values, even for this severely nonlinear case. In Fig. 4 we present the errors of the CLQ and CLQ4 schemes for different numbers of nodes N, and therefore, mesh spacing Δx as a function of the time step size. This can help one to optimize the spatial and the temporal discretization.

4.2 Experiment 2: Nonlinear Nagumo's Equation

Now PDE (1.4) with $\alpha = 1$ is going to be solved in the domain $(x,\ t) \in [x_0,\ x_N] \times \left[0,\ t^{\text{fin}}\right]$. We found the following analytical solution of Eq. (1.4) in the literature [12], valid in

Table 2 Largest numerical order and the appropriate time step sizes and L_∞ errors of different methods for Experiment 1

Method	Larger h_1 and error	Smaller h_2 and error	Numerical order
CNe	2.33×10^{-6} and 1.14×10^{-2}	1.79×10^{-6} and 8.90×10^{-3}	0.940
LNe	2.33×10^{-6} and 1.48×10^{-3}	1.79×10^{-6} and 9.16×10^{-4}	1.836
CLL	3.03×10^{-4} and 1.99×10^{-4}	2.33×10^{-6} and 9.90×10^{-5}	2.654
CLQ	2.33×10^{-6} and 1.44×10^{-4}	1.79×10^{-6} and 7.18×10^{-5}	2.659
CLQ2	6.66×10^{-6} and 4.31×10^{-4}	5.12×10^{-6} and 1.86×10^{-4}	3.199
CLQ3	1.13×10^{-5} and 5.80×10^{-4}	8.66×10^{-6} and 2.22×10^{-4}	3.664
CLQ4	1.46×10^{-5} and 4.37×10^{-4}	1.13×10^{-5} and 1.49×10^{-4}	4.105

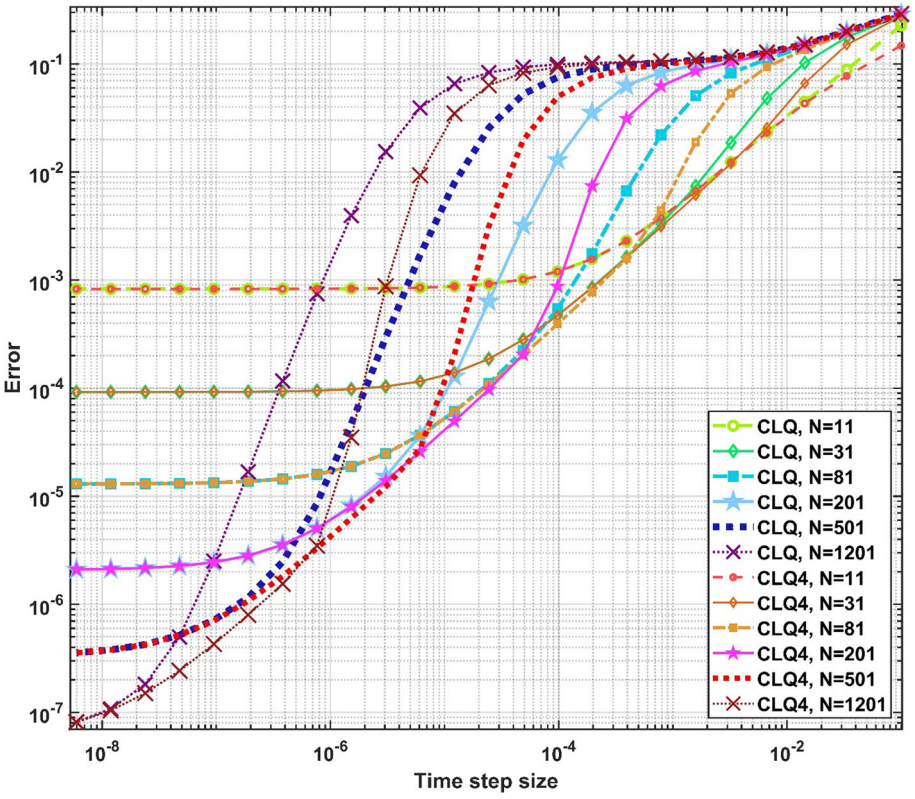

Fig. 4 Errors as a function of the time step size h for different mesh spacings in the case of Experiment 1

the $\delta = 1$ case:

$$u^{\text{anal}}(x, t) = \frac{1}{2}\left(1 - \tanh\left[\sqrt{\frac{\beta}{8}}(x - ct)\right]\right),$$

where $c = \sqrt{\frac{\beta}{2}}(1-2\gamma)$. We use parameter values $\alpha = 1,\ \beta = 10,\ \gamma = 0.4,\ \delta = 1$ in Eq. (1.4), and $x_0 = 0,\ x_N = 3,\ t^{\text{fin}} = 0.6,\ \Delta x = 0.02$. Similarly to the previous subsection, we obtain the initial and the Dirichlet boundary conditions by taking this function at the appropriate time and space points: $u(x,\ 0)$ and $u(x_0,\ t),\ u(x_N,\ t)$, respectively. In paper [12] four nonstandard finite difference schemes were proposed with the following formulas:

$$\text{NSFD1} : u_i^{n+1} = \frac{(1-2R)u_i^n + Ru_{i\pm1}^n + \beta\phi_2\left((1+\gamma)\left(u_{i-1}^n\right)^2 + \left(u_{i-1}^n\right)^3/2\right)}{1+\beta\gamma\phi_2 + {}^3\!/_2\beta\phi_2\left(u_{i-1}^n\right)^2}, \tag{4.3}$$

$$\text{NSFD2} : u_i^{n+1} = \frac{(1-2R)u_i^n + Ru_{i\pm1}^n}{1+\beta\gamma\phi_2 - \beta(1+\gamma)\phi_2 u_i^n + \beta\phi_2\left(u_i^n\right)^2}, \tag{4.4}$$

$$\text{NSFD3} : u_i^{n+1} = \frac{(1-2R)u_i^n + Ru_{i\pm1}^n + \beta\phi_2\left(u_i^n\right)^3 + (1+\gamma)\left(u_i^n\right)^2}{1+\beta\gamma\phi_2 + 2\beta\phi_2\left(u_i^n\right)^2}, \tag{4.5}$$

$$\text{NSFD4} : u_i^{n+1} = \frac{(1-2R)u_i^n + Ru_{i\pm1}^n + \beta\phi_2\left((1+\gamma)\left(u_i^n\right)^2 + \left(u_i^n\right)^3/2\right)}{1+\beta\gamma\phi_2 + {}^3\!/_2\beta\phi_2\left(u_i^n\right)^2}, \tag{4.6}$$

where $\phi_2 = \frac{e^{\beta\Delta t}-1}{\beta}$, $\psi_1 = \frac{1-e^{-\beta\Delta x}}{\beta}$, $\psi_2 = \frac{e^{\beta\Delta x}-1}{\beta}$ and $R = \frac{\phi_2}{\psi_1\psi_2}$. It is analytically proved that these methods have dynamical consistency, i.e. positivity and boundedness. However, these are not unconditional, and the most important condition is $R \le {}^1\!/_2$, which is actually a similar condition than the standard mesh Fourier number for the explicit Euler method. We use these NSFD schemes for comparison purposes.

We present here the set of the error curves in Fig. 5, while some numerical errors are shown in Table 3. One can see that the new methods behave well for this strongly nonlinear case as well. The error as a function of h decreases smoothly with h, there are no signs of instability or unphysical, e.g. oscillatory behaviour. In Fig. 6 the concentration u as a function of the space variable x is presented for the case of the analytical solution at the final time and two numerical solutions. The initial u_0 function is also plotted to see how the variable changes in time. One can see that even for this relatively large time step size, where the maximum error is 0.044, the numerical solution is qualitatively the same as the analytical one, and no unphysical oscillations appear. A further decrease of the time step size yields numerical curves which are indistinguishable from the analytical one.

4.3 Experiment 3: Nonlinear Huxley's Equation

Now PDE (1.4) with $\alpha = 1$ and $\gamma = 0$ is going to be solved. We found the following analytical solution in the literature [16, p. 34]

$$u^{\text{exact}}(x,\ t) = e^{\sqrt{\frac{\beta}{2}}x + \frac{\beta}{2}t}\left(c + e^{\sqrt{\frac{\beta}{2}}x + \frac{\beta}{2}t}\right)^{-1}.$$

The parameters are $\beta = 7,\ c = 8$ and $x_0 = 0,\ x_N = 2,\ t^0 = 0,\ t^{\text{fin}} = 0.4$, $\Delta x = 0.005$. Similarly to the previous subsection, we can construct the initial and the Dirichlet boundary conditions by taking this function at the appropriate time and space points. For comparison purposes, we use the NSFD schemes (4.3)–(4.6) again. The obtained error-curves are presented in Fig. 7. One can see that the methods behave well for this case as well.

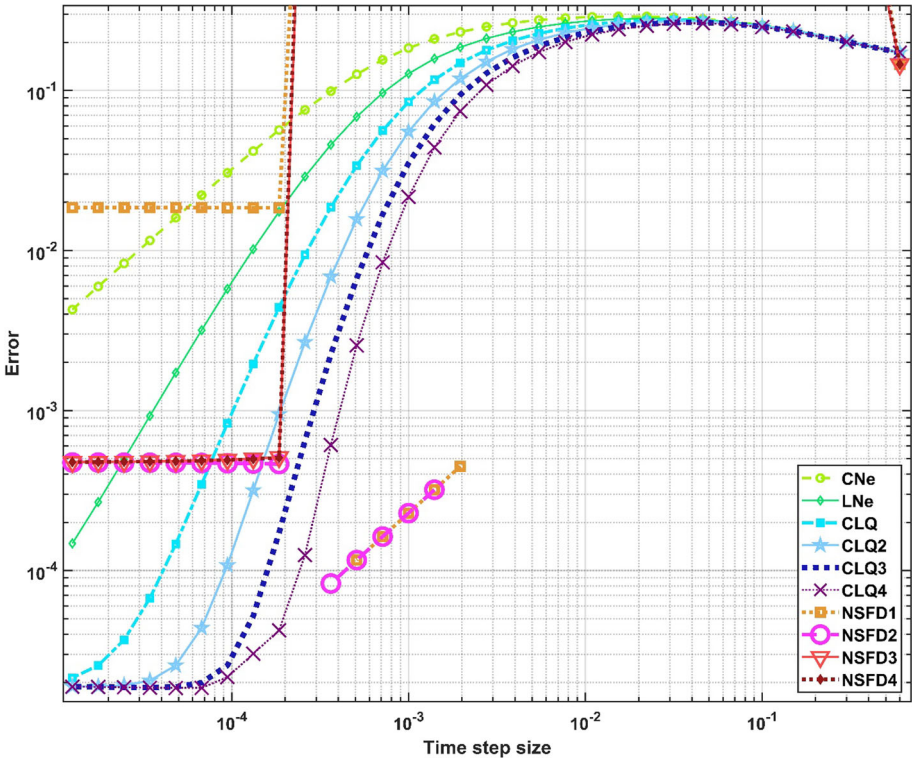

Fig. 5 L_∞ errors as a function of the time step size h for Nagumo's equation, Experiment 2

In Table 4 we present the maximum of numerical orders with the appropriate larger and smaller time step sizes and errors. The numerical orders can exceed their theoretical values for the CLQ2, CLQ3 and CLQ4 methods. In Fig. 8 the concentration u as a function of the space variable x is presented for the case of the analytical solution at the final time and two numerical solutions. One can see that even for this case, where the maximum errors of the CLQ and CLQ4 algorithms are 0.0321 and 0.0123, respectively, the numerical solution is qualitatively the same as the analytical one.

Table 3 Largest numerical order and the appropriate L_∞ errors of different methods for Experiment 2

Method	Larger h_1 and error	Smaller h_2 and error	Numerical order
CNe	1.75×10^{-5} and 5.96×10^{-3}	1.25×10^{-5} and 4.28×10^{-3}	0.988
LNe	3.44×10^{-5} and 9.26×10^{-4}	2.46×10^{-5} and 4.96×10^{-3}	1.855
CLL	1.32×10^{-4} and 1.40×10^{-3}	9.44×10^{-5} and 6.02×10^{-4}	2.501
CLQ	9.44×10^{-5} and 8.32×10^{-4}	6.74×10^{-5} and 3.46×10^{-4}	2.605
CLQ2	1.85×10^{-4} and 9.50×10^{-4}	1.32×10^{-4} and 3.18×10^{-4}	3.253
CLQ3	2.59×10^{-4} and 6.48×10^{-4}	1.85×10^{-4} and 1.73×10^{-4}	3.929
CLQ4	3.63×10^{-4} and 6.09×10^{-4}	2.59×10^{-4} and 1.25×10^{-4}	4.701

4.4 Experiment 4: Diffusion Equation with Strongly Space-Dependent Coefficient

We use the recently published [57] analytical solution of the diffusion equation where the coefficient has a power-law space dependence: $\alpha(x) = \overline{\alpha} x^m$. With this one has the following equation

$$\frac{\partial u(x,\,t)}{\partial t} = \overline{\alpha} \frac{\partial}{\partial x}\left(x^m \frac{\partial u(x,\,t)}{\partial x} \right).$$

The analytical solution of this equation is highly nontrivial, since it has the form

$$u(x,\,t) \;=\; \frac{t^{-\sigma}}{\sqrt{\eta}} \mathrm{e}^{-\frac{\eta^{-s}}{2\overline{\alpha} s^2}} \left\{ c_1 \cdot M_{\frac{(1+2s\sigma)|s|}{2s^2},\,\frac{s+1}{2s}}\left(\frac{|s|\eta^{-s}}{s^3\overline{\alpha}} \right) \;+\; c_2 \cdot W_{\frac{(1+2s\sigma)|s|}{2s^2},\,\frac{s+1}{2s}}\left(\frac{|s|\eta^{-s}}{s^3\overline{\alpha}} \right) \right\},$$

where $s = m - 2$, $\eta = x \cdot t^{1/s} \in \mathbb{R}$, σ, c_1 and c_2 are arbitrary constants, while M and W are the Whittaker functions [58, 59]. Here we consider only the $\mathrm{c}_1 = 0$ and $\mathrm{c}_2 = 1$ case and we set $m = 20$, $\sigma = 6$, $\overline{\alpha} = 1$, $t^0 = 2.3$, $t^{\mathrm{fin}} = 2.5$. The initial condition and the Dirichlet boundary conditions are calculated using the analytical solution, as in the previous examples. Let us consider the 1D interval $x \in [x_0,\; x_N]$, where $x_0 = 0.6$ and $N = 299$. The equidistant spatial grid is constructed using the $x_j = x_{j-1} + \Delta x$, $j = 1, ..., N$ rule, where x_0, x_1, ..., x_N are the coordinates of the cell borders and $\Delta x = 2 \cdot 10^{-3}$. The cell centres X_1, ..., X_N can be expressed as $X_j = x_{j-1} + \Delta x/2$, $j = 1, ..., N$. The power-law space dependence $\alpha(x) = \overline{\alpha} x^m$ of the diffusion coefficient is taken into account only at the level of the k material coefficients, so take $c \equiv 1$ and $\rho \equiv 1$ for simplicity, thus $C_j = \Delta x_j$, $j = 1, ..., N$. The resistances are calculated as follows:

$$R_{i,\,i+1} = \frac{X_{i+1} - X_i}{k_{i,\,i+1}} = \frac{\Delta x}{\overline{\alpha}(x_i)^m}, \quad i = 1, ..., N-1$$

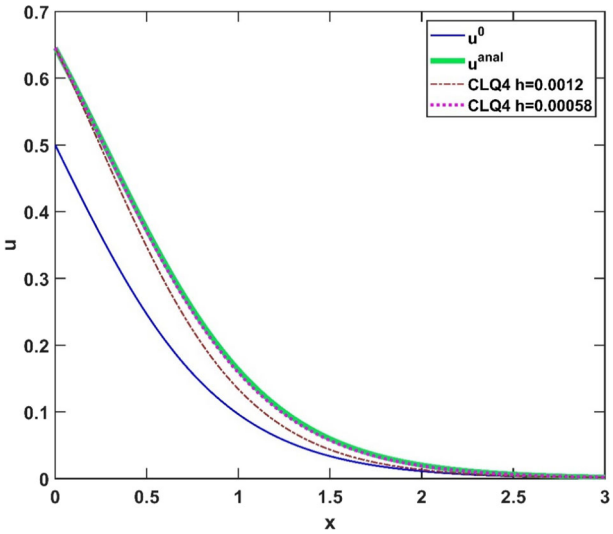

Fig. 6 The initial, analytical and numerical values of the variable u as a function of the space variable x in the case of Experiment 2

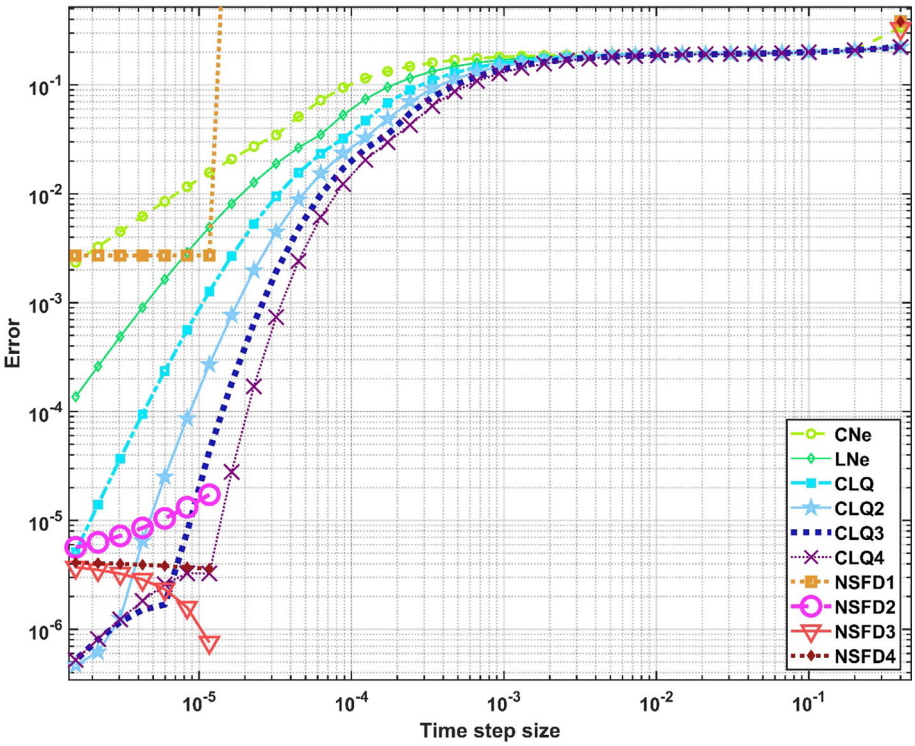

Fig. 7 L_∞ errors as a function of the time step size h for Experiment 3

Table 4 Largest numerical order and the appropriate L_∞ errors of different methods for Experiment 3

Method	Larger h_1 and error	Smaller h_2 and error	Numerical order
CNe	2.17×10^{-6} and 3.26×10^{-3}	1.55×10^{-6} and 2.35×10^{-3}	0.976
LNe	2.17×10^{-6} and 2.60×10^{-4}	1.55×10^{-6} and 1.37×10^{-4}	1.909
CLL	3.04×10^{-6} and 2.66×10^{-5}	2.17×10^{-6} and 1.06×10^{-5}	2.735
CLQ	2.17×10^{-6} and 1.40×10^{-5}	1.55×10^{-6} and 5.14×10^{-6}	2.971
CLQ2	4.26×10^{-6} and 6.48×10^{-6}	3.04×10^{-6} and 1.26×10^{-6}	4.867
CLQ3	1.17×10^{-5} and 4.32×10^{-5}	8.35×10^{-6} and 7.80×10^{-6}	5.091
CLQ4	1.64×10^{-5} and 2.80×10^{-5}	1.17×10^{-5} and 3.25×10^{-6}	6.391

Note that the concentrations, including the Dirichlet boundary conditions are calculated at the X_1 , ..., X_N cell centres while the conductivities are taken into account at the x_1 , ..., x_{N-1} cell borders. The stiffness ratio of this problem is 5.3×10^8, and $h^{\mathrm{EE}}_{\mathrm{MAX}} = 6.3 \times 10^{-8}$. We use the generalized methods, for example, Algorithm 6b. For comparison purposes, we use the UPFD method [32], which in our case has the form:

$$u_i^{n+1} = \frac{u_i^n + a_i h}{1 + 2r_i},$$

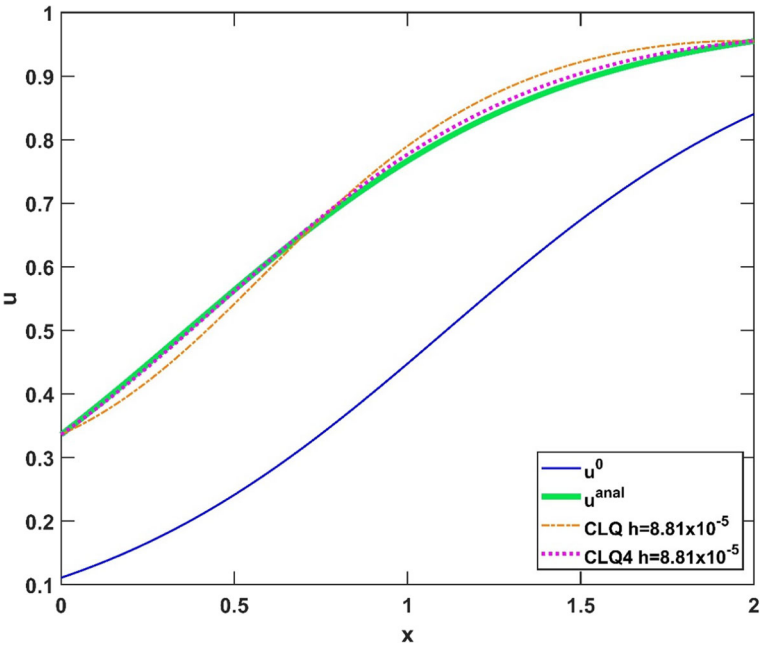

Fig. 8 The initial, analytical and numerical values of the variable u as a function of the space variable x in the case of Experiment 3

and the most standard 4th order Runge–Kutta method [60, p. 737]. In Fig. 9 we present the errors as a function of the time step size h, while in Table 5 the numerical orders are tabulated. One can see that the unconditionally stable methods behave well for this very stiff problem as well, but the numerical order of convergence is significantly less for the higher order schemes. Only one point is shown for the RK4 method, since above the smallest time step size, that scheme was unstable.

In Fig. 10 the analytical and a numerical solution at $t = t_{\text{fin}}$ by two methods are presented. One can see that apart from a small delay at about $x = 0.8$, the numerical solutions are close to the analytical one even if we use time step size, which is 3–4 orders of magnitude larger than the stability threshold for the RK methods.

Remark 4.1 If in Experiment 4 one increases N from 299 to 799, then the stiffness ratio and the mesh Fourier number drastically change to 6.7×10^{14}, and $h^{\text{EE}}_{\text{MAX}} = 5.6 \times 10^{-14}$. This implies that Runge–Kutta methods cannot be used at all, while the unconditionally stable methods produce almost the same error curves as in Fig. 9.

4.5 Experiment 5, General, Strongly Nonlinear Nagumo Equation with Non-equidistant Grid

We use a solitary wave analytical solution [61, 62] of Eq. (1.4), which is valid if $\alpha = 1$, and has the form

$$u(x, t) = \left\{ \frac{\gamma}{2} \left[1 + \tanh(\sigma \gamma (x - \vartheta t)) \right] \right\}^{1/\delta}.$$

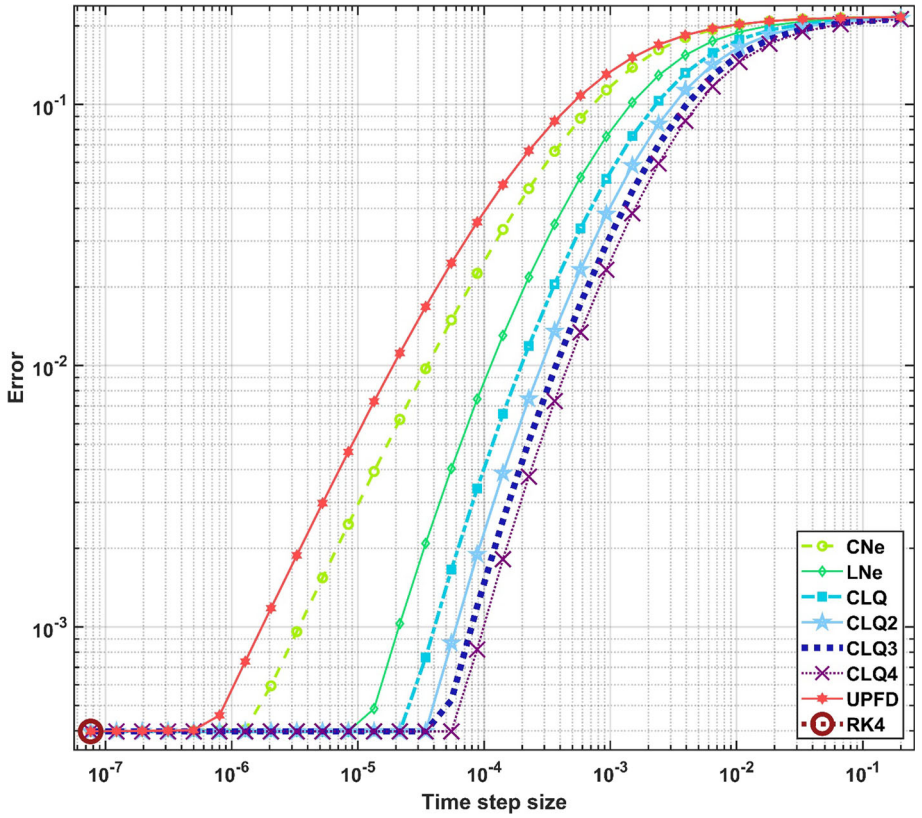

Fig. 9 L_∞ errors as a function of the time step size h for the linear Eq. (1.3) with a spatially varying diffusion coefficient (Experiment 4)

Table 5 Largest numerical order and the appropriate L_∞ errors of different methods for Experiment 4

Method	Larger h_1 and error	Smaller h_2 and error	Numerical order
CNe	3.28×10^{-6} and 9.59×10^{-4}	2.05×10^{-6} and 5.95×10^{-4}	1.016
LNe	2.15×10^{-5} and 1.02×10^{-3}	1.34×10^{-5} and 4.86×10^{-4}	1.593
CLQ	5.50×10^{-5} and 1.66×10^{-3}	3.44×10^{-5} and 7.63×10^{-4}	1.651
CLQ2	8.81×10^{-5} and 1.90×10^{-3}	5.50×10^{-5} and 8.68×10^{-4}	1.662
CLQ3	8.81×10^{-5} and 1.20×10^{-3}	5.50×10^{-5} and 5.24×10^{-4}	1.759
CLQ4	1.41×10^{-4} and 1.82×10^{-3}	8.81×10^{-5} and 8.19×10^{-4}	1.695

here $\vartheta = \frac{(1+\delta-\gamma)\rho}{2(1+\delta)}$ is the speed of the travelling wave, while $\rho = \sqrt{4\beta(1+\delta)}$, and $\sigma = \frac{\rho\delta}{4(1+\delta)}$. We use the following parameters: $\alpha = 1$, $\beta = 24$, $\gamma = 0.8$, $\delta = 6$ and $t^0 = 0$, $t^{\text{fin}} = 0.1$. The initial condition and the Dirichlet boundary conditions are calculated using the analytical solution, as in the previous examples. Now a non-equidistant spatial grid is constructed on the 1D interval $x \in [x_0, x_N]$ using the following procedure. We start with the definition of

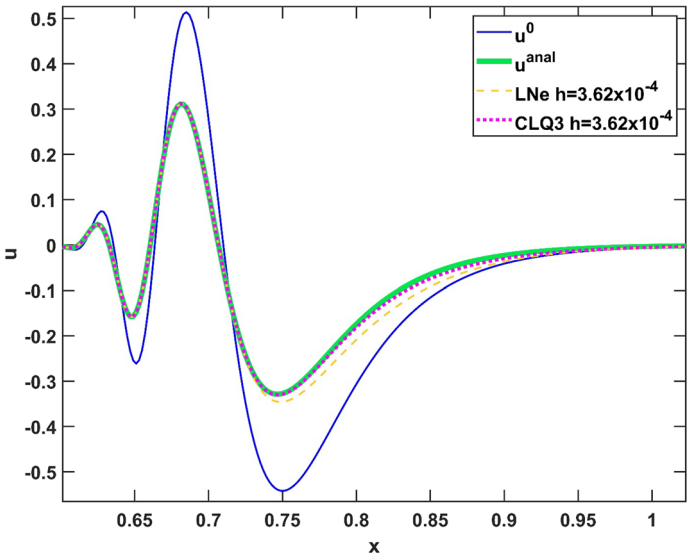

Fig. 10 The initial, analytical and numerical values of the variable u as a function of the space variable x in the case of Experiment 4

the coordinates $x_0\,,\ x_1\,,\ ...,\ x_N$ of the cell borders:

$$\begin{aligned} \Delta x_i &= \varepsilon \Delta x_{i-1}, \quad i = 2, ..., N \\ x_j &= x_{j-1} + \Delta x_j, \ j = 1, ..., N \end{aligned},$$

where $x_0 = -5$, $\Delta x_1 = 2 \cdot 10^{-3}$ and $\varepsilon = 1.002$. It means that the cell sizes are increasing from the left to the right. Now the cell centres $X_1\,,\ ...,\ X_N$ can be expressed as follows:

$$X_j = x_{j-1} + \frac{\Delta x_j}{2}, \quad j = 1, ..., N.$$

Now $C_j = \Delta x_j\,,\ \ j = 1, ..., N$, and the resistances are calculated as $R_{i,\,i+1} = X_{i+1} - X_i$, $i = 1, ..., N-1$. The capacities and the resistances are both increasing from 0.002 to about 0.0219. The stiffness ratio is 9.7×10^6, and $h^{\mathrm{EE}}_{\mathrm{MAX}} = 2.1 \times 10^{-6}$. For comparison purposes, we have coded the explicit exponential finite difference method proposed by Inan [62], which is a one-stage scheme with the following formula

$$u_i^{n+1} = u_i^n \exp\left\{h\left[\beta\left(1 - \left(u_i^n\right)^\delta\left(\left(u_i^n\right)^\delta - \gamma\right)\right) + \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{u_i^n \Delta x^2}\right]\right\}$$

for an equidistant mesh. We adapted this to our non-equidistant case as follows:

$$u_i^{n+1} = u_i^n \exp\left\{\frac{h}{u_i^n}\left[\beta u_i^n\left(1 - \left(u_i^n\right)^\delta\left(\left(u_i^n\right)^\delta - \gamma\right)\right) + \frac{u_{i-1}^n - u_i^n}{R_{i,i-1}C_i} + \frac{u_{i+1}^n - u_i^n}{R_{i,i+1}C_i}\right]\right\}.$$

In Fig. 11 we present the L_∞ errors as a function of the time step size h. One can see that the proposed methods behave quite similarly than in the previous cases, but some order-reduction can be observed for the higher order schemes (Table 6).

We also present the analytical and a numerical solution at $t = t_{\mathrm{fin}}$ by the method for a concrete time step size in Fig. 12.

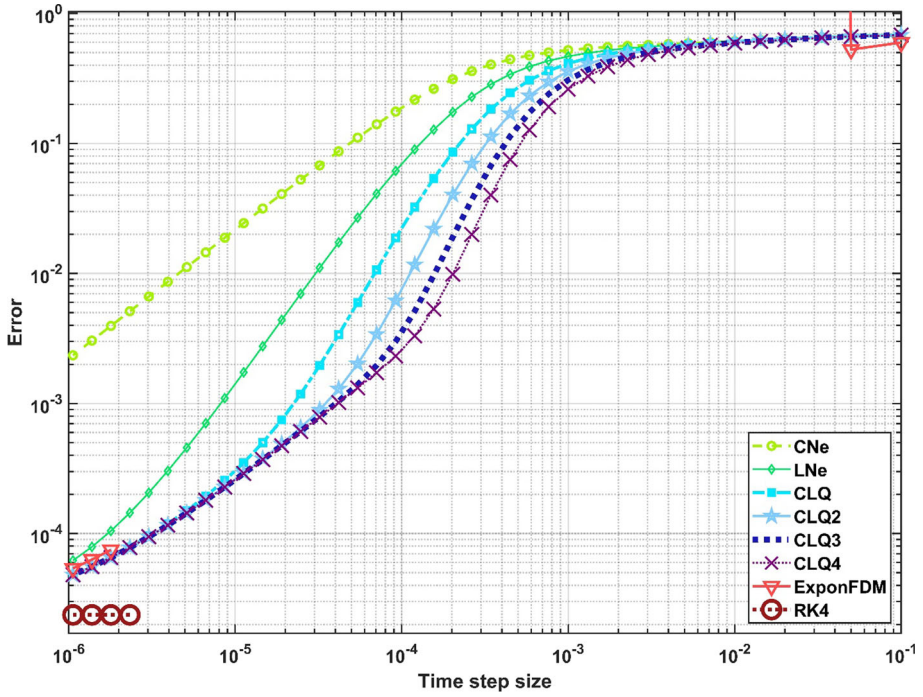

Fig. 11 L_∞ errors as a function of the time step size h for the highly nonlinear Eq. (1.4) with a spatially varying diffusion coefficient (Experiment 5)

Table 6 Largest numerical order and the appropriate L_∞ errors of different methods for Experiment 5

Method	Larger h_1 and error	Smaller h_2 and error	Numerical order
CNe	3.03×10^{-6} and 6.65×10^{-3}	2.33×10^{-6} and 5.12×10^{-3}	0.993
LNe	1.90×10^{-5} and 4.39×10^{-3}	1.46×10^{-5} and 2.76×10^{-3}	1.771
LNe3	5.44×10^{-5} and 7.71×10^{-3}	4.18×10^{-5} and 4.32×10^{-3}	2.204
CLQ	7.08×10^{-5} and 1.06×10^{-2}	5.44×10^{-5} and 5.98×10^{-3}	2.192
CLQ2	1.56×10^{-4} and 2.20×10^{-2}	1.20×10^{-4} and 1.17×10^{-2}	2.413
CLQ3	2.03×10^{-4} and 1.91×10^{-2}	1.56×10^{-4} and 9.64×10^{-3}	2.580
CLQ4	2.64×10^{-4} and 1.99×10^{-2}	2.03×10^{-4} and 9.87×10^{-3}	2.670

5 Numerical Experiments in 2D

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

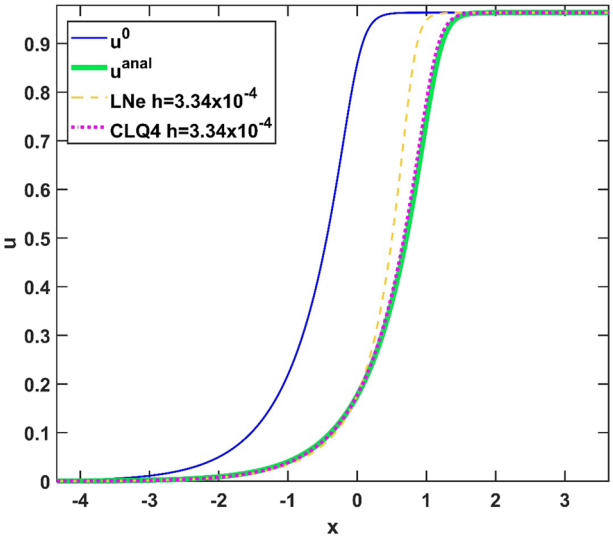

Fig. 12 The initial, analytical and numerical values of the variable u as a function of the space variable x in the case of Experiment 5

5.1 The Adaptive Time-Step Controller

In references [63–65], the authors proposed adaptive procedures for the solution of diffusion–reaction equations. They used one-dimensional systems to test their methods. However, according to our experience, the successful validation of a method in one dimension does not guarantee the same performance in two dimensions. This is why our adaptive-CLQ2 method will be applied to two-dimensional systems with a high stiffness ratio.

On the other hand, in [66–68] the authors suggested a step size control algorithm for the numerical integrations of ODEs. The authors tested their algorithms by applying them to a small system of ODEs (maximum 25 equations). We do believe that the step size controller is usually efficient when it is applied to a system of ODEs containing a limited number of equations, but the same controller can show a poor performance when it is applied to a big system of ODEs containing, for instance, 50 equations. So, if we want to apply those proposed controllers to get the numerical solution of Eq. (2.2), then we will be confined to a small size of the matrix M, which does not match the engineering applications. Our adaptive-CLQ2 will be applied to a system of 900 equations, and it can still show good performance, as we will see. Moreover, in [66] (Paragraph 4.1) one can straightforwardly calculate the stiffness ratio of the system, which is 4.66, while, as we will see, the stiffness ratio of our tested systems can be of order 10^{11}.

The main objective of this section is to design an adaptive time-step controller in order to achieve a given level of accuracy in the solution at a minimal computational cost. Numerical calculations based on an adaptive controller allow changing the time step h_n such that the error can be lower than a prescribed value. The most common way (see e.g. [69]) to adapt the step size is the elementary controller:

$$h_{n+1} = 0.9\left(\frac{\varepsilon}{r_n}\right)^{\frac{1}{p}} h_n \tag{5.1}$$

here ε is the desired tolerance and p is the order of the used method, while r_n is the local error estimator at time level n. Recall that, by definition, the local error for an algorithm is the error generated by a single step of the algorithm, under the assumption that we start the step with the exact solution. There are many adaptive techniques for local error estimations. One of them is the so-called embedded Runge–Kutta pairs. In that technique, two RK methods are used to calculate the local error estimation, where one of them is more accurate than the other [70]. The Runge–Kutta-Fehlberg method (denoted RKF45) has two methods of orders 4 and 5, as it is well explained in the literature [71]. Inspired by that, and since our CLQ2 is a multi-stage embedded method, we decided to follow the same fashion of embedded Runge–Kutta pairs in calculating the local truncation error.

Let $\hat{u}_i^{i+1}$ and u_i^{n+1} be the approximated solutions at time level $n+1$ obtained by the CLQ (third order) and CLQ2 (fourth order) methods respectively. We estimate the local error using the following formula:

$$r_n = \max_{1 \le i \le N} \left| \hat{u}_i^{n+1} - u_i^{n+1} \right| \tag{5.2}$$

If $r_n \le \varepsilon$, we accept the solution and set $t_{n+1} = t_n + h_n$, while the solution is u_i^{n+1} (since u_i^{n+1} is more accurate than $\hat{u}_i^{n+1}$).

If $r_n > \varepsilon$, we reject the time step and try again with a new time step calculated by Eq. (5.1).

Our adaptive-CLQ2 method will be applied not only to linear diffusion–reaction problems but also to nonlinear types. In the CLQ2 method, the effect of the nonlinear reaction term (Nagumo term) is only considered at the fourth stage, but not in the third stage, as we can see in Eq. (3.4). In other words, $\hat{u}^{n+1}$ does not include the effect of the nonlinear reaction term, while the value of u^{n+1} includes that effect. So, the difference between those two values, which represents r_n, will not reflect the real behaviour of the local error. To fix this, we calculated $\hat{u}_i^{n+1}$ taking into account the effect of the nonlinear reaction term. If we applied here the same pseudo-implicit treatment as in Eq. (3.4), then the difference between u_i^{n+1} and $\hat{u}_i^{n+1}$ would be extremely small when the reaction term dominates the diffusion term, and we could not trust in the error estimator. Therefore, we choose the following simple explicit treatment of the reaction term:

$$\hat{u}_i^{n+1} = u_i^{CLQ} + \beta u_i^{CLQ} \left(1 - \left(u_i^{CLQ} \right)^{\delta} \right) \left(\left(u_i^{CLQ} \right)^{\delta} - \gamma \right) h \tag{5.3}$$

In the last equation, the value of u_i^{CLQ} can be calculated using stages 1, 2 and 3 in Algorithm 6. This algorithm is valid in the linear ($\beta = 0$) and in the nonlinear case ($\beta > 0$) as well.

In the following experiments, we will test our improved adaptive time step controller in the case of the linear diffusion equation with a heat source and the nonlinear diffusion equation with the Nagumo term. We do not claim that our adaptive time step controller is efficient for solving all kinds of systems. However, this algorithm can reduce the time required for achieving the numerical calculations when applied to some stiff systems. The advantages and disadvantages of this algorithm will be illustrated for each experiment individually. The adaptive-CLQ2 will be compared to the CLQ2, RK4, and RKF45 methods as well.

5.2 Experiment 6: Heat Equation with Heat Source

In this experiment we consider Eq. (2.2) in 2 space-dimensions $(x, y, t) \in [0, 1] \times [0, 1] \times \left[0, 2 \times 10^{-4}\right]$, subjected to zero Neumann boundary conditions. The space domain was

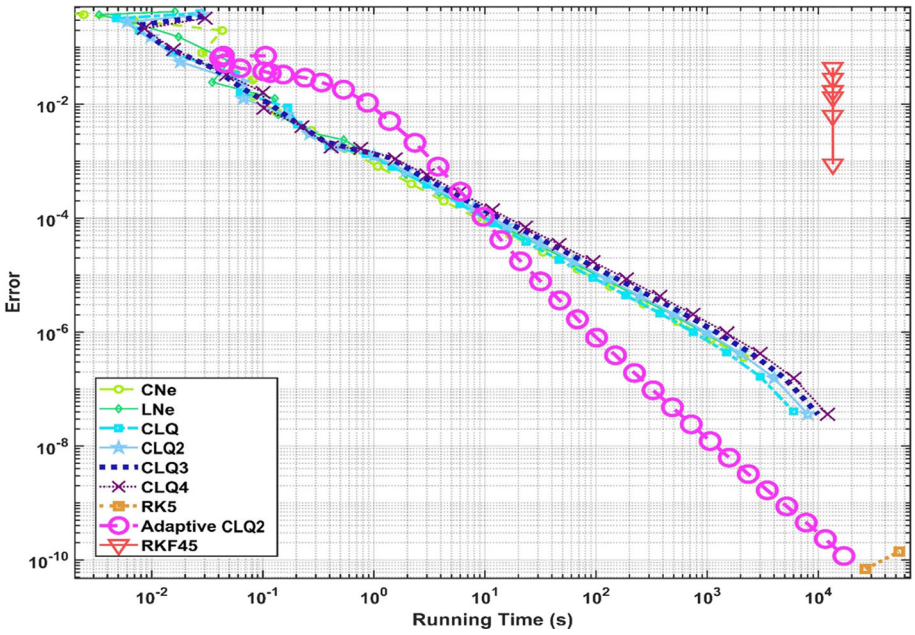

Fig. 13 L_∞ errors as a function of the total running times in case of a system of 900 cells with heat source

divided into $N = N_x \times N_y = 30 \times 30$, thus we have 900 cells. The initial conditions were $u_i(0) = rand$, where *rand* is a pseudo-random number in the unit interval. The resistances and capacities obeyed the following equations:

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

The heat source can be given by the following equation:

$$q = 10\sin(\pi x)$$

The stiffness ratio of the introduced system is 1.54×10^{11} ($h^{\mathrm{EE}}_{\mathrm{MAX}} = 6.4 \times 10^{-12}$). Figure 13 shows the error as a function of the total running time at different levels of accuracy. One can see that the adaptive-CLQ2 method is more efficient than the CLQ2 and the other methods when high-accuracy calculations are required. The order-reduction due to the high stiffness is smaller for the adaptive than for the non-adaptive CLQ2 algorithm. The RK5 method can provide reliable calculations only when the time step size is less than 6.4×10^{-12}, otherwise it is unstable. A possible remedy for the problem of instability could be the usage of the Runge–Kutta-Fehlberg (RKF45) method, but it is not much faster, and it reduces the accuracy of the calculations, as we can see in the figure.

5.3 Experiment 7: FitzHugh-Nagumo equation

In this experiment we consider Eq. (1.4) with $\beta = 10,\ \ \gamma = 0.1,\ \ \delta = 0.5$. All other settings and circumstances were the same as in Experiment 6, except that now $q \equiv 0$. For comparison purposes, we used not only the non-adaptive RK5 and the adaptive RKF45, but the implicit and non-adaptive Crank-Nicolson (Cr-Ni) scheme with Strang-splitting. It means that the

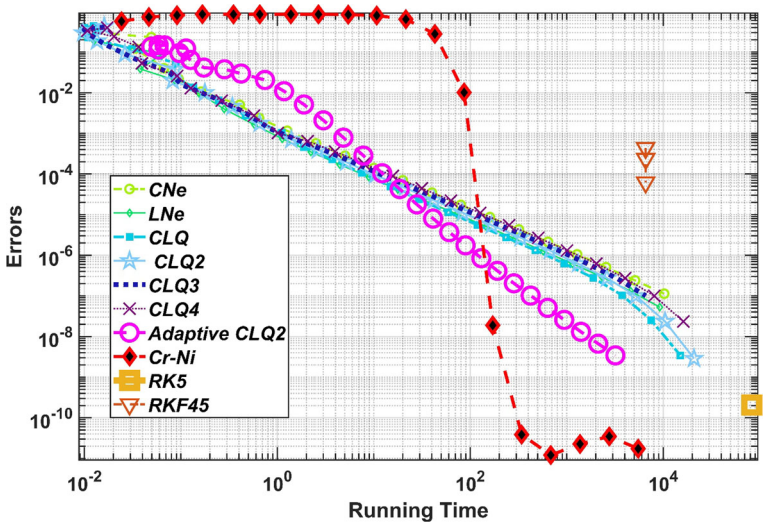

Fig. 14 L_∞ errors as a function of the total running times in case of system of 900 cells with Nagumo term

PDE contains only the diffusion term is solved by the standard Cr-Ni scheme, but before and after this stage, formula (3.4) is applied with half time step sizes to take into account the effect of the nonlinear term.

Figure 14 suggests that the adaptive-CLQ2 is more efficient than the CLQ2 when high accuracy is required. Time steps of order 10^{-11} have to be used in the case of RK5 to reach the stability region of the algorithm. The RKF45 method is present with only a few points on the right of the figure, so none of these can be proposed for this type of problem. The Cr-Ni-Strang algorithm is completely useless for small and medium time step sizes, but suddenly becomes extremely accurate when h is decreased. So, if high accuracy is required, this implicit method can be recommended, but only for these relatively small system sizes (e.g. below $N = 1000$). We will see soon that for a larger system, the implicit solvers are much slower than the explicit ones.

5.4 Experiment 8: Heat Equation with Moving and Pulsing Heat Source

In this experiment we consider Eq. (2.2) in 2 space-dimensions $(x,\ y,\ t) \in [0,\ 1] \times [0,\ 1] \times [0,\ 1]$, subjected to zero Neumann boundary conditions. The space domain was divided into $N = N_x \times N_y = 100 \times 100$ cells, thus we have 10,000 cells. The initial conditions were $u_i(0) = rand$, while the resistances and capacities were set to unity, $C = 1$, $R_x = 1$, $R_y = 1$ to obtain a non-stiff system. The heat source can be given by the following equation:

$$q(x,\ t) = 100t^2\big(\max\{\cos^2(6\pi t),\ \ 0.5\} - 0.5\big)\exp\frac{(x - x_0 - v_x t)^2 + \big(y - y_0 - \frac{a_y}{2}t^2\big)^2}{r_0}.$$

It means we have a Gaussian spot-like heat source with effective diameter $r_0 = 4\Delta x = 0.04$, moving with initial horizontal velocity $v_x = 0.8$ and vertical acceleration $a_y = -2$ from the initial position $x_0 = 0.05$, $y_0 = 0.9$. Figure 15 shows the contour of the temperature

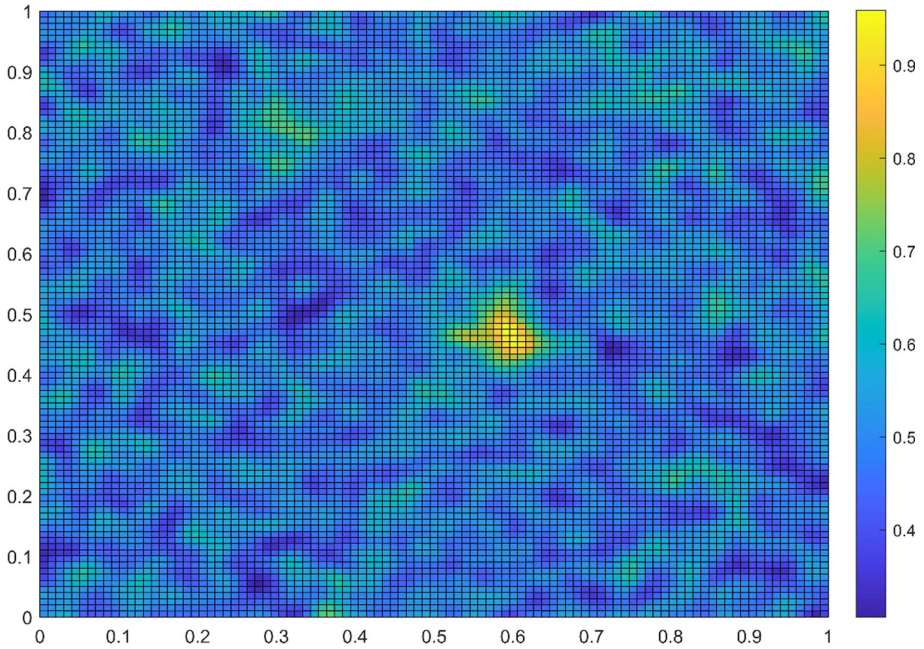

Fig. 15 The contour of the temperature distribution at the end of the simulation in Experiment 8

distribution at $T = 1$. For comparison purposes, we used not only the RK5, RKF45 and the standard (non-adaptive) Crank-Nicolson, but the four implicit ODE solvers of MATLAB:

- ode15s, variable-step, variable-order solver based on the numerical differentiation formulas (NDFs) of orders 1 to 5
- ode23t, an implementation of the trapezoidal rule using a "free" interpolant.
- ode23s, a 2nd order modified Rosenbrock formula.
- ode23tb, a combination of the backward differentiation formula and the trapezoidal rule;

Figure 16 shows the error as a function of the total running time. One can see that the effect of conditional consistency is not present owing to the adaptive controller in the case of the adaptive-CLQ2 method. It makes this solver more efficient than all the other methods for high- and medium accuracy, and the whole CNe-CLQ family is quite efficient for low and medium accuracy.

6 Conclusions

Our aim was to construct a family of fully explicit numerical algorithms to solve parabolic PDEs, especially the non-stationary diffusion (or heat) and diffusion–reaction equations. The new algorithms are one step time-integrators, and consist of 3–6 stages. The algorithms can be applied to general linear ODE systems (for example, the spatially discretized heat equation, even in the case of unstructured grids as well), and for this case, we show that the convergence in the time step size of the 3-stage CLQ scheme is three, while it is four for the 4, 5 and 6-stage algorithms CLQ2, CLQ3 and CLQ4. Although the accuracy is not always good for large time step sizes, the dynamical consistency, as well as stability, is guaranteed for

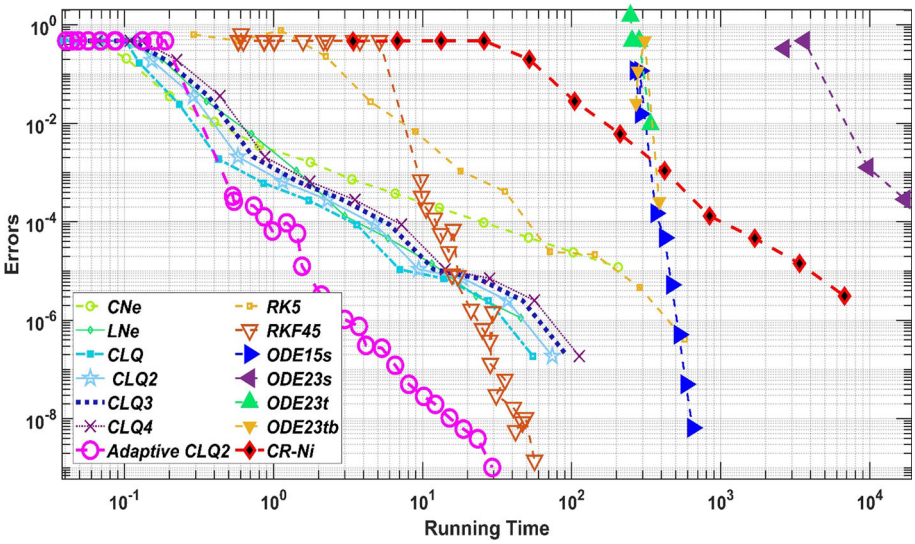

Fig. 16 L_∞ errors as a function of the total running times in case of a system of 10,000 cells with moving heat source

arbitrary time step sizes for the linear diffusion equation, the nonlinear Fisher's and Huxley's equation, and, within a wide parameter domain, for the Nagumo's equation. This property is especially advantageous when one models physical processes, where the variable is the concentration, which must be in the unit interval.

We used one-dimensional analytical solutions of the linear diffusion equation with a space-dependent diffusion coefficient, as well as the Fisher's, Huxley's and Nagumo's equations with rather large nonlinear coefficients to verify and test the new methods. After this, we developed the adaptive step size control version of the algorithms. The values predicted in the third stage are compared to the values calculated in the fourth stage to obtain an estimation of the local error. In this way an embedded method was implemented, where the adaptation of the time step size is performed with very little extra computational effort. The performance of the non-adaptive and adaptive methods was examined in the case of three 2-dimensional systems with inhomogeneous parameters and randomly generated discontinuous initial conditions. We showed that the proposed methods are competitive as they can give quite accurate results orders of magnitude faster than the standard Runge–Kutta and Crank-Nicolson schemes, depending on the system parameters.

In the near future, we plan to apply the proposed methods to other nonlinear equations as well. We are going to systematically investigate the possible treatment of the nonlinear terms, including linearizations [72] and operator-splitting techniques.

Author Contribution Conceptualization, design, methodology, supervision and the one-dimensional numerical investigation are due to E.K. Analytical proofs were performed by J.M. The research on the adaptive step size controller and the related investigation were done by M.S. The first draft of the manuscript was written by E.K. and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding This research was supported 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 the first author.

Data Availability The datasets generated during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Dattagupta, S.: Diffusion Formalism and Applications. Taylor & Francis, Milton Park (2014)
2. Ghez, R.: Diffusion Phenomena: Cases and Studies. Dover Publications Inc, New York (2001)
3. Bennett, T.: Transport by Advection and Diffusion. Wiley, New York (2012)
4. Pisarenko, I., Ryndin, E.: Numerical drift-diffusion simulation of GaAs p-i-n and Schottky-Barrier photodiodes for high-speed AIIIBV on-chip optical interconnections. Electronics **5**(4), 52 (2016). https://doi.org/10.3390/electronics5030052
5. Fisher, D.J.: Defects and Diffusion in Carbon Nanotubes. Trans Tech Publications, Wollerau (2014)
6. Fradin, C.: On the importance of protein diffusion in biological systems: the example of the Bicoid morphogen gradient. Biochim. Biophys. Acta Proteins Proteom **1865**(11), 1676–1686 (2017). https://doi.org/10.1016/j.bbapap.2017.09.002
7. Yu, H., et al.: The moisture diffusion equation for moisture absorption of multiphase symmetrical sandwich structures. Mathematics **10**(15), 2669 (2022). https://doi.org/10.3390/math10152669
8. Zimmerman, R.W.: The Imperial College lectures in petroleum engineering. World Scientific Publishing, Singapore, London (2018)
9. Li, Y., van Heijster, P., Marangell, R., Simpson, M.J.: Travelling wave solutions in a negative nonlinear diffusion–reaction model. J. Math. Biol. **81**, 1495–1522 (2020). https://doi.org/10.1007/s00285-020-01547-1
10. Weickenmeier, J., Jucker, M., Goriely, A., Kuhl, E.: A physics-based model explains the prion-like features of neurodegeneration in Alzheimer's disease, Parkinson's disease, and amyotrophic lateral sclerosis. J. Mech. Phys. Solids **124**, 264–281 (2019). https://doi.org/10.1016/j.jmps.2018.10.013
11. Campos, D., Llebot, J.E., Fort, J.: Reaction-diffusion pulses: a combustion model. J. Phys. A. Math. Gen. **37**, 6609–6621 (2004). https://doi.org/10.1088/0305-4470/37/26/001
12. Agbavon, K.M., Appadu, A.R.: Construction and analysis of some nonstandard finite difference methods for the FitzHugh–Nagumo equation. Numer. Methods Partial Differ. Equ. **36**(5), 1145–1169 (2020). https://doi.org/10.1002/num.22468
13. Zhao, Y., Billings, S.A., Guo, Y., Coca, D., Dematos, L.L., Ristic, R.I.: Spatio-temporal modeling of wave formation in an excitable chemical medium based on a revised Fitzhugh-Nagumo model. Int. J. Bifurc. Chaos **21**(2), 505–512 (2011). https://doi.org/10.1142/S0218127411028556
14. Seadawy, A.R., Rizvi, S.T.R., Ahmed, S.: Multiple lump, generalized breathers, Akhmediev breather, manifold periodic and rogue wave solutions for generalized Fitzhugh-Nagumo equation: applications in nuclear reactor theory. Chaos Solitons Fractals **161**, 112326 (2022). https://doi.org/10.1016/j.chaos.2022.112326
15. Appadu, A.R., Inan, B., Tijani, Y.O.: Comparative study of some numerical methods for the Burgers-Huxley equation. Symmetry (Basel) **11**, 1333 (2019). https://doi.org/10.3390/sym11111333
16. Bradshaw-Hajek, B.: Reaction-Diffusion Equations for Population Genetics. Wollongong (2004)
17. Hassan, M.M., Abdel-Razek, M.A., Shoreh, A.A.H.: Explicit exact solutions of some nonlinear evolution equations with their geometric interpretations. Appl. Math. Comput. **251**, 243–252 (2015). https://doi.org/10.1016/j.amc.2014.11.046
18. Ramos, J.I.: A review of some numerical methods for reaction-diffusion equations. Math. Comput. Simul **25**(6), 538–548 (1983). https://doi.org/10.1016/0378-4754(83)90127-1
19. Kovács, E., Nagy, Á., Saleh, M.: A set of new stable, explicit, second order schemes for the non-stationary heat conduction equation. Mathematics **9**(18), 2284 (2021). https://doi.org/10.3390/math9182284
20. Appau, P.O., Dankwa, O.K., Brantson, E.T.: A comparative study between finite difference explicit and implicit method for predicting pressure distribution in a petroleum reservoir. Int. J. Eng. Sci. Technol. (2019). https://doi.org/10.4314/ijest.v11i4.3

21. Moncorgé, A., Tchelepi, H.A., Jenny, P.: Modified sequential fully implicit scheme for compositional flow simulation. J. Comput. Phys. **337**(15), 98–115 (2017). https://doi.org/10.1016/j.jcp.2017.02.032
22. Albert, C., Ruiz-Gironés, E., Sarrate, J.: High-order HDG formulation with fully implicit temporal schemes for the simulation of two-phase flow through porous media. Int. J. Numer. Methods Eng. (2021). https://doi.org/10.1002/nme.6674
23. Ramos, J.I.: Modified equation techniques for reactive-diffusive systems. Part 1: explicit, implicit and quasilinear methods. Comput. Methods Appl. Mech. Eng. **64**(1–3), 195–219 (1987). https://doi.org/10.1016/0045-7825(87)90040-5
24. Ramos, J.I.: Linearization methods for reaction-diffusion equations: 1-D problems. Appl. Math. Comput. **88**(2–3), 199–224 (1997). https://doi.org/10.1016/S0096-3003(96)00328-1
25. Ramos, J.I.: Linearization methods for reaction-diffusion equations: Multidimensional problems. Appl. Math. Comput. **88**(2–3), 225–254 (1997). https://doi.org/10.1016/S0096-3003(96)00327-X
26. Manaa, S., Sabawi, M.: Numerical solution and stability analysis of huxley equation. AL-Rafidain J. Comput. Sci. Math. **2**(1), 85–97 (2005). https://doi.org/10.33899/csmj.2005.164070
27. Kadioglu, S.Y., Knoll, D.A.: A fully second order implicit/explicit time integration technique for hydrodynamics plus nonlinear heat conduction problems. J. Comput. Phys. **229**(9), 3237–3249 (2010). https://doi.org/10.1016/j.jcp.2009.12.039
28. Chen, H., Kou, J., Sun, S., Zhang, T.: Fully mass-conservative IMPES schemes for incompressible two-phase flow in porous media. Comput. Methods Appl. Mech. Eng. **350**, 641–663 (2019). https://doi.org/10.1016/j.cma.2019.03.023
29. Lee, S.H., Ţene, M., Du, S., Wen, X., Efendiev, Y.: A conservative sequential fully implicit method for compositional reservoir simulation. J. Comput. Phys. **428**, 109961 (2021). https://doi.org/10.1016/j.jcp.2020.109961
30. Gagliardi, F., Moreto, M., Olivieri, M., Valero, M.: The international race towards Exascale in Europe. CCF Trans. High Perform. Comput. **1**, 3–13 (2019). https://doi.org/10.1007/s42514-019-00002-y
31. Reguly, I.Z., Mudalige, G.R.: Productivity, performance, and portability for computational fluid dynamics applications. Comput. Fluids (2020). https://doi.org/10.1016/j.compfluid.2020.104425
32. Chen-Charpentier, B.M., Kojouharov, H.V.: An unconditionally positivity preserving scheme for advection-diffusion reaction equations. Math. Comput. Model. **57**, 2177–2185 (2013). https://doi.org/10.1016/j.mcm.2011.05.005
33. Sun, G.F., Liu, G.R., Li, M.: An efficient explicit finite-difference scheme for simulating coupled biomass growth on nutritive substrates. Math. Probl. Eng. (2015). https://doi.org/10.1155/2015/708497
34. Appadu, A.R.: Performance of UPFD scheme under some different regimes of advection, diffusion and reaction. Int. J. Numer. Methods Heat Fluid Flow **27**(7), 1412–1429 (2017). https://doi.org/10.1108/HFF-01-2016-0038
35. Kolev, M.K., Koleva, M.N., Vulkov, L.G.: An unconditional positivity-preserving difference scheme for models of cancer migration and invasion. Mathematics **10**(1), 1–22 (2022). https://doi.org/10.3390/math10010131
36. Chertock, A., Kurganov, A.: A second-order positivity preserving central-upwind scheme for chemotaxis and haptotaxis models. Numer. Math. **111**(2), 169–205 (2008). https://doi.org/10.1007/s00211-008-0188-0
37. Chapwanya, M., Lubuma, J.M.S., Mickens, R.E.: Positivity-preserving nonstandard finite difference schemes for cross-diffusion equations in biosciences. Comput. Math. Appl. **68**(9), 1071–1082 (2014). https://doi.org/10.1016/j.camwa.2014.04.021
38. Songolo, M.E.: A positivity-preserving nonstandard finite difference scheme for parabolic system with cross-diffusion equations and nonlocal initial conditions. Am. Sci. Res. J. Eng. Technol. Sci. **18**(1), 252–258 (2016)
39. Agbavon, K.M., Appadu, A.R., Khumalo, M.: On the numerical solution of Fisher's equation with coefficient of diffusion term much smaller than coefficient of reaction term. Adv. Differ. Equ. **146**, 1–33 (2019). https://doi.org/10.1186/s13662-019-2080-x
40. Liu, H., Leung, S.: An alternating direction explicit method for time evolution equations with applications to fractional differential equations. Methods Appl. Anal. **26**(3), 249–268 (2020). https://doi.org/10.4310/MAA.2019.v26.n3.a3
41. Gordon, P.: Nonsymmetric difference equations. J. Soc. Ind. Appl. Math. **13**(3), 667–673 (1965). https://doi.org/10.1137/0113044
42. Gourlay, A.R.: Hopscotch: a fast second-order partial differential equation solver. IMA J. Appl. Math. **6**(4), 375–390 (1970)

43. Saleh, M., Nagy, Á., Kovács, E.: Part 3: construction and investigation of new numerical algorithms for the heat equation. Multidiszcip. Tudományok **10**(4), 349–360 (2020). https://doi.org/10.35925/j.multi.2020.4.38
44. Kovács, E.: A class of new stable, explicit methods to solve the non-stationary heat equation. Numer. Methods Partial Differ. Equ. **37**(3), 2469–2489 (2020). https://doi.org/10.1002/num.22730
45. Saleh, M., Endre, K., Gábor, P.: Testing and improving a non-conventional unconditionally positive finite difference method. Multidiszcip. Tudományok **10**(4), 206–213 (2020)
46. Kovács, E., Nagy, Á.: A new stable, explicit, and generic third-order method for simulating conductive heat transfer. Numer. Methods Partial Differ. Equ. **39**(2), 1504–1528 (2023). https://doi.org/10.1002/num.22943
47. Kovács, E.: New stable, explicit, first order method to solve the heat conduction equation. J. Comput. Appl. Mech. **15**(1), 3–13 (2020). https://doi.org/10.32973/jcam.2020.001
48. Fayazbakhsh, M.A., Bagheri, F., Bahrami, M.: A resistance-capacitance model for real-time calculation of cooling load in HVAC-R systems. J. Therm. Sci. Eng. Appl. (2015). https://doi.org/10.1115/1.4030640
49. Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numer. **19**, 209–286 (2010). https://doi.org/10.1017/S0962492910000048
50. Hiriart-Urruty, J.-B., Lemaréchal, C.: Fundamentals of Convex Analysis. Springer Verlag, Berlin (2001)
51. Holmes, M.H.: Introduction to Numerical Methods in Differential Equations. Springer, New York (2007)
52. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic problems. Springer Verlag, Berlin (1991)
53. Fletcher, C.A.J.: Computational Techniques for Fluid Dynamics, vol. I. Springer Verlag, New South Wales (2006)
54. Mbroh, N.A., Munyakazi, J.B.: A robust numerical scheme for singularly perturbed parabolic reaction-diffusion problems via the method of lines. Int. J. Comput. Math. (2021). https://doi.org/10.1080/00207160.2021.1954621
55. Appadu, A.R., Tijani, Y.O., Munyakazi, J.B.: Computational study of some numerical methods for the generalized burgers-huxley equation. First Int. Conf. CSMCS **2021**, 56–67 (2020)
56. Bastani, M., Salkuyeh, D.K.: A highly accurate method to solve Fisher's equation. Pramana J. Phys. **78**, 335–346 (2012). https://doi.org/10.1007/s12043-011-0243-8
57. Saleh, M., Kovács, E., Barna, I.F., Mátyás, L.: New analytical results and comparison of 14 numerical schemes for the diffusion equation with space-dependent diffusion coefficient. Mathematics **10**(15), 2813 (2022). https://doi.org/10.3390/math10152813
58. Olver, F.W.J., Lozier, D.W., Boisvert, R.F., Clark, C.W.: NIST Handbook of Mathematical Functions, vol. 66. Cambridge Univ. Press, New York (2011)
59. Wikipedia, "Whittaker function." [Online]. Available: https://en.wikipedia.org/wiki/Whittaker_function.
60. Chapra, S.C., Canale, R.P.: Numerical Methods for Engineers, Seventh Edition, 7th edn. McGraw-Hill Science/Engineering/Math, New York (2015)
61. Wang, X.Y., Zhu, Z.S., Lu, Y.K.: Solitary wave solutions of the generalised Burgers-Huxley equation. J. Phys. A. Math. Gen. **23**(3), 271–274 (1990). https://doi.org/10.1088/0305-4470/23/3/011
62. Inan, B.: Finite difference methods for the generalized Huxley and Burgers-Huxley equations. Kuwait J. Sci. **44**(3), 20–27 (2017)
63. Yu, J.L.: Adaptive optimal m-stage Runge-Kutta methods for solving reaction-diffusion-chemotaxis systems. J. Appl. Math. (2011). https://doi.org/10.1155/2011/389207
64. Jannelli, A.: Adaptive numerical solutions of time-fractional advection–diffusion–reaction equations. Commun. Nonlinear Sci. Numer. Simul. **105**, 106073 (2022). https://doi.org/10.1016/j.cnsns.2021.106073
65. Ramos, J.I.: Adaptive methods of lines for one-dimensional reaction-diffusion equations. Int. J. Numer. Methods Fluids **16**(8), 697–723 (1993). https://doi.org/10.1002/fld.1650160804
66. Eriksson, K., Johnson, C., Logg, A.: Adaptive computational methods for parabolic problems. In: Encyclopedia of Computational Mechanics. John Wiley & Sons Ltd, Chichester, UK (2004)
67. Fedoseev, P., Pesterev, D., Karimov, A., Butusov, D.: New step size control algorithm for semi-implicit composition ODE solvers. Algorithms **15**(8), 275 (2022). https://doi.org/10.3390/a15080275
68. Gustafsson, K., Lundh, M., Söderlind, G.: A PI stepsize control for the numerical solution of ordinary differential equations. BIT **28**(2), 270–287 (1988). https://doi.org/10.1007/BF01934091
69. Söderlind, G., Wang, L.: Adaptive time-stepping and computational stability. J. Comput. Appl. Math. **185**(2), 225–243 (2006). https://doi.org/10.1016/j.cam.2005.03.008
70. Fekete, I., Conde, S., Shadid, J.N.: Embedded pairs for optimal explicit strong stability preserving Runge-Kutta methods. J. Comput. Appl. Math. **412**, 114325 (2022). https://doi.org/10.1016/j.cam.2022.114325

71. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes 3rd Edition: The Art of Scientific Computing, vol. 1, 3rd edn. Cambridge University Press, Cambridge (2007)
72. Ramos, J.I.: A piecewise time-linearized method for the logistic differential equation. Appl. Math. Comput. **93**(2–3), 139–148 (1998). https://doi.org/10.1016/S0096-3003(97)10049-2