algorithms

MDPI

Article

Consistency and Convergence Properties of 20 Recent and Old Numerical Schemes for the Diffusion Equation

Ádám Nagy, János Majár and Endre Kovács *

Institute of Physics and Electrical Engineering, University of Miskolc, 3515 Miskolc, Hungary
* Correspondence: kendre01@gmail.com or fizendre@uni-miskolc.hu

Abstract: We collected 20 explicit and stable numerical algorithms for the one-dimensional transient diffusion equation and analytically examined their consistency and convergence properties. Most of the methods used have been constructed recently and their truncation errors are given in this paper for the first time. The truncation errors contain the ratio of the time and space steps; thus, the algorithms are conditionally consistent. We performed six numerical tests to compare their performance and try to explain the observed accuracies based on the truncation errors. In one of the experiments, the diffusion coefficient is supposed to change strongly in time, where a nontrivial analytical solution containing the Kummer function was successfully reproduced.

Keywords: truncation error; diffusion; heat conduction; explicit time-integration; unconditionally stable numerical methods

Citation: Nagy, Á.; Majár, J.; Kovács, E. Consistency and Convergence Properties of 20 Recent and Old Numerical Schemes for the Diffusion Equation. *Algorithms* **2022**, *15*, 425. https://doi.org/10.3390/a15110425

Academic Editors: Dunhui Xiao and Shuai Li

Received: 15 September 2022
Accepted: 7 November 2022
Published: 10 November 2022

1. Introduction

Fickian diffusion or Fourier-type heat conduction is one of the most fundamental mass- or energy-transport processes. In the simplest case, it can be described by a single linear partial differential equation (PDE) of space and time:

$$\frac{\partial u(x,t)}{\partial t} = \alpha \frac{\partial^2 u(x,t)}{\partial x^2} \tag{1}$$

where $x,\ t \in \mathbb{R}$, $u = u(x,t)$ is the concentration (temperature in case of heat conduction) as a function of space and time, and α is the constant diffusion coefficient. The generalizations of this equation, such as the advection (or drift)–diffusion equation and different kinds of advection–diffusion–reaction equations [1], can model the transport and propagation of various particles in physical, chemical and biological systems [2]. Moreover, very similar, albeit more complicated, PDEs are used to model fluid flow through porous media [3], such as moisture [4], ground water or crude oil in underground reservoirs [5].

It is well-known that countless numerical methods have been proposed to solve Equation (1) and its generalizations. Most of them belong to the broad family of finite difference schemes (FDM) [6,7], which often means a kind of method of lines [1], where the spatial variables are discretized first to obtain an ordinary differential equation (ODE) system and then an ODE solver is employed. Nevertheless, finite element methods (FEM) are also used [8], especially if the geometry is complex. Most of these methods can be considered either as an explicit or an implicit method, and sometimes they have a mixed character [9]. If the physical properties, i.e., the coefficients of the equation, are highly non-uniform in space, the eigenvalues of the system matrix likely have a range of several orders of magnitude, which implies that one has a highly stiff problem. In this case, the so-called Courant–Friedrichs–Lewy (CFL) limit (which refers to conventional explicit methods, e.g., the Runge–Kutta or Adams–Bashforth schemes) is very small. This means that these algorithms are unstable unless the time step size is smaller than this low threshold, and therefore the simulation can be unacceptably slow.

Since implicit methods have much better stability properties, they are frequently used for these equations [10–16]. Nevertheless, they involve the solution of a system of algebraic equations at each time step whose parallelization is much less straightforward than that of explicit methods. Thus, the calculations can be time-consuming for very-large-sized and non-tridiagonal matrices, which is the case when the number of space dimensions is larger than one. In these cases, even the simplest explicit Euler time integration can be much faster than the implicit one [10,11,17]. Regrettably, the formerly rapid increase of the clock frequencies of processors disappeared about two decades ago, which reinforced the progression towards increasing parallelism in high-performance computing [18,19].

These are the main reasons why there is a growing interest in explicit and unconditionally stable algorithms [20–26]. In the last few years, our research group has developed a couple of new algorithms of this kind. In our original papers [14,27–33], we examined our new methods analytically and proved that they are indeed unconditionally stable for the linear heat equation. However, when we examined the consistency, we always found that they were only conditionally consistent, in accordance with the literature [11,12,25,34] (p. 120). The goal of this paper is to collect a large number of these methods, give their truncation error, and draw some consequences about the accuracy of the methods. Note that in textbooks and monographs, typically the standard methods, such as the explicit and implicit Euler, as well as the Crank–Nicolson method, are analyzed [35,36]. Richtmyer and Morton [12,13,37], as well as Mitchell and Griffiths [13,14,38], give the principal terms of the truncation error of the Dufort–Frankel method, but none of them do the same for the odd–even hopscotch method, albeit the latter book devotes four pages to that method. We emphasize that the truncation errors of some of the methods have never been calculated before and this might constitute the main novelty of our paper. In the case of our methods, the convergence properties were analyzed previously not via the truncation errors, but in a different way. As we will explain in detail later, this means that the numerical solutions given by the methods were compared to the *known* analytical solution of the ODE system obtained after the space discretization of the PDE (1). If possible, we will provide both kinds of errors here and compare them. We will also perform some numerical tests aiming to underpin the theoretical results. We note that it was demonstrated that these methods can be used for extremely stiff systems in more than one space dimension. They can provide fairly accurate results much faster than the widely used Runge–Kutta algorithms [14,15,39] or the professionally optimized MATLAB '`ode`' solvers. Nevertheless, in the current paper, the purpose of the numerical examples is not to test them under extreme conditions, but to illustrate the theoretical results and see which method is more competitive for the studied Equation (1). We stress that unconditional stability is not the rule but rather the exception in the huge crowd of explicit methods. It is well known, for example, that no explicit Runge–Kutta method can be A-stable [40] (p. 60). Therefore, unconditionally stable explicit methods are rarely tested against one another, and performing tests with 20 of them is absolutely unique.

The rest of the paper is organized as follows. In Section 2 we briefly present the numerical algorithms. In Section 3 we describe the different methods for the investigation of consistency and convergence, then presents the calculated error terms. The numerical experiments are presented in Section 4, while the conclusions are summarized in Section 5.

2. The Analyzed Numerical Methods

We considered the 1D interval $x \in [x_0\,,\ x_N]$ with length $L = x_N - x_0$ and constructed an equidistant spatial grid with the node coordinates $x_0\,,\ x_1\,,\ \ldots,\ x_N$, so $x_j = x_{j-1} + \Delta x$, $j = 1, \ldots, N,\ \ \Delta x = L/N$. The time variable is also discretized, so if $t \in \left[t^0,\ t^{\text{fin}}\right]$, then $t^j = t^0 + j\Delta t\,,\ \ j = 1, \ldots,\ T, T\Delta t = t^{\text{fin}} - t^0$. The mesh-ratio is defined as usual, $r = \frac{\alpha\,\Delta t}{\Delta x^2}$. In this work, only the simplest case of Equation (1) (one space-dimensional equidistant mesh and constant α coefficient) was examined, but we give the original publications where more details can be found about the application of the methods for general meshes as well. Let us start with a brief presentation of the defining formulas of the methods.

1. The UPFD method is proposed by Chen-Charpentier and Kojouharov [41] for the linear diffusion–advection–reaction PDE. It is a non-standard combination of the explicit and implicit Euler discretizations, where only the actual node is considered implicitly and the neighbors explicitly. In the case of Equation (1) it reads:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\alpha}{\Delta x^2}\left(u_{i-1}^n - 2u_i^{n+1} + u_{i+1}^n\right). \tag{2}$$

The fully explicit form is:

$$u_i^{n+1} = \frac{u_i^n + r\left(u_{i-1}^n + u_{i+1}^n\right)}{1+2r}, \tag{3}$$

2. The first scheme we invented is the so-called constant neighbor (CNe) algorithm [32,42], which has the following formula:

$$u_i^{n+1} = u_i^n \cdot e^{-2r} + \frac{u_{i-1}^n + u_{i+1}^n}{2}\left(1 - e^{-2r}\right), \tag{4}$$

3. The CpC method [31] is the organization of the CNe scheme into a two-stage algorithm. The first stage is a fractional time step with length $p\Delta t$ with the CNe formula:

$$u_i^{\mathrm{Cp}} = u_i^n e^{-2pr} + \frac{u_{i-1}^n + u_{i+1}^n}{2}\left(1 - e^{-2pr}\right).$$

These new and the old values are combined linearly to obtain the predictor values:

$$u_i^{\mathrm{pred}} = \left(1 - \frac{1}{2p}\right)u_i^n + \frac{1}{2p}u_i^{\mathrm{Cp}}.$$

These are used at the second stage, which is a full-time step size corrector step; thus, the final values are:

$$u_i^{n+1} = u_i^n \cdot e^{-2r} + \frac{u_{i-1}^{\mathrm{pred}} + u_{i+1}^{\mathrm{pred}}}{2}\left(1 - e^{-2r}\right), \tag{5}$$

4. CpCC: We now make an attempt to improve the accuracy of the CpC method by iterating one more time using the values obtained by the CpC method as predictors. So, after executing Formula (5) in the CpC method, we set $u_i^{\mathrm{pred}} = u_i^{n+1}$ and then calculate Formula (5) again.
5. The two-stage linear-neighbor (LNe or Lne2) method [32] starts with taking a full-size time step with the CNe method to obtain the predictor values $u_{\mathrm{i}}^{\mathrm{pred}}$, which are valid at the end of the current time step. Using these values, a quantity can be introduced:

$$s_i = u_{i-1}^{\mathrm{pred}} + u_{i+1}^{\mathrm{pred}} - u_{i-1}^{\mathrm{n}} - u_{i+1}^{\mathrm{n}},$$

which is proportional to the aggregated or effective slopes of the neighbors of node i. Now the corrector values of the two-stage LNe method are:

$$u_i^{n+1} = u_i^n e^{-2r} + \frac{u_{i-1}^n + u_{i+1}^n}{2}\left(1 - e^{-2r}\right) + \frac{s_i}{2}\left(1 - \frac{1 - e^{-2r}}{2r}\right). \tag{6}$$

6. The LNe3 method is a three-stage algorithm [32] whose first two stages are the same as the LNe schemes. At the end of its second stage, based on the corrector values in Equation (6), one can set $u_i^{\mathrm{pred}} = u_i^{n+1}$, recalculate the slopes s_i and then repeat (6) to obtain new corrector results. This procedure gives a three-stage scheme altogether. This algorithm is still second-order, but more accurate than LNe2.

7. The LNe4 method is a four-stage algorithm, which is obtained by repeating the procedure explained in the previous point after the calculations of the LNe3 method.
8. The CLL method [33] is very similar to the LNe3 method, but it applies fractional time steps at the first and second stages with step sizes $\Delta t_1 = p\Delta t$, $\Delta t_2 = p_2\Delta t$. Due to these, it achieves third-order convergence in the time step size, but only if $p_2 = 2/3$. If the length of the second stage is not fixed to this value, then low-order expressions with coefficient $3p_2 - 2$ appear in the truncation errors, which make the method second-order only. In this paper, we fixed $p_2 = 2/3$ and keep p as a free parameter. Therefore, the first stage applies the following formula:

$$u_i^{\mathrm{C}} = u_i^n \cdot e^{-2pr} + \frac{u_{i-1}^n + u_{i+1}^n}{2}\left(1 - e^{-2pr}\right)$$

In the second stage, a formula similar to (6) is used, but with a reduced time step size:

$$u_i^{\mathrm{CL}} = u_i^n e^{-4r/3} + \frac{u_{i-1}^n + u_{i+1}^n}{2}\left(1 - e^{-4r/3}\right) + \frac{s_i^1}{2p}\left(\frac{2}{3} - \frac{1 - e^{-4r/3}}{2r}\right), \tag{7}$$

where $s_i^1 = u_{i-1}^{\mathrm{C}} + u_{i+1}^{\mathrm{C}} - u_{i-1}^n - u_{i+1}^n$. In the third stage, a full time step is taken:

$$u_i^{n+1} = u_i^n e^{-2r} + \frac{u_{i-1}^n + u_{i+1}^n}{2}\left(1 - e^{-2r}\right) + \frac{3s_i^2}{4}\left(1 - \frac{1 - e^{-2r}}{2r}\right), \tag{8}$$

where $s_i^2 = u_{i-1}^{\mathrm{CL}} + u_{i+1}^{\mathrm{CL}} - u_{i-1}^n - u_{i+1}^n$.

9. The pseudo-implicit (PI) two-stage method was developed in our previous publication [14] (Algorithm 5 there) for the conduction–convection–radiation equation. Here, we apply it only to the pure diffusion Equation (1) with parameter $\lambda = 1$, which means that a half time step is taken to obtain the predictor values and then a full time step for the corrector values. The formulas are the following:

Stage 1: $u_i^{\mathrm{pred}} = \frac{u_i^n + \frac{r}{2}\left(u_{i-1}^n + u_{i+1}^n\right)}{1+r}$

Stage 2: $u_i^{n+1} = \frac{(1-r)u_i^n + r\left(u_{i-1}^{\mathrm{pred}} + u_{i+1}^{\mathrm{pred}}\right)}{1+r}$,

Note that the trick of the implicit and explicit treatment of the actual node and its neighbors is the same as in the UPFD method, so this PI method can actually be considered as a generalization of the UPFD scheme.

10. The alternating direction explicit (ADE) scheme is a known [26,43] but non-conventional method for which the condition of consistency is also known. We include it here for comparison purposes. In a one-dimensional equidistant mesh, one splits the calculation, i.e., first sweeps the mesh from the left to right, and then vice versa. In the case of Dirichlet boundary conditions at nodes 0 and N, one sets:

$$p_i^n = u_i^n\,,\ i = 1, \ldots N\,,\ \ p_0^{n+1} = u_0^{n+1} \text{ and } q_i^n = u_i^n\,,\ i = N-1, \ldots 1\,,\ \ q_N^{n+1} = u_N^{n+1}.$$

Then, the following equations are solved from left to right and from right to left, respectively:

$$\frac{p_i^{n+1} - p_i^n}{\Delta t} = \frac{\alpha}{\Delta x^2}\left(p_{i-1}^{n+1} - p_i^{n+1} - p_i^n + p_{i+1}^n\right) \text{ and } \frac{q_i^{n+1} - q_i^n}{\Delta t} = \frac{\alpha}{\Delta x^2}\left(q_{i-1}^n - q_i^{n+1} - q_i^n + q_{i+1}^{n+1}\right). \tag{9}$$

The explicit expressions are the following:

$$p_i^{n+1} = \frac{(1-r)p_i^n + r\left(p_{i-1}^{n+1} + p_{i+1}^n\right)}{1+r} \text{ and } q_i^{n+1} = \frac{(1-r)q_i^n + r\left(q_{i-1}^n + q_{i+1}^{n+1}\right)}{1+r}. \tag{10}$$

The final values are the simple averages of the two half-sided terms: $u_i^{n+1} = (p_i^{n+1} + q_i^{n+1})/2$. If one expresses the final value of u, one obtains:

$$(1+r)u_i^{n+1} = (1-r)u_i^n + r/2\left(u_{i-1}^n + u_{i+1}^{n+1} + u_{i-1}^{n+1} + u_{i+1}^n\right),$$

which is actually the same formula as in the Crank–Nicolson method, though the solution of this system of equations is completely different in the two methods. We note that for non-uniform meshes, the ADE method loses its fully explicit character and matrix calculations will be necessary.

11. The Dufort–Frankel (DF) method [44] (p. 313) is the textbook example of explicit and unconditionally stable methods. It is a one-stage but two-step algorithm with the formula:

$$u_i^{n+1} = \frac{(1-2r)u_i^{n-1} + 2r\left(u_{i-1}^n + u_{i+1}^n\right)}{1+2r}.$$

This method is not a self-starter, and thus u_i^1 has to be calculated from u_i^0 using another method. We use two half-sized UPFD time step for this purpose.

For the application of odd–even hopscotch methods, the space must be discretized by a special, so-called bipartite grid. The cells are labelled as odd and even, and all the nearest neighbors of the odd nodes are even and vice versa. The spatial and temporal structure of the examined algorithms in space and time are presented in Figure 1. Only one odd and one even cell for each method is shown in the figure, where the time flows from the top to the bottom. The stages are symbolized by colored rectangles, while the repeating units of blocks are surrounded by dashed blue lines. In the case of the shifted-hopscotch structure (b), for example, this unit consists of two half and three full time steps which altogether span two full time steps for each node. First, a half-sized time step (symbolized by a green box with the number '1' inside in Figure 1b) is taken for the odd cells using the initial values, then full time steps are taken strictly alternately for the even and odd cells until the end of the last time step (orange boxes), which should be halved for odd cells to reach exactly the same final time as the even nodes do. The main point is that when a new value of u_i is calculated, the latest values of the neighbors $u_{i\pm1}$ must always be used, which ensures stability and quite fast convergence at the same time.

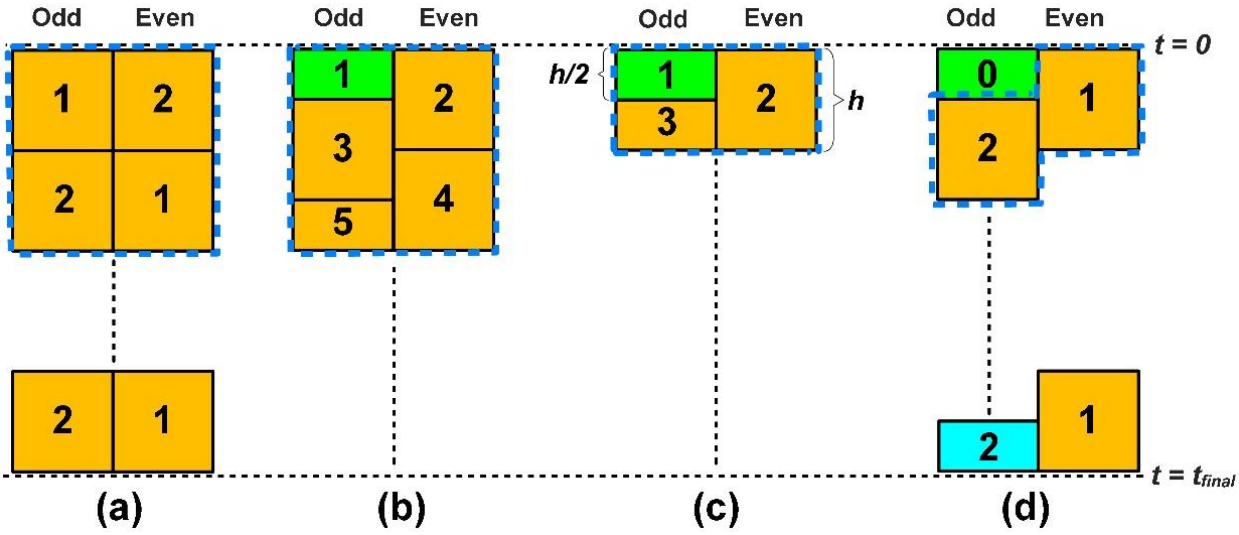

Figure 1. Space-time structure of the hopscotch-type algorithms. (**a**) Original odd–even hopscotch (OOEH), (**b**) shifted-hopscotch (SH), (**c**) asymmetric hopscotch (ASH), (**d**) leapfrog-hopscotch (LH).

12. The original odd–even hopscotch (OOEH) algorithm has been used for half a century [45]. The structure of this algorithm is shown in Figure 1a. It uses the usual

FTCS formula (based on explicit Euler time discretization) at the first stage (labels '1' in the orange boxes in the figure) and the backward time central space (BTCS) formula (implicit Euler time discretization, labels '2' in the figure) at the second stage, which are the following: FTCS: $u_i^{n+1} = (1-2r)u_i^n + r\left(u_{i-1}^n + u_{i+1}^n\right)$, implicit Euler: $u_i^{n+1} = \frac{u_i^n + r\left(u_{i-1}^{n+1} + u_{i+1}^{n+1}\right)}{1+2r}$.

13. The RH, or reversed (odd–even) hopscotch scheme [28], applies the same structure as the OOEH method, but the order of the formulas are interchanged, so the implicit formula comes first followed by the explicit one. One might think that this algorithm cannot be explicit, since the new values of the neighbors are not known when first-stage calculations start. To resolve this, the implicit formula is applied with the UPFD trick, which means the neighbors are treated not implicitly, but explicitly; thus, the first-stage formulas are:

$$u_i^{n+1} = \frac{u_i^n + r\left(u_{i-1}^n + u_{i+1}^n\right)}{1+2r}. \tag{11}$$

Since, as we mentioned, the latest values of the neighbors are always used, it is enough to change the formulas of the first and second stages in the code of the OOEH to obtain the code of the RH algorithm. We demonstrated that this RH algorithm has much smaller errors in the case of extremely stiff systems than the OOEH method.

14. The OEH-CNe (odd–even hopscotch with CNe) applies the same structure as the OOEH method again, but uses only the CNe formulas (4) appropriately.
15. The shifted-hopscotch (SH) algorithm [29] uses the theta formulas:

$$u_i^{\text{new}} = \frac{(1-2r\theta)u_i^{\text{old}} + r\left(u_{i-1}^{\text{latest}} + u_{i+1}^{\text{latest}}\right)}{1+2r(1-\theta)}, \tag{12}$$

where the 'old' and 'latest' labels refer to the appropriate time levels, which can be seen in Figure 1b. We always use the θ values which are proven to yield the most effective version (S4 algorithm in [29]). More concretely, $\theta = 0$ is used at the first stage and $\theta = 1/2$ in the second, third and fourth, while $\theta = 1$ is used at the fifth stage. When the length of the stage is halved, Δt and thus r must be divided by 2 as well in (12).

16. The shifted-hopscotch–CNe (SH-CNe) algorithm [29] applies the same space-time structure as the SH method, but uses only the CNe formulas (4) appropriately. For example, the calculation at the first stage is the following:

$$u_i^{n+1/2} = u_i^n \cdot e^{-r} + \frac{u_{i-1}^n + u_{i+1}^n}{2}\left(1 - e^{-r}\right). \tag{13}$$

17. The asymmetric-hopscotch (ASH) method [46] is similar to the SH one, but with only three stages. As is shown in Figure 1c, there are two half and one full time step size stages with the θ formula, which together span one full time step for all nodes. Based on numerical experiments, the algorithms have optimal performance if $\theta = 0$ is used at the first, $\theta = 1/2$ at the second, and $\theta = 1$ at the third stage (A1 algorithm in [46]).
18. The asymmetric-hopscotch–CNe (ASH-CNe) algorithm applies the same structure as the SH method, but uses the CNe Formulas (13) and (4) appropriately.
19. The next method is the leapfrog-hopscotch (LH) method [28]; see Figure 1d. As we will see, this is a very powerful method if one uses $\theta = 0$ at the first stage and $\theta = 1/2$ in all other stages (L2 algorithm in [30]).
20. The leapfrog-hopscotch–CNe (LH-CNe) method [30] is obtained if one replaces the θ formulas in the previous LH algorithm with the CNe formula in each stage of the LH structure (Figure 1d), appropriately.

The algorithms and some of their properties are summarized in Table 1; for example, the order of temporal convergence, marked by 'O'. All of the twenty algorithms are unconditionally stable for the linear diffusion equation, i.e., the previously mentioned CFL

limit is not valid for them. Some of them have the additional, stronger feature that the new u_i^{n+1} values are the convex combination of the initial u_j^0 values, which is indicated in the last column of the table. It implies that the maximum and minimum principles [34] (p. 87) are always fulfilled and unphysical oscillations are never produced, even for very large time step sizes. On the other hand, this favorable property restricts the speed of the convergence of these methods, so they are often the least accurate for small and medium time step sizes, as we will observe later. It means they significantly underestimate the speed of the diffusion process, especially for large and medium time step sizes.

Table 1. List of the methods and their abbreviations.

	Abbrev.	Name of the Method	Recent	Known LTE	O	Conv. Comb.
1.	UPFD	Uncond. positive finite difference	-	+	1	+
2.	CNe	Constant neighbor	+	+	1	+
3.	CpC	Two-stage iterated CNe	+	-	2	+
4.	CpCC	Three-stage iterated CNe	new	-	1	+
5.	LNe	Two-stage Linear neighbor	+	+	2	+
6.	LNe3	Three-stage iterated CNe	+	-	2	+
7.	LNe4	Four-stage iterated CNe	+	-	2	+
8.	CLL	Const.-Lin.-Lin. neighbor fract. step	+	+	3	-
9.	PI	Pseudo-implicit	+	-	2	-
10	ADE	Alternating direction explicit	-	+	2	-
11	DF	Dufort-Frankel	-	+	2	-
12	OOEH	Original odd–even hopscotch	-	?	2	-
13	RH	Reversed hopscotch	+	-	2	-
14	OEH-CNe	OEH structure with CNe formulas	+	-	2	+
15	SH	Shifted hopscotch	+	-	2	-
16	SH-CNe	SH structure with CNe formulas	+	-	2	+
17	ASH	Asymmetric hopscotch	+	-	2	-
18	ASH-CNe	ASH structure with CNe formulas	+	-	2	+
19	LH	Leapfrog-hopscotch	+	-	2	-
20	LH-CNe	LH structure with CNe formulas	+	-	2	+

All methods containing the CNe or LNe formula, as well as the LH, RH, SH, ASH and the PI methods, have been recently constructed by our research group, which is indicated after the name of the method in the Table. In our original papers, some analytical proofs, verifications using analytical solutions and numerical tests to compare performances are presented. However, in most cases, the truncation error has not been calculated (marked by a '-' sign in the "known LTE" column).

3. Analytical Results

The most widespread definition of the truncation error is the difference between the discretized and the exact *equation* [47] (p. 31). For example, in the case of Equation (1) and the forward time central space (FTCS) method,

$$\tau = D_t^+ u - D_x^2 u, \tag{14}$$

where

$$D_t^+ u = \frac{u\left(x_i^n,\ t^n + \Delta t\right) - u\left(x_i^n,\ t^n\right)}{\Delta t} \text{ and } D_x^2 u = \frac{u\left(x_i^n - \Delta x,\ t^n\right) - 2u\left(x_i^n,\ t^n\right) + u\left(x_i^n + \Delta x,\ t^n\right)}{\Delta x^2}$$

are the first-order forward and second-order central difference operators for the time and the space variable, respectively. The Taylor-series expansions of u are inserted into Equation (14). The function u is supposed to be the exact solution, thus the substitutions $u_t = \alpha u_{(2x)}$, $u_{(2t)} = \left(\alpha u_{(2x)}\right)_t = \alpha^2 u_{(4x)}$, $u_{(3t)} = \alpha^3 u_{(6x)}$ and so on can be used. The discretization error of D_x^2 is:

$$\tau_0 = \varepsilon_0 = -\frac{\alpha}{12}u_{(4x)}\Delta x^2 - \frac{\alpha}{360}u_{(6x)}\Delta x^4 - \frac{\alpha}{20,160}u_{(8x)}\Delta x^6 + \text{h. o. t.}, \tag{15}$$

The discretization error of the D_t^+ operator depends only on Δt, thus the space- and time-dependent terms in the truncation error can be clearly separated. This is not true in our cases, where there are terms containing the ratio of Δt and Δx. Moreover, for multistage methods, an expression with a similar form as (15) cannot be obtained, or the method is not clearly a FDM at all, since the differentiation with respect to time is not approximated by a finite difference operator. Nevertheless, all of the examined methods are explicit, and thus they can almost always be expressed in the fully explicit form $u_i^{n+1} = f\left(u_i^n,\ u_{i\pm1}^n, \ldots\right)$, where there are no quantities at the new n+1 time level present in the right-hand side, but the f function has the parameter r. Note that in the numerical codes, these forms are used.

The Taylor series expansion can be performed at this level as well [30]. In this way, one obtains the difference between the numerical and the exact *solution*. Let us denote the truncation error obtained by choosing this way by ε. To present a simple example, in the case of the UPFD method, τ is obtained when the expansions are applied to Equation (2), while ε is obtained when it is applied to Equation (3). In the latter case, the $(1+2r)^{-1}$ coefficient must also be expanded and the result will be different from τ. One of the sources of the difference is that when Equation (3) is obtained from Equation (2), we multiplied it by Δt, which is quite common; see p. 20 in [37]. Therefore, ε must be divided by Δt to make the two kinds of errors comparable. In fact, after this division, both τ and ε give an estimation for the global error [30]. However, we will see that τ and ε are usually still different and it is a nontrivial question which one reflects more the properties of the numerical scheme, i.e., which one is more useful for practical purposes.

The term (15) is present in case of all the examined methods, both in τ and in ε. We note that the higher order terms (h. o. t.) will be usually omitted for the sake of brevity. The τ and ε errors have been calculated in most cases by both the first and the second author completely independently.

If one follows the so-called method of lines, where Equation (1) is discretized only spatially using the central difference formula, one obtains the ODE system:

$$\frac{d\vec{u}}{dt} = M\vec{u}\ ,\ \vec{u}(\mathrm{t}=0) = \vec{u}^0, \tag{16}$$

where $\vec{u}^0$ is the arbitrary initial vector. The elements of M can be given as:

$$m_{ii} = -\frac{2\alpha}{\Delta x^2}\ , \quad m_{i,i+1} = m_{i,i-1} = \frac{\alpha}{\Delta x^2}\ \ (1 < i < N),$$

while the first and last row are determined by the boundary condition. The exact solution of (16) at the end of a time step is the following:

$$\vec{u}^{n+1} = \mathrm{e}^{M\Delta t}\vec{u}^n = \left(1 + M\Delta t + M^2\frac{\Delta t^2}{2} + M^3\frac{\Delta t^3}{3!} + O\left(\Delta t^4\right)\right)\vec{u}^n.$$

Here, as well as in the case of the e^{-2r}-type terms in the CNe, LNe, etc., methods, the usual series expansion $e^{\pm x} = 1 \pm x + x^2/2!\ \pm x^3/3!\ + O\left(x^4\right)$ is used up to the appropriate order. After some simple algebraic calculations, we obtained the expressions of a general element of the exact solution after one and two time steps:

$$u_i^{n+1} = \frac{r^2}{2}u_{i\pm2}^n + \left(r - 2r^2\right)u_{i\pm1}^n + \left(1 - 2r + 3r^2\right)u_i^n + O\left(r^3\right), \tag{17}$$

and

$$u_i^{n+2} = 2r^2u_{i\pm2}^n + \left(2r - 8r^2\right)u_{i\pm1}^n + \left(1 - 4r + 12r^2\right)u_i^n + O\left(r^3\right), \tag{18}$$

respectively. The simple notation $u^n_{i\pm j} = u^n_{i+j} + u^n_{i-j}$, $i, j \in \mathbb{N}^+$ has been introduced for brevity. Formulas (17) and (18) are valid if we are far enough from the boundaries. To ensure this, we always examine the middle element of a large enough matrix. A similar expression can be obtained for each numerical method. More precisely, for the hopscotch methods, we have different formulas for the odd and the even nodes. If the smallest repeating unit spans one or two time steps, then Equation (17) or (18) must be used, respectively. The local error is the difference between the analytical and the numerical expressions, which is divided by Δt again to obtain the global error, which is denoted by δ. We say that a method is, e.g., second-order if the zeroth- and the first-order terms vanish in δ, and the second-order terms have the lowest order. This kind of investigation has been performed for some of our methods in the original publications, but only in order to prove the order of the method. The first non-vanishing terms are given for the first time here, which is also a novelty in the current paper.

The calculated errors are presented in the Appendix A.

4. Numerical Experiments

In this section, we reproduce nontrivial analytical solutions in one space dimension. The Dirichlet boundary conditions at the two end points of the interval and the initial $u^0(x,\ t)$ function were obtained by simply substituting the appropriate x and t values into the analytical solution. The MATLAB R2019b software was used for all numerical calculations. We calculated the eigenvalues of the M matrix in Equation (16) for this problem. Using these, the stiffness ratio and the largest time step size where the standard FTCS method is stable can be determined as usual [33].

To measure the accuracy, the widely used L_∞ error is calculated, which compares the exact value u_i^{analytic} and the result u_i^{num} obtained by the actual numerical method at the final time t^{fin}:

$$Error = \max_{1\le i\le N} \left| u_{\text{i}}^{\text{analytic}}(t^{\text{fin}}) - u_{\text{i}}^{\text{num}}(t^{\text{fin}}) \right|. \tag{19}$$

Experiment 1. We are going to reproduce the following reference solution, which is a quite recent [48] analytical solution:

$$u^{\text{analytic}}(x,t) = \frac{x}{t^{7/2}} \left(1 - \frac{x^2}{3\alpha t} + \frac{x^4}{60\alpha^2 t^2} \right) e^{-\frac{x^2}{4\alpha t}}. \tag{20}$$

We considered the domain with the boundaries $x_0 = -5$, $x_N = 5$, $t^0 = 0.3$, $t^{\text{fin}} = 0.56$, while $\alpha = 1$. We set $\Delta x = 0.02$, which implies that the stiffness ratio is $1.01 \cdot 10^5$, while $\Delta t^{\text{FTCS}}_{\text{MAX}} = 3.9 \cdot 10^{-4}$. We examined the error defined in (19) as a function of the time step size Δt. First, the errors were calculated for a very large Δt, then with decreased time step sizes until small error values were obtained. In Figures 2 and 3, the errors as a function of the time step size are shown in log-log diagrams. Since we used a fixed-space step size and decreased only the time step size, the errors cannot go to zero, but only to the residual error due to space discretization, which is given by (15). This can be seen in the bottom left of the figures. One can notice that although the CLL method is third-order and it is much more accurate than the LNe method, it converges more slowly than the hopscotch-type methods. In accordance with some experiments in our previous papers, the LH method was the most accurate.

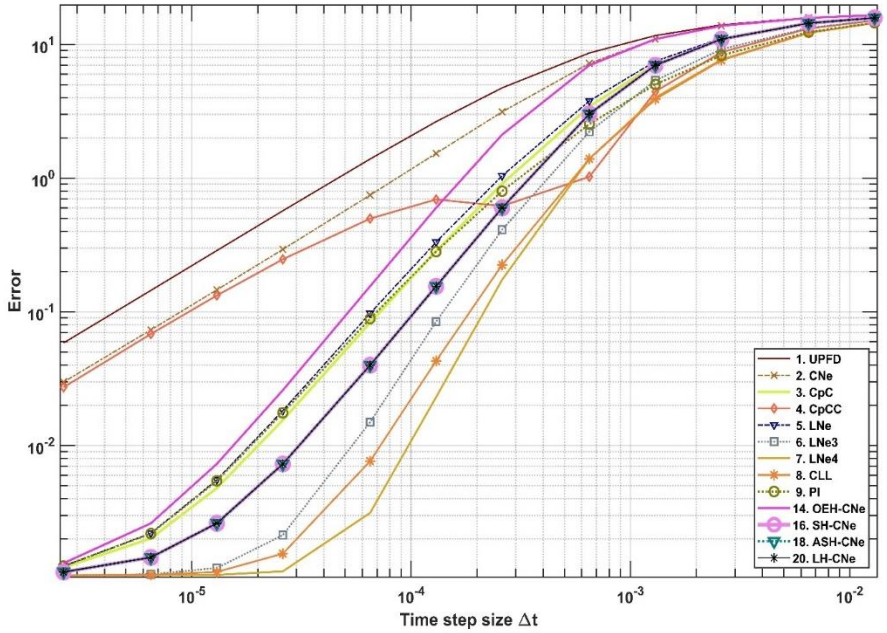

Figure 2. Errors as a function of the time step size for the slowly converging methods.

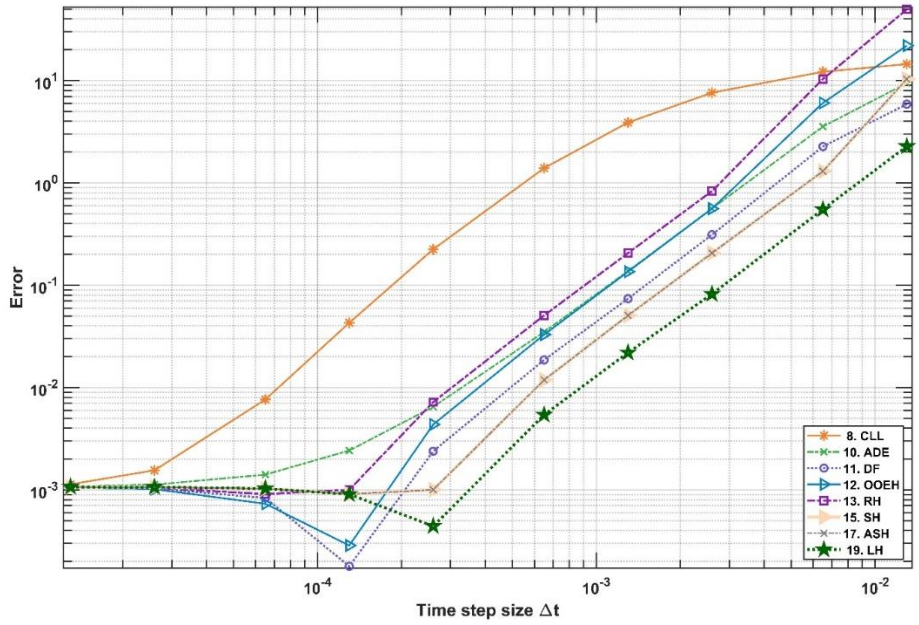

Figure 3. The errors as a function of the time step size for the quickly converging methods, as well as the CLL method in the case of Experiment 1. The error curves reach the residual error τ_0 on the left side of the figure.

Figures 4–6 show how the numerical solutions approximate the analytical solution as the time step size is decreased. In the case of the one-stage CNe method and the three-stage CLL methods, the curves are smooth without unphysical oscillations, while the ASH method produces significant oscillations for larger time step sizes. On the other hand, the CNe method converges very slowly and the CLL converges at a medium rate, while the ASH converges much faster. Some investigation of the behaviour of the errors are presented in our previous paper [39], as well for fixed time step size and space-dependent diffusion coefficient.

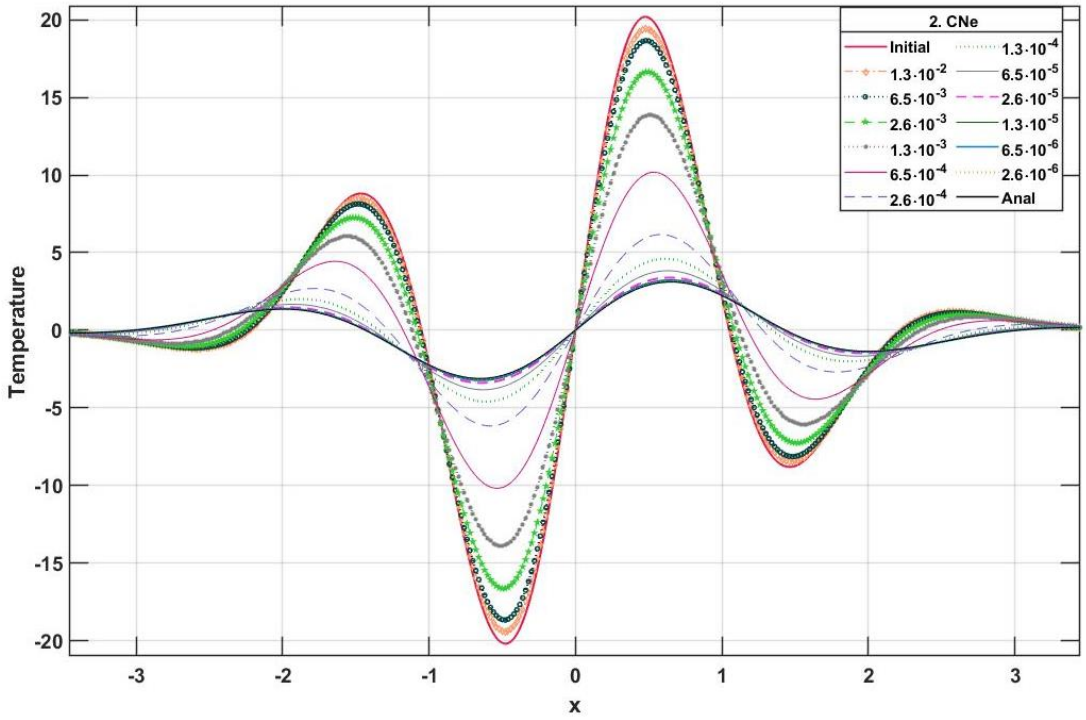

Figure 4. The numerically calculated temperatures u for the three-stage CNe method as a function of the space variable x for different time step sizes in Experiment 1. The red and the black solid lines are for the initial and the analytically calculated temperatures, respectively.

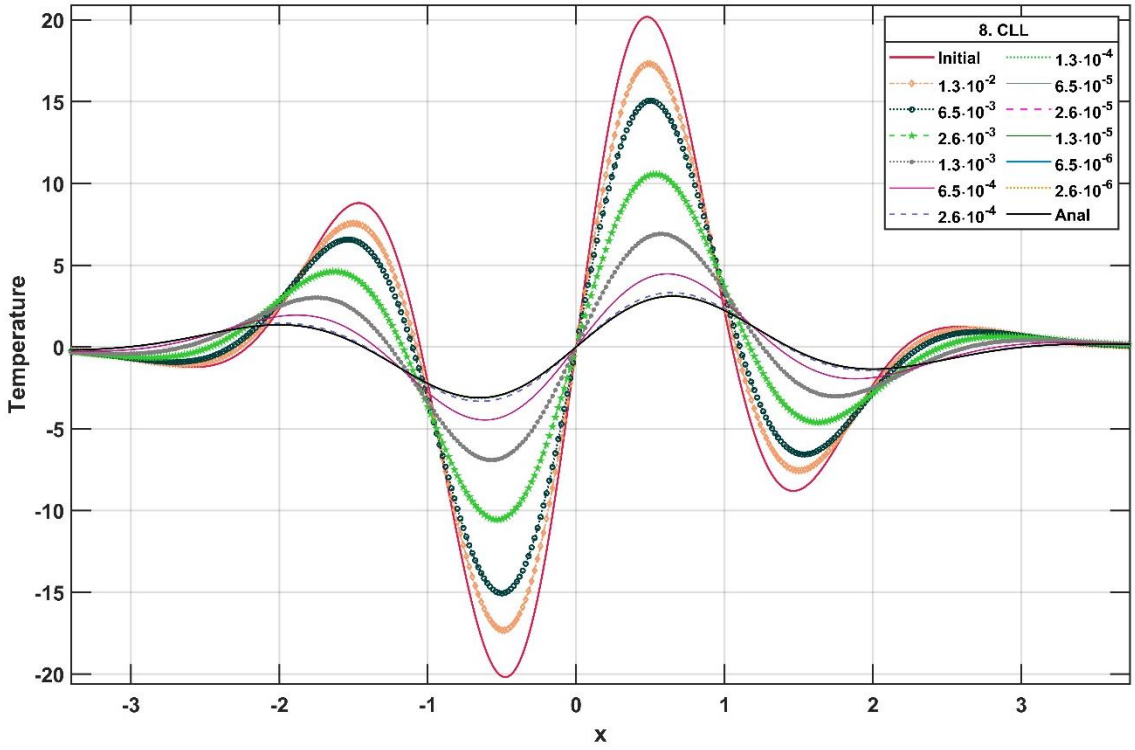

Figure 5. The numerically calculated temperatures u for the three-stage CLL method as a function of the space variable x for different time step sizes in Experiment 1. The red and the black solid lines are for the initial and the analytically calculated temperatures, respectively.

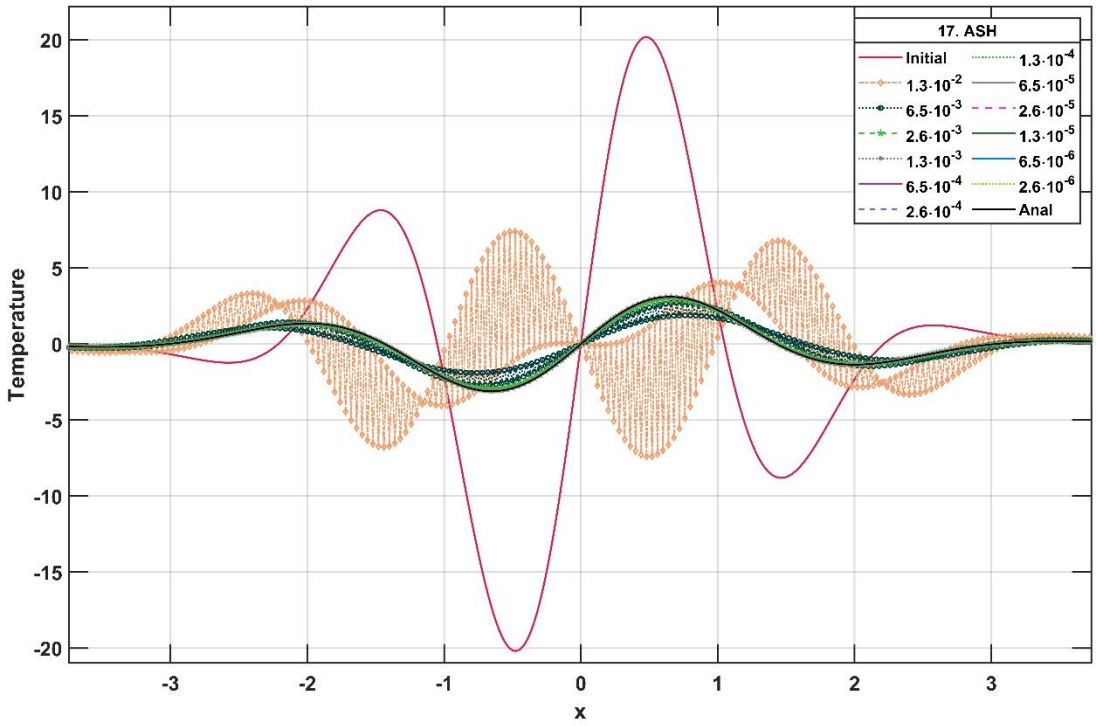

Figure 6. The numerically calculated temperatures u for the three-stage ASH method as a function of the space variable x for different time step sizes in Experiment 1. The red and the black solid lines are for the initial and the analytically calculated temperatures, respectively.

Experiment 2. The solution of the semi-discretized Equation (16) was also calculated numerically by the ode15s MATLAB routine with a very stringent tolerance. The spatial domain was extended to $x_0 = -10$, $x_N = 10$, but all other circumstances of Experiment 1, including the analytical solution (20) and the spatial step size $\Delta x = 0.02$, still hold. In Figure 7, we present the errors for six methods when not only the analytical, but the numerical reference was used for the error calculation in Formula (19). One can see that if the numerical reference is used, the residual errors disappear, and for very small time step sizes, the error of the third-order CLL method goes below the errors of the second-order methods.

Experiment 3. In this subsection, $x_0 = -5$, $x_N = 5$, $t^0 = 0.3$, $t^{\text{fin}} = 0.5$. The space step size Δx varied, but all other parameters of Experiment 1 were preserved. For comparison purposes, we used the very common implicit (Euler) method:

$$u_i^{n+1} = u_i^n + r\left(u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}\right),$$

which was implemented by matrix inversion.

In our previous publications, we measured the running times and conveyed plenty of examples where the traditional methods were outperformed by the explicit and unconditionally stable ones. Therefore, our main purpose with these standard schemes was not the comparison of the performances, but to compare the behavior of the errors of the 20 conditionally consistent schemes with an unconditionally consistent case, so that we could explain properly the connection between the truncation errors and the observed errors.

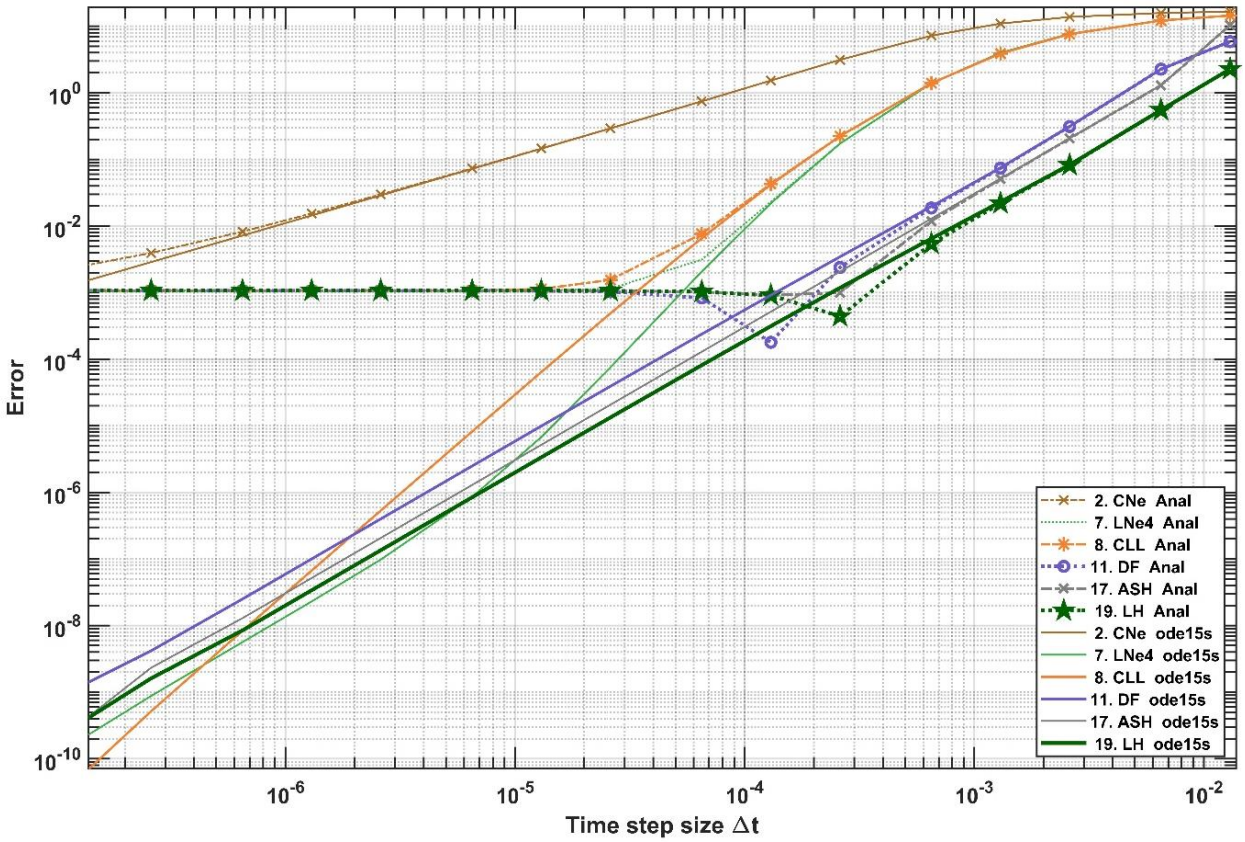

Figure 7. The errors as a function of the time step size for six methods in Experiment 2 when not only the analytical but a numerical reference solution was used to calculate the errors.

First, the space step is considered as a parameter with six different values of Δx (five in the case of the implicit method, since it has much larger running times). The error-curves are presented for the first-order UPFD and implicit methods in Figure 8, while Figure 9 contains the results for the second-order LNe and LH methods. The curves of the UPFD, LNe and LH method cross one another, because if the space step size is smaller, the convergence is slower due to the $\Delta t^2 \Delta x^{-4}$ type terms in the truncation error. It is easy to see that for a fixed time step size, decreasing space step sizes does not yield better accuracy. In the case of the implicit method, however, the error decreases if *either* the time *or* the space step is decreased. In the case of the UPFD and the LNe method, the horizontal distance between the neighboring curves (belonging to a certain Δx and its half $\Delta x/2$) is a factor of 4, while it is only a factor of 2 for the LH method. This can be explained by the truncation error terms of $\Delta t \Delta x^{-2}$ of the UPFD and $\Delta t^2 \Delta x^{-4}$ of LNe schemes, and, on the other hand, the term $\Delta t^2 \Delta x^{-2}$ of the LH scheme. If the exponent in the denominator is twice as large as in the numerator, then Δt must be divided by four to counterbalance the division of Δx by two in order to keep the error constant.

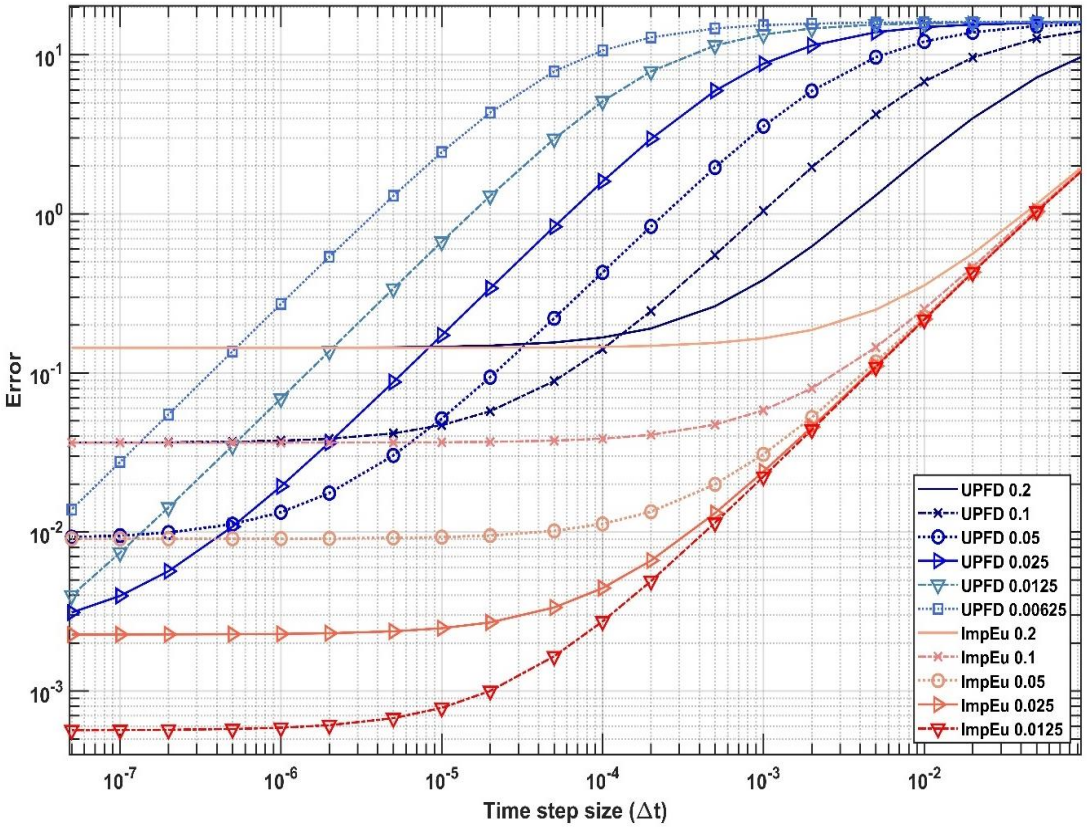

Figure 8. Errors as a function of the time step size for the UPFD and the implicit Euler (ImpEu) methods with different space step sizes in Experiment 3. Δx changes from 0.2 to 0.00625 and to 0.0125 for the UPFD and the implicit method, respectively.

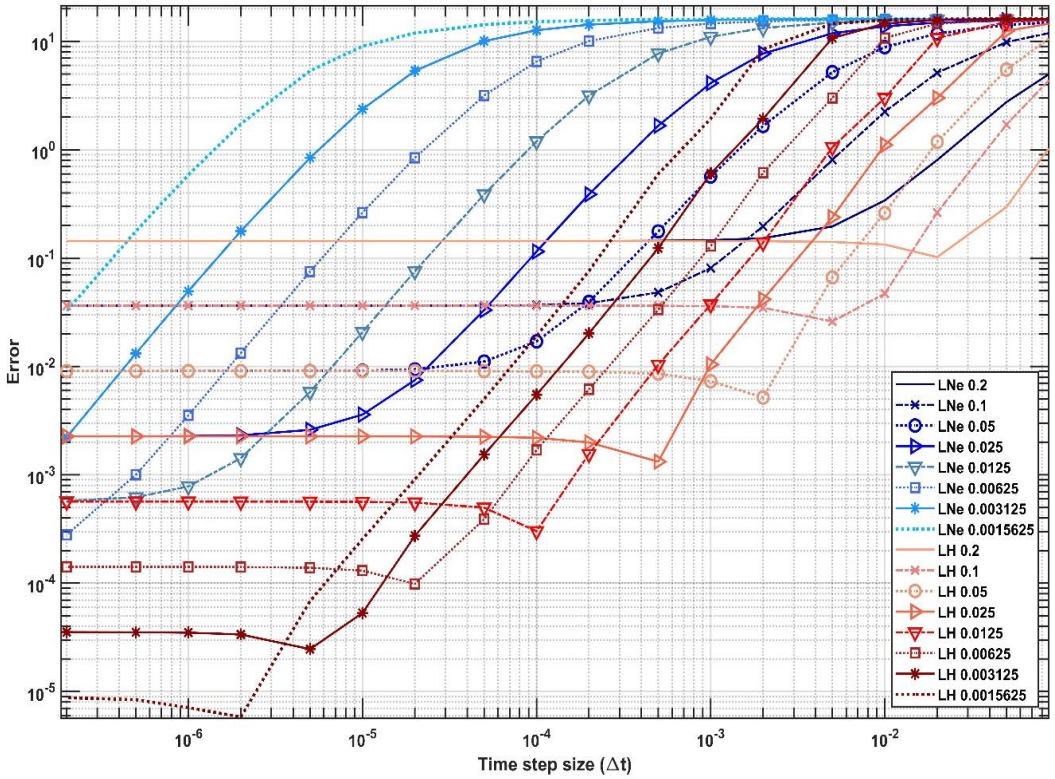

Figure 9. Errors as a function of the time step size for the LNe and the LH methods with different space step sizes in Experiment 3.

Experiment 4. We conducted a case study considering $\Delta t \to 0$ when $\Delta t/(\Delta x)^2 = 0.2$ was fixed. All parameters were the same as in Experiment 3. The error curves can be seen in Figure 10 for all the 20 methods. Two groups of the methods can be clearly distinguished. The errors of the first group tended to constant values, since their truncation errors contained terms where the exponent of the space step size in the denominator was twice as large as that of the time step size in the numerator. These algorithms were not convergent under these circumstances. On the other hand, the errors of the second group tended to zero, which means they did not have truncation error terms of $\Delta t^n \Delta x^{-2n}$. The results of this numerical investigation are consistent with the analytically calculated truncation error expressions which were detailed in Section 2. The exceptions are the SH and the ASH methods, for which we were not able to calculate the τ errors, only the ε ones (see Equations (A1) and (A2)), which contain the critical $\Delta t^n \Delta x^{-2n}$ terms. In these cases, the ε errors do not properly reflect the observed properties of these methods from this point of view; thus, we think that further investigations would be necessary.

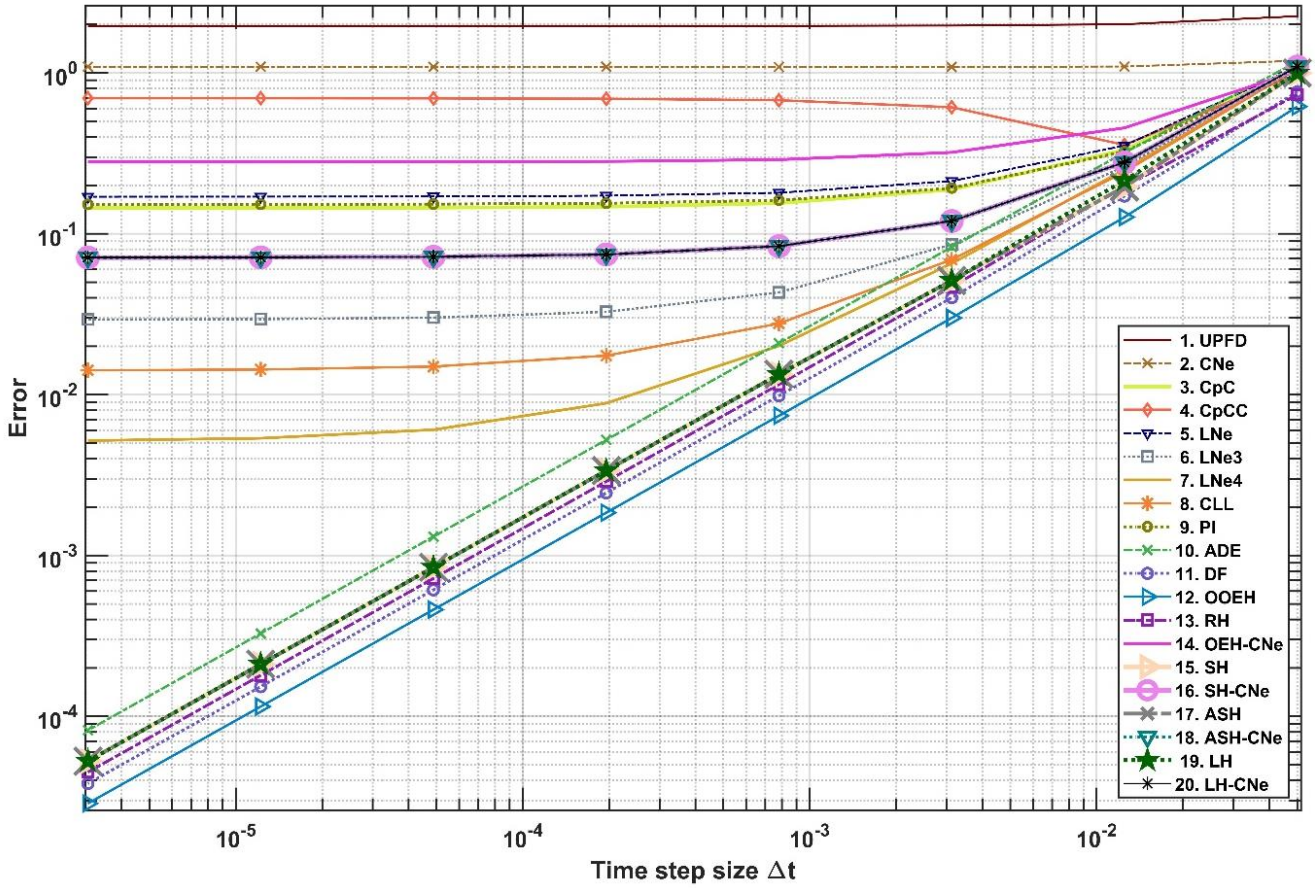

Figure 10. The maximum errors as a function of the time step size in Experiment 4 for a constant $r = \alpha\, \Delta t/\Delta x^2$. This means that the space step size also decreases from the right to the left side of the figure.

Experiment 5. Here, everything was the same as in Experiment 1, with the exception that the diffusion coefficient and the final time were increased to $\alpha = 2$ and $t^{\text{fin}} = 5$, while $\Delta x = 0.01$. The error-curves are presented in Figures 11 and 12 for the slowly and the quickly converging methods. We also examined how the maximum errors developed as time elapsed for a given medium-sized time step size. Figures 13 and 14 show that after a sharp increase, the accumulated errors themselves diffuse away and the errors tend slowly toward zero, after some small fluctuations in case of a few quickly converging methods.

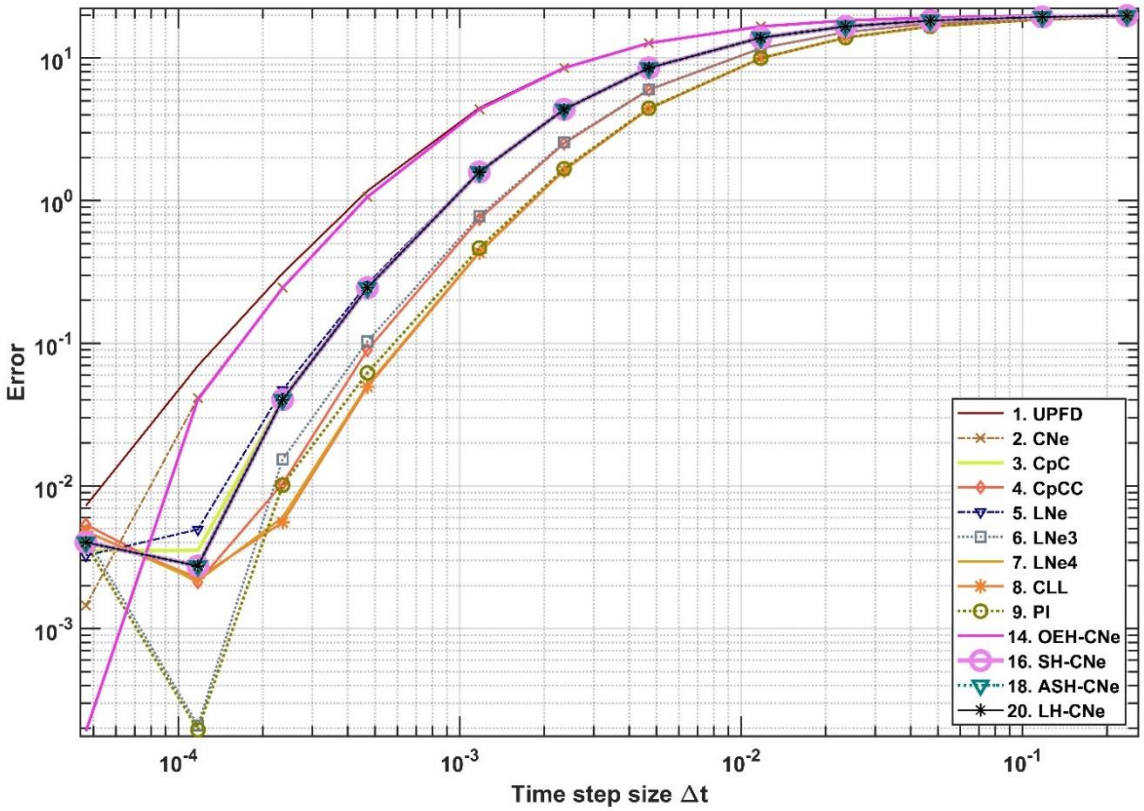

Figure 11. The maximum errors as a function of the time step size Δt for the slowly converging methods in the case of Experiment 5.

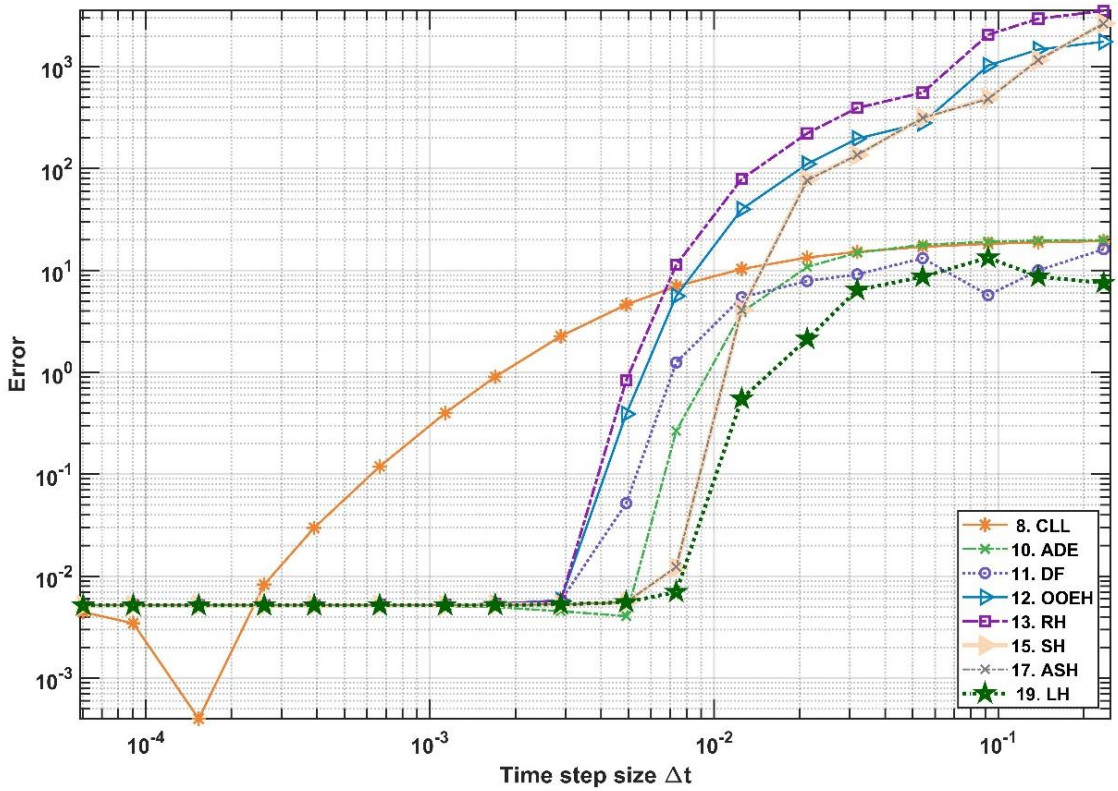

Figure 12. The errors as a function of the time step size for the quickly converging methods, as well as the CLL method, in the case of Experiment 5.

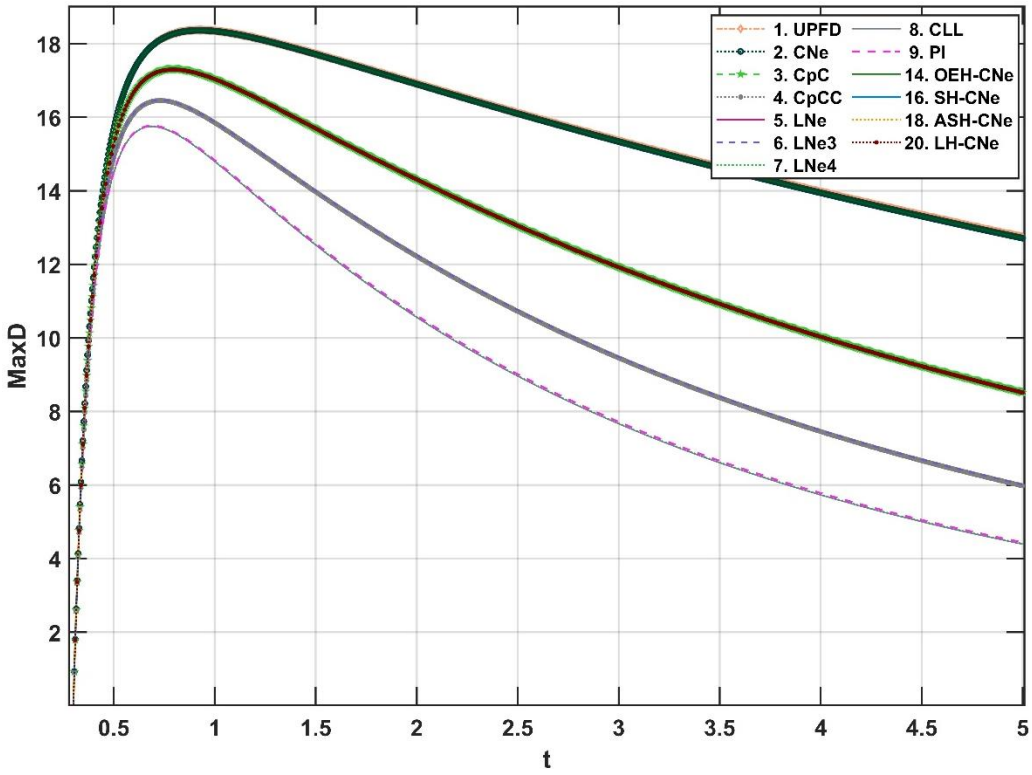

Figure 13. The evolution of the maximum errors as a function of the physical time t for the slowly converging methods in Experiment 5. The used time step size is $\Delta t = 4.7 \cdot 10^{-3}$.

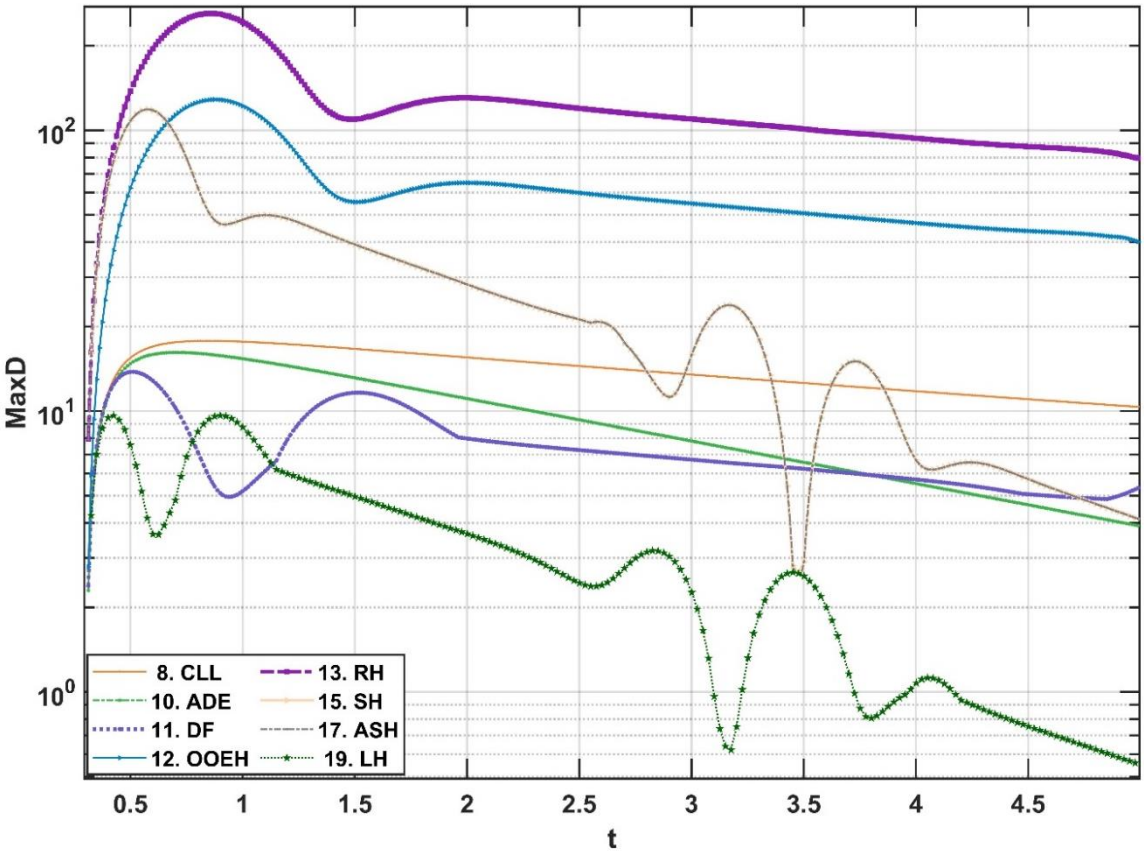

Figure 14. The evolution of the maximum errors as a function of the physical time t in the case of fast converging methods in Experiment 5. The used time step size is rather large: $\Delta t = 1.25 \cdot 10^{-2}$.

Experiment 6. We next considered the generalized diffusion equation where the diffusion constant has a power-law time dependence:

$$\frac{\partial u(x,t)}{\partial t} = \hat{\alpha}\, t^n \frac{\partial^2 u(x,t)}{\partial x^2}, \tag{21}$$

where $\hat{\alpha}$ is considered a constant. The function:

$$u(x,t) = \frac{x}{t^{b-a}} e^{-\frac{(1+n)x^2}{4\hat{\alpha}t^{2b}}} K_M\left[\frac{1+n-a}{1+n}, \quad \frac{3}{2}, \quad \frac{(1+n)x^2}{4\hat{\alpha}t^{2b}}\right]. \tag{22}$$

is a very recent [30] analytical solution of Equation (21) if $b = (1+n)/2$ and K_M is the Kummer–M function, which was calculated here by the hypergeom function of MATLAB. We set the parameter values $\hat{\alpha} = 3.1$, $a = 39.2$, $n = 10.4$, $x_0 = -2$, $x_N = 2$, $t^0 = 0.9$, $t^{\text{fin}} = 0.94$ and $\Delta x = 0.01$. Figures 15 and 16 show the maximum errors as a function of the time step size for the slowly and quickly converging schemes, respectively. Figure 17 shows the initial temperatures u^0, the analytical solution u^{anal} and some numerical results, which were obtained with four different methods using different time step sizes.

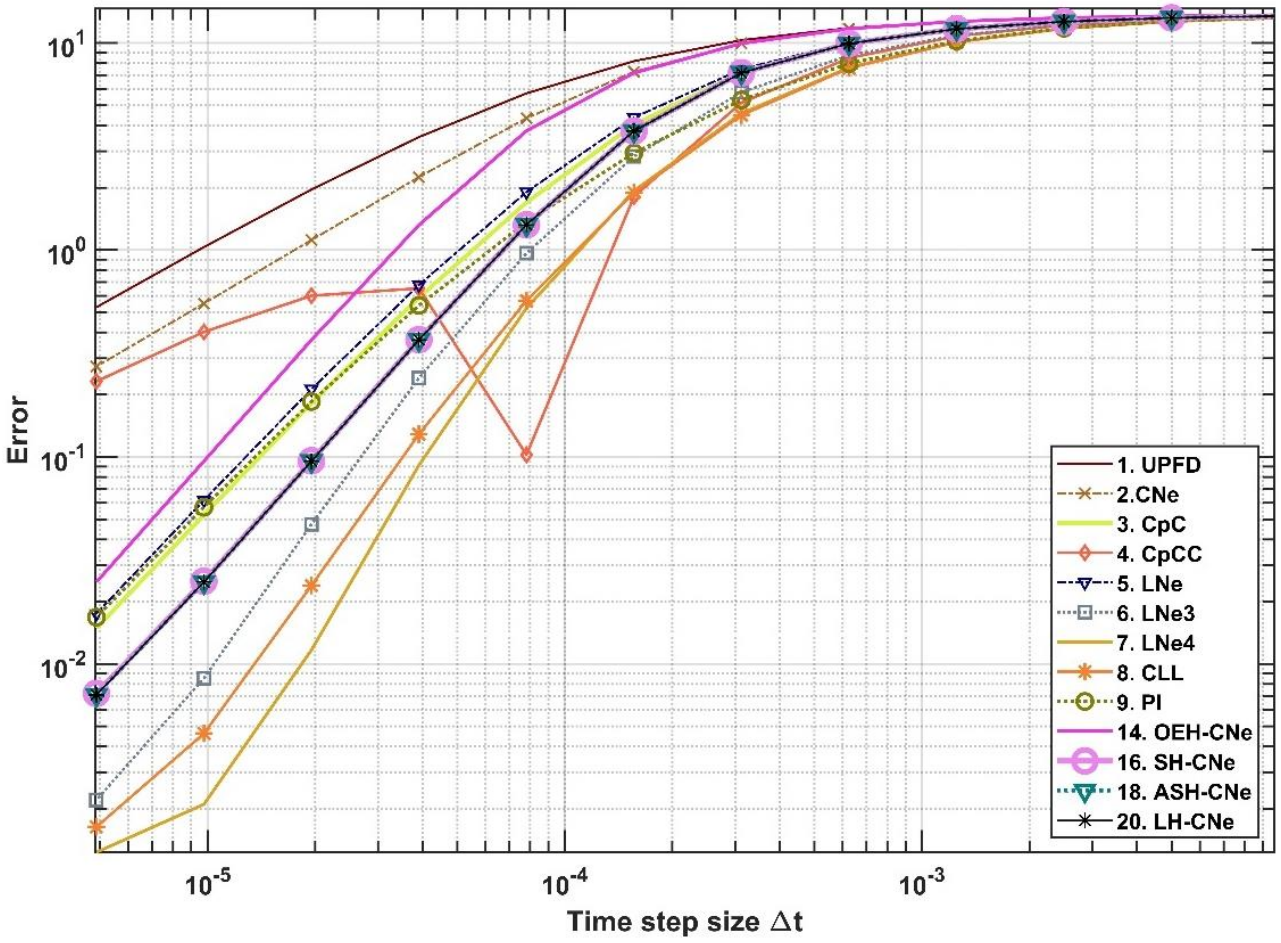

Figure 15. The maximum errors as a function of the time step size Δt for the slowly converging methods in the case of Experiment 6, where the diffusion constant strongly depends on the time.

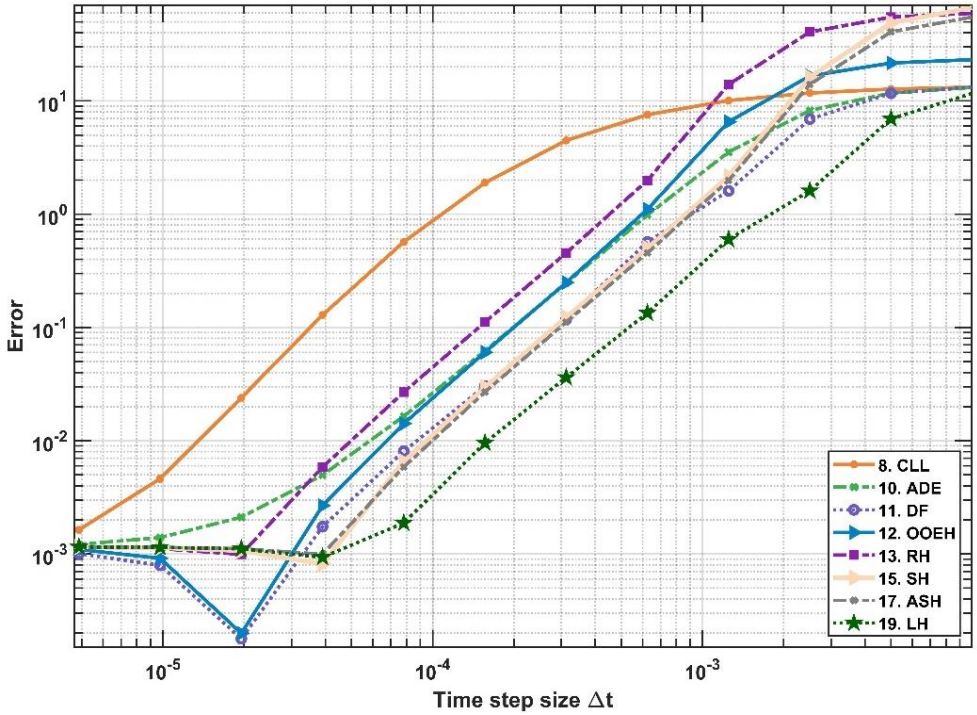

Figure 16. The maximum errors as a function of the time step size Δt for the fast convergent numerical schemes in the case of Experiment 6.

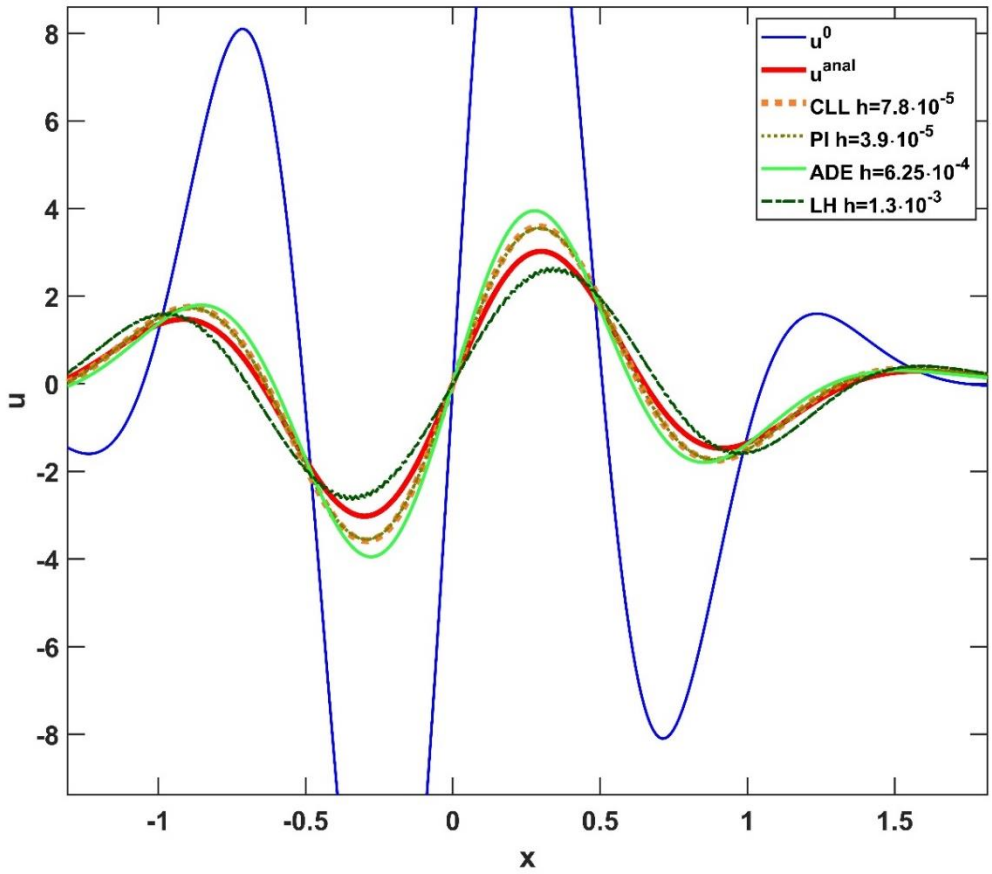

Figure 17. The temperature u as a function of the space coordinate x in Experiment 6. This figure shows the initial temperatures u^0, the analytical solution u^{anal} and the numerical results, which were obtained with four different methods using different time step sizes.

5. Discussion and Summary

We have studied 20 numerical methods for the non-steady-state linear diffusion equation. All of the algorithms are unconditionally stable explicit methods; 16 of them have been recently constructed by our research group. We have calculated the truncation errors for each algorithm, which was unknown until now in several cases. Moreover, we have considered another method of investigation, in which the numerical solution is compared to the concrete analytical solution of the ODE system, which is obtained by the spatial discretization of the PDE. We think that this latter method produces errors which explain the observed numerical order of the algorithms more clearly if the spatial discretization is kept fixed and only the time step size is decreased, especially when the stiffness ratio is high.

We have reproduced a recent analytical solution by the studied 20 numerical methods in six numerical experiments. We have made attempts to explain the observed numerical errors based on the analytically calculated truncation errors. We emphasize again that although these algorithms are only conditionally consistent, they give accurate results for time step sizes orders of magnitude larger than the standard explicit methods, and faster (for large systems) than the implicit methods.

Author Contributions: Conceptualization, supervision and resources, E.K.; methodology, E.K., Á.N. and J.M.; analytical investigation, J.M. and Á.N.; software, numerical investigation and visualization, Á.N.; writing—original draft preparation, E.K. and J.M.; writing—review and editing, Á.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available on request from the authors.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Here, we present those errors which we were able to calculate.

1. UPFD

$$\tau_{\mathrm{UPDF}} = \tau_0 + 2\alpha^2\frac{u_{(2x)}}{\Delta x^2}\Delta t + \frac{\alpha^2}{2}u_{(4x)}\Delta t + \alpha^3 u_{(4x)}\frac{\Delta t^2}{\Delta x^2} + \frac{\alpha^3}{6}u_{(6x)}\Delta t^2$$

$$\varepsilon_{\mathrm{UPDF}} = \varepsilon_0 + 2\alpha^2\frac{u_{(2x)}}{\Delta x^2}\Delta t + \frac{2}{3}\alpha^2 u_{(4x)}\Delta t + \frac{\alpha^2}{180}u_{(6x)}\Delta x^2\Delta t - 4\alpha^3 u_{(2x)}\frac{\Delta t^2}{\Delta x^4} - \frac{\alpha^3}{3}u_{(4x)}\frac{\Delta t^2}{\Delta x^2} + \frac{7}{45}\alpha^3 u_{(6x)}\Delta t^2$$

$$\delta_{\mathrm{UPFD}} = -\frac{\beta^2}{2}\left(u^n_{i\pm1} - 2u^n_i\right)\Delta t + \frac{5\beta^3}{2}\left(u^n_{i\pm1} - 2u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right).$$

2. CNe

$$\tau_{CNe} = \varepsilon_{CNe} = \varepsilon_0 + \alpha^2\frac{u_{(2x)}}{\Delta x^2}\Delta t + \frac{7}{12}\alpha^2 u_{(4x)}\Delta t + \frac{\alpha^2}{360}u_{(6x)}\Delta x^2\Delta t - \frac{2}{3}\alpha^3 u_{(2x)}\frac{\Delta t^2}{\Delta x^4} - \frac{\alpha^3}{18}u_{(4x)}\frac{\Delta t^2}{\Delta x^2} + \frac{89}{540}\alpha^3 u_{(6x)}\Delta t^2$$

$$\delta_{\mathrm{CNe}} = \frac{\beta^2}{2}\left(u^n_{i\pm1} - 2u^n_i\right)\Delta t - \frac{5\beta^3}{2}\left(u^n_{i\pm1} - 2u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right)$$

3. CpC. If the parameter p is kept, we have:

$$\varepsilon_{\mathrm{CpC}} = \varepsilon_0 + \frac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t + \frac{\alpha^3\left(360(1+3p)u_{(2x)} + 30(19+21p)\Delta x^2 u_{(4x)} + (271+93p)\Delta x^4 u_{(6x)}\right)}{1080\Delta x^4}\Delta t^2.$$

One can see that the error decreases with decreasing p. However, the stability analysis in our original paper showed that p must be at least $\frac{1}{2}$ for unconditional stability, thus this is the optimal choice for p. With this, we have:

$$\varepsilon_{C\frac{1}{2}C} = \varepsilon_0 + \frac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t + \frac{5}{6}\alpha^3 u_{(2x)}\frac{\Delta t^2}{\Delta x^4} + \frac{59}{72}\alpha^3 u_{(4x)}\frac{\Delta t^2}{\Delta x^2} + \frac{127}{432}\alpha^3 u_{(6x)}\Delta t^2$$

$$\delta_{C\frac{1}{2}C} = \frac{\beta^3}{12}\left(u^n_{i\pm2} + u^n_{i\pm1} - 2u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right)$$

4. CpCC: Let us keep p as a free parameter. However, during the calculations, p cancels out from all the considered terms and we obtain:

$$\varepsilon_{CpCC} = \varepsilon_0 + \alpha^2\frac{u_{(2x)}}{\Delta x^2}\Delta t + \frac{7}{12}\alpha^2 u_{(4x)}\Delta t + \frac{\alpha^2}{360}u_{(6x)}\Delta x^2\Delta t - \frac{2}{3}\alpha^3 u_{(2x)}\frac{\Delta t^2}{\Delta x^4} - \frac{\alpha^3}{18}u_{(4x)}\frac{\Delta t^2}{\Delta x^2} + \frac{89}{540}\alpha^3 u_{(6x)}\Delta t^2$$

$$\delta_{CpCC} = \frac{1}{2}\beta^2\left(u^n_{i\pm2} - 2u^n_{i\pm1} + 2u^n_i\right)\Delta t^2 + \frac{1}{3}\beta^3\left[u^n_{i\pm3} - 6u^n_{i\pm2} + 11u^n_{i\pm1} - 12u^n_i\right]\Delta t^3 + O\left(h^4\right).$$

One can see that the extra iteration does not improve but deteriorates the accuracy. That is why we need the linear-neighbor approximation for further improvement.

5. LNe:

$$\varepsilon_{LNe} = \varepsilon_0 - \tfrac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t - \tfrac{\alpha^2}{160}u_{(8x)}\Delta x^4\Delta t + \alpha^3 u_{(2x)}\tfrac{\Delta t^2}{\Delta x^4} + \tfrac{11}{12}\alpha^3 u_{(4x)}\tfrac{\Delta t^2}{\Delta x^2} + \tfrac{37}{120}\alpha^3 u_{(6x)}\Delta t^2$$
$$+\tfrac{211}{20{,}160}\alpha^3 u_{(8x)}\Delta x^2\Delta t^2 - \tfrac{4}{3}\alpha^4 u_{(2x)}\tfrac{\Delta t^3}{\Delta x^6} - \tfrac{17}{18}\alpha^4 u_{(4x)}\tfrac{\Delta t^3}{\Delta x^4} - \tfrac{77}{540}\alpha^4 u_{(6x)}\tfrac{\Delta t^3}{\Delta x^2} + \tfrac{943}{30{,}240}\alpha^4 u_{(8x)}\Delta t^3$$

$$\delta_{LNe} = -\frac{\beta^3}{6}\left(u^n_{i\pm1} - 2u^n_i\right)\Delta t^2 + \frac{7\beta^4}{24}\left(u^n_{i\pm1} - 2u^n_i\right)\Delta t^3 + O\left(\Delta t^4\right).$$

6. LNe3:

$$\varepsilon_{LNe3} = \varepsilon_0 - \tfrac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t - \tfrac{\alpha^2}{160}u_{(8x)}\Delta x^4\Delta t + \tfrac{1}{6}\alpha^3 u_{(4x)}\tfrac{\Delta t^2}{\Delta x^2} + \tfrac{1}{9}\alpha^3 u_{(6x)}\Delta t^2 + \tfrac{31}{480}\alpha^3 u_{(8x)}\Delta x^2\Delta t^2$$
$$+\alpha^4 u_{(2x)}\tfrac{\Delta t^3}{\Delta x^6} + \tfrac{19}{12}\alpha^4 u_{(4x)}\tfrac{\Delta t^3}{\Delta x^4} + \tfrac{301}{360}\alpha^4 u_{(6x)}\tfrac{\Delta t^3}{\Delta x^2} + \tfrac{4159}{20{,}160}\alpha^4 u_{(8x)}\Delta t^3$$

$$\delta_{LNe3} = -\frac{\beta^3}{12}\left(u^n_{i\pm3} - 4u^n_{i\pm2} + 7u^n_{i\pm1} - 8u^n_i\right)\Delta t^2 - \frac{\beta^4}{24}\left(7u^n_{i\pm3} - 20u^n_{i\pm2} + 34u^n_{i\pm1} - 42u^n_i\right)\Delta t^3 + O\left(\Delta t^4\right).$$

7. LNe4:

$$\varepsilon_{LNe4} = \varepsilon_0 - \tfrac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t - \tfrac{\alpha^2}{160}u_{(8x)}\Delta x^4\Delta t + \tfrac{1}{6}\alpha^3 u_{(4x)}\tfrac{\Delta t^2}{\Delta x^2} + \tfrac{1}{9}\alpha^3 u_{(6x)}\Delta t^2 + \tfrac{31}{480}\alpha^3 u_{(8x)}\Delta x^2\Delta t^2$$
$$-\tfrac{1}{6}\alpha^4 u_{(6x)}\tfrac{\Delta t^3}{\Delta x^2} - \tfrac{1}{8}\alpha^4 u_{(8x)}\Delta t^3$$

$$\delta_{LNe4} = -\frac{\beta^3}{12}\left(u^n_{i\pm3} - 4u^n_{i\pm2} + 7u^n_{i\pm1} - 8u^n_i\right)\Delta t^2 - \frac{\beta^4}{12}\left(u^n_{i\pm4} - 6u^n_{i\pm3} + 16u^n_{i\pm2} - 26u^n_{i\pm1} + 30u^n_i\right)\Delta t^3 + O\left(\Delta t^4\right)$$

One can see that the largest inconsistent term $\Delta t^2\Delta x^{-4}$ of LNe disappeared in LNe3, and similarly, $\Delta t^3\Delta x^{-6}$ and $\Delta t^3\Delta x^{-4}$ of LNe3 disappeared from LNe4, which can explain the observed improvement in the numerical accuracy of these iterated algorithms. On the other hand, the terms in the δ errors are not really smaller in the case of the LNe3 and the LNe4 methods. This can explain why we experienced in our earlier publication [32] that these are not much more accurate than the LNe2 method for fixed and rather stiff spatially discretized systems.

8. CLL:

$$\varepsilon_{CLL} = \varepsilon_0 - \tfrac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t - \tfrac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t - \tfrac{\alpha^2}{160}u_{(8x)}\Delta x^4\Delta t - \tfrac{\alpha^3}{24}u_{(8x)}\Delta x^2\Delta t^2 + \tfrac{2}{3}p\alpha^4 u_{(2x)}\tfrac{\Delta t^3}{\Delta x^6}$$
$$+\tfrac{1}{54}(39p+11)\alpha^4 u_{(4x)}\tfrac{\Delta t^3}{\Delta x^4} + \tfrac{453p+355}{1620}\alpha^4 u_{(6x)}\tfrac{\Delta t^3}{\Delta x^2} + \tfrac{17}{30{,}240}(89p+161)\alpha^4 u_{(8x)}\Delta t^3$$

$$\delta_{CLL} = \frac{\beta^4}{216}\left[(23-36p)u^n_{i\pm3} - 8(7-9p)u^n_{i\pm2} + 4(20-27p)u^n_{i\pm1} - 2(47-72p)u^n_i\right]\Delta t^3 + O\left(\Delta t^4\right)$$

It is obvious that, unlike the previous algorithms, this one is third-order. From the truncation error, it follows that the value of p should be as small as possible. The stability analysis in the original paper revealed that p should be at least $2/3$, thus this value is the optimal. In the expression of δ, the $p = 2/3$ choice makes all the rounded brackets equal to one in absolute value, which also suggests that this is close to the optimal p. With this choice we obtain:

$$\delta_{\mathrm{CLL}} = \frac{\beta^4}{216}\left[-u^n_{i\pm3} - 8u^n_{i\pm2} + 8u^n_{i\pm1} + 2u^n_i\right]\Delta t^3 + O\left(\Delta t^4\right).$$

Note that since the exponents $e^{-\frac{2}{3}\cdot\frac{r}{2}}$ must be calculated anyway due to the second stage, the code will be faster with this choice, which is a significant gain in the non-uniform case, where these exponents are different for each node.

9. PI:

$$\varepsilon_{\mathrm{PI}} = \varepsilon_0 - \frac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t + \alpha^3 u_{(2x)}\frac{\Delta t^2}{\Delta x^4} + \frac{13}{12}\alpha^3 u_{(4x)}\frac{\Delta t^2}{\Delta x^2} + \frac{121}{360}\alpha^3 u_{(6x)}\Delta t^2,$$

$$\delta_{\mathrm{PI}} = -\frac{\beta^3}{6}\left(u^n_{i\pm2} - 3u^n_{i\pm1} + 4u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right).$$

10. ADE: If we analyze the core Formulas (9) and (10), where only nearest neighbors are taken into account, we obtain:

$$\tau_{\mathrm{ADE}} = \varepsilon_0 - \frac{\alpha^2}{24}u_{(6x)}\Delta x^2\Delta t - \frac{1}{12}\alpha^3 u_{(6x)}\Delta x^3\Delta t^2$$

$$\varepsilon_{\mathrm{ADE}} = \varepsilon_0 + \frac{\alpha^2}{12}u_{(4x)}\Delta t - \frac{7}{180}\alpha^2 u_{(6x)}\Delta x^2\Delta t - \frac{\alpha^3}{12}u_{(4x)}\frac{\Delta t^2}{\Delta x^2} + \frac{2}{45}\alpha^3 u_{(6x)}\Delta t^2.$$

The expression of τ and ε are definitely not the same. The first one demonstrates a second-order consistent scheme; the second is first-order and conditionally consistent. According to the numerical experiments, ADE is second-order but conditionally consistent. The problem may be caused by the fact that (9) and (10) are actually implicit formulas and only the alternating direction splitting procedure makes them explicit. In other words, the information where the p^{n+1}_{i-1} and q^{n+1}_{i+1} values come from is not present in (9) and (10). Now we make an attempt to involve the whole alternating direction procedure to obtain a more reliable result. This is performed by expressing the new values p^{n+1}_{i-1} and q^{n+1}_{i+1} of the neighbors by formulas (10) with indices shifted by one, and substituted into (9) and (10) again. With this, one extra neighbor is involved on both sides and we obtain:

$$\tau_{\mathrm{ADE}} = \varepsilon_0 - \frac{\alpha^2}{24}u_{(6x)}\Delta x^2\Delta t - \frac{1}{12}\alpha^2 u_{(4x)}\Delta t - \frac{31}{360}\alpha^2 u_{(6x)}\Delta x^2\Delta t - \frac{11}{12}\alpha^3 u_{(4x)}\frac{\Delta t^2}{\Delta x^2} - \frac{149}{360}\alpha^3 u_{(6x)}\Delta t^2,$$

$$\varepsilon_{\mathrm{ADE}} = \varepsilon_0 - \frac{1}{12}\alpha^2 u_{(6x)}\Delta x^2\Delta t - \frac{11}{12}\alpha^3 u_{(4x)}\frac{\Delta t^2}{\Delta x^2} - \frac{119}{360}\alpha^3 u_{(6x)}\Delta t^2.$$

Repeating this procedure recursively, two or more extra neighbors are involved. In these cases we obtain:

$$\tau_{\mathrm{ADE}} = \varepsilon_0 - \frac{\alpha^2}{24}u_{(6x)}\Delta x^2\Delta t - \frac{1}{12}\alpha^2 u_{(4x)}\Delta t - \frac{31}{360}\alpha^2 u_{(6x)}\Delta x^2\Delta t - \alpha^3 u_{(4x)}\frac{\Delta t^2}{\Delta x^2} - \frac{7}{12}\alpha^3 u_{(6x)}\Delta t^2$$

$$\varepsilon_{\mathrm{ADE}} = \varepsilon_0 - \frac{1}{12}\alpha^2 u_{(6x)}\Delta x^2\Delta t - \alpha^3 u_{(4x)}\frac{\Delta t^2}{\Delta x^2} - \frac{\alpha^3}{2}u_{(6x)}\Delta t^2$$

We believe that this last expression reflects the real nature of the numerical method since it means a second-order scheme, which is conditionally consistent. However, the leading inconsistent term contains $\Delta t^2\Delta x^{-2}$, instead of $\Delta t^2\Delta x^{-4}$ as in the case of the previous methods, thus this method converges much faster than those.

11. DF: We have to face a similar problem as in the case of the ADE method. If the core formulas are the starting point, we obtain different expressions for τ and ε:

$$\tau_{\mathrm{DF}} = \varepsilon_0 - \alpha^3 u_{(4x)} \frac{\Delta t^2}{\Delta x^2} + \frac{\alpha^3}{6} u_{(6x)} \Delta t^2,$$

$$\varepsilon_{\mathrm{DF}} = \varepsilon_0 + \frac{\alpha^2}{6} u_{(4x)} \Delta t + \frac{1}{180} \alpha^2 u_{(6x)} \Delta x^2 \Delta t + \frac{2}{3} \alpha^3 u_{(4x)} \frac{\Delta t^2}{\Delta x^2} + \frac{7}{45} \alpha^3 u_{(6x)} \Delta t^2.$$

We note that τ_{DF} contains the terms which were given in the classical books [12–14,37,38]. If we consider the very first time step taken by the θ formula, and then the second time step by the DF formula, the ε error term also shows second-order accuracy.

In the case of the odd–even hopscotch-type methods, the procedure to obtain the odd and even node-values is different; we have to calculate both the odd and the even errors. We list them on the following pages.

12. OOEH:

$$\varepsilon_{\mathrm{OOEH,\ odd}} = \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t + 2\alpha^3 u_{(4x)} \tfrac{\Delta t^2}{\Delta x^2} + \alpha^3 u_{(6x)} \Delta t^2,$$
$$\varepsilon_{\mathrm{OOEH,\ even}} = \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t - \tfrac{\alpha^3}{3} u_{(6x)} \Delta t^2.$$

$$\delta_{\mathrm{OOEH,\ odd}} = \tfrac{2}{3}\beta^3 \left(-u^n_{i\pm3} + 3u^n_{i\pm2} - 3u^n_{i\pm1} + 2u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right),$$
$$\delta_{\mathrm{OOEH,\ even}} = \tfrac{1}{3}\beta^3 \left(u^n_{i\pm3} - 6u^n_{i\pm2} + 15u^n_{i\pm1} - 20u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right).$$

13. RH:

$$\varepsilon_{\mathrm{RH,\ odd}} = \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t + 4\alpha^3 u_{(2x)} \tfrac{\Delta t^2}{\Delta x^4} + \tfrac{13}{3} \alpha^3 u_{(4x)} \tfrac{\Delta t^2}{\Delta x^2} + \tfrac{121}{90} \alpha^3 u_{(6x)} \Delta t^2,$$
$$\varepsilon_{\mathrm{RH,\ even}} = \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t - 4\alpha^3 u_{(2x)} \tfrac{\Delta t^2}{\Delta x^4} - \tfrac{7}{3} \alpha^3 u_{(4x)} \tfrac{\Delta t^2}{\Delta x^2} - \tfrac{61}{90} \alpha^3 u_{(6x)} \Delta t^2.$$

$$\delta_{\mathrm{RH,\ odd}} = -\tfrac{2}{3}\beta^3 \left(u^n_{i\pm3} - 3u^n_{i\pm1} + 4u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right),$$
$$\delta_{\mathrm{RH,\ even}} = \tfrac{1}{3}\beta^3 \left(u^n_{i\pm3} + 3u^n_{i\pm1} - 8u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right).$$

The truncation errors of the RH method are larger than the OOEH's ones, but the δ errors are smaller, especially for the even nodes. This may be the reason why the original method is more accurate for equidistant and mildly stiff systems, but less accurate in extremely stiff cases.

14. OEH-CNe:

$$\varepsilon_{\mathrm{OEH-CNe,\ odd}} = \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t + \tfrac{10}{3} \alpha^3 u_{(2x)} \tfrac{\Delta t^2}{\Delta x^4} + \tfrac{59}{18} \alpha^3 u_{(4x)} \tfrac{\Delta t^2}{\Delta x^2} + \tfrac{127}{108} \alpha^3 u_{(6x)} \Delta t^2,$$
$$\varepsilon_{\mathrm{OEH-CNe,\ even}} = \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t - \tfrac{2}{3} \alpha^3 u_{(2x)} \tfrac{\Delta t^2}{\Delta x^4} - \tfrac{19}{18} \alpha^3 u_{(4x)} \tfrac{\Delta t^2}{\Delta x^2} - \tfrac{271}{540} \alpha^3 u_{(6x)} \Delta t^2.$$

$$\delta_{\mathrm{OEH-CNe,\ odd}} = \tfrac{1}{3}\beta^3 \left(-2u^n_{i\pm3} + 3u^n_{i\pm2} - 4u^n_{i\pm1} + 6u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right),$$
$$\delta_{\mathrm{OEH-CNe,\ even}} = \tfrac{1}{3}\beta^3 \left(u^n_{i\pm3} - 3u^n_{i\pm2} + 5u^n_{i\pm1} - 6u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right).$$

15. SH:

$$\begin{aligned} \varepsilon_{\mathrm{SH,\ odd}} &= \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t - \alpha^3 u_{(2x)} \tfrac{\Delta t^2}{\Delta x^4} - \tfrac{7}{12} \alpha^3 u_{(4x)} \tfrac{\Delta t^2}{\Delta x^2} - \tfrac{61}{360} \alpha^3 u_{(6x)} \Delta t^2, \\ \varepsilon_{\mathrm{SH,\ even}} &= \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t + \alpha^3 u_{(2x)} \tfrac{\Delta t^2}{\Delta x^4} + \tfrac{13}{12} \alpha^3 u_{(4x)} \tfrac{\Delta t^2}{\Delta x^2} + \tfrac{121}{360} \alpha^3 u_{(6x)} \Delta t^2. \end{aligned} \tag{A1}$$

$$\delta_{\mathrm{SH,\ odd}} = \tfrac{1}{12}\beta^3 \left(u^n_{i\pm3} + 3u^n_{i\pm1} - 8u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right),$$
$$\delta_{\mathrm{SH,\ even}} = -\tfrac{1}{6}\beta^3 \left(u^n_{i\pm3} - 3u^n_{i\pm1} + 4u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right).$$

16. SH-CNe:

$$\varepsilon_{\mathrm{SH-CNe,\ odd}} = \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t - \tfrac{1}{6} \alpha^3 u_{(2x)} \tfrac{\Delta t^2}{\Delta x^4} - \tfrac{19}{72} \alpha^3 u_{(4x)} \tfrac{\Delta t^2}{\Delta x^2} - \tfrac{271}{2160} \alpha^3 u_{(6x)} \Delta t^2,$$
$$\varepsilon_{\mathrm{SH-CNe,\ even}} = \varepsilon_0 - \tfrac{\alpha^2}{6} u_{(6x)} \Delta x^2 \Delta t + \tfrac{5}{6} \alpha^3 u_{(2x)} \tfrac{\Delta t^2}{\Delta x^4} + \tfrac{59}{72} \alpha^3 u_{(4x)} \tfrac{\Delta t^2}{\Delta x^2} + \tfrac{127}{432} \alpha^3 u_{(6x)} \Delta t^2.$$

$$\delta_{\text{SH-CNe, odd}} = \tfrac{1}{12}\beta^3\left(u^n_{i\pm3} - 3u^n_{i\pm1} + 5u^n_{i\pm1} - 6u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right),$$
$$\delta_{\text{SH-CNe, even}} = -\tfrac{1}{12}\beta^3\left(2u^n_{i\pm3} - 3u^n_{i\pm2} + 4u^n_{i\pm1} - 6u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right).$$

17. ASH:

$$\begin{aligned}\varepsilon_{\text{ASH, odd}} &= \varepsilon_0 - \tfrac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t - \alpha^3 u_{(2x)}\tfrac{\Delta t^2}{\Delta x^4} - \tfrac{7}{12}\alpha^3 u_{(4x)}\tfrac{\Delta t^2}{\Delta x^2} - \tfrac{61}{360}\alpha^3 u_{(6x)}\Delta t^2,\\ \varepsilon_{\text{ASH, even}} &= \varepsilon_0 - \tfrac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t + \alpha^3 u_{(2x)}\tfrac{\Delta t^2}{\Delta x^4} + \tfrac{13}{12}\alpha^3 u_{(4x)}\tfrac{\Delta t^2}{\Delta x^2} + \tfrac{121}{360}\alpha^3 u_{(6x)}\Delta t^2.\end{aligned} \tag{A2}$$

$$\delta_{\text{ASH, odd}} = \tfrac{1}{12}\beta^3\left(u^n_{i\pm3} + 3u^n_{i\pm1} - 8u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right),$$
$$\delta_{\text{ASH, even}} = -\tfrac{1}{6}\beta^3\left(u^n_{i\pm3} - 3u^n_{i\pm1} + 4u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right).$$

18. ASH-CNe:

$$\varepsilon_{\text{ASH-CNe, odd}} = \varepsilon_0 - \tfrac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t - \tfrac{1}{6}\alpha^3 u_{(2x)}\tfrac{\Delta t^2}{\Delta x^4} - \tfrac{19}{72}\alpha^3 u_{(4x)}\tfrac{\Delta t^2}{\Delta x^2} - \tfrac{271}{2160}\alpha^3 u_{(6x)}\Delta t^2,$$
$$\varepsilon_{\text{ASH-CNe, even}} = \varepsilon_0 - \tfrac{\alpha^2}{12}u_{(6x)}\Delta x^2\Delta t + \tfrac{5}{6}\alpha^3 u_{(2x)}\tfrac{\Delta t^2}{\Delta x^4} + \tfrac{59}{72}\alpha^3 u_{(4x)}\tfrac{\Delta t^2}{\Delta x^2} + \tfrac{127}{432}\alpha^3 u_{(6x)}\Delta t^2.$$

$$\delta_{\text{ASH-CNe, odd}} = \tfrac{1}{12}\beta^3\left(u^n_{i\pm3} - 3u^n_{i\pm1} + 5u^n_{i\pm1} - 6u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right),$$
$$\delta_{\text{ASH-CNe, even}} = -\tfrac{1}{12}\beta^3\left(2u^n_{i\pm3} - 3u^n_{i\pm2} + 4u^n_{i\pm1} - 6u^n_i\right)\Delta t^2 + O\left(\Delta t^3\right).$$

19. LH. In this case, one cannot distinguish individual time steps and the stages of the whole calculations are entangled. We made an attempt to calculate the truncation errors, but we do not think this issue was solved by our tentative work. Similarly to the ADE and DF schemes, we examined only the diamond-shaped core formula and obtained:

$$\tau_{\text{LH}} = \varepsilon_0 + \frac{\alpha^3}{4}u_{(4x)}\frac{\Delta t^2}{\Delta x^2} + \frac{\alpha^3}{24}u_{(6x)}\Delta t^2, \tag{A3}$$

$$\begin{aligned}\varepsilon_{\text{LH}} = \varepsilon_0 &+ \tfrac{\alpha^2}{12}u_{(4x)}\Delta t + \tfrac{\alpha^2}{360}u_{(6x)}\Delta x^2\Delta t + \tfrac{\alpha^2}{20{,}160}u_{(8x)}\Delta x^4\Delta t\\ &+ \tfrac{1}{6}\alpha^3 u_{(4x)}\tfrac{\Delta t^2}{\Delta x^2} + \tfrac{187}{5040}\alpha^3 u_{(6x)}\Delta t^2 - \tfrac{1}{20{,}160}\alpha^3 u_{(8x)}\Delta x^2\Delta t^2.\end{aligned} \tag{A4}$$

The τ error is very small, which may explain that usually this method is the most accurate among the 20 examined ones. On the other hand, the ε error is much larger and there is even a first-order term in it.

20. LH-CNe: the problems are similar, but here at least the τ and ε errors are the same.

$$\begin{aligned}\tau_{\text{LH-CNe}} = \varepsilon_{\text{LH-CNe}} = \varepsilon_0 &+ \tfrac{\alpha^2}{12}u_{(4x)}\Delta t + \tfrac{\alpha^2}{360}u_{(6x)}\Delta x^2\Delta t + \tfrac{\alpha^2}{20{,}160}u_{(8x)}\Delta x^4\Delta t\\ &+ \tfrac{1}{3}\alpha^3 u_{(2x)}\tfrac{\Delta t^2}{\Delta x^4} + \tfrac{7}{36}\alpha^3 u_{(4x)}\tfrac{\Delta t^2}{\Delta x^2} + \tfrac{43}{1080}\alpha^3 u_{(6x)}\Delta t^2 - \tfrac{1}{30{,}240}\alpha^3 u_{(8x)}\Delta t^2.\end{aligned}$$

However, this truncation error says this formula is first-order, but according to the numerical experiment, it has the same accuracy as the SH-CNe and ASH-CNe. We think that some further nontrivial investigations would be necessary for the LH and LH-CNe methods.

References

1. Hundsdorfer, W.H.; Verwer, J.G. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*; Springer: Berlin, Germany, 2003.
2. Acton, Q.A. *Issues in Biophysics and Geophysics Research and Application: 2011 Edition*; ScholarlyEditions: Atlanta, GA, USA, 2012; ISBN 9781464964299.
3. Zhokh, A.; Strizhak, P. Advection–diffusion in a porous medium with fractal geometry: Fractional transport and crossovers on time scales. *Meccanica* **2022**, *57*, 833–843. [CrossRef]
4. Yu, H.; Yao, L.; Ma, Y.; Hou, Z.; Tang, J.; Wang, Y.; Ni, Y. The Moisture Diffusion Equation for Moisture Absorption of Multiphase Symmetrical Sandwich Structures. *Mathematics* **2022**, *10*, 2669. [CrossRef]
5. Zimmerman, R.W. *The Imperial College Lectures in Petroleum Engineering*; World Scientific Publishing: Singapore; London, UK, 2018; ISBN 9781786345004.
6. Savović, S.M.; Djordjevich, A. Numerical solution of diffusion equation describing the flow of radon through concrete. *Appl. Radiat. Isot.* **2008**, *66*, 552–555. [CrossRef] [PubMed]
7. Suárez-Carreño, F.; Rosales-Romero, L. Convergency and stability of explicit and implicit schemes in the simulation of the heat equation. *Appl. Sci.* **2021**, *11*, 4468. [CrossRef]

8. Hundsdorfer, W.; Verwer, J. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*; Springer Series in Computational Mathematics; Springer: Berlin/Heidelberg, Germany, 2003; Volume 33, ISBN 978-3-642-05707-6.
9. Lima, S.A.; Kamrujjaman, M.; Islam, M.S. Numerical solution of convection-diffusion-reaction equations by a finite element method with error correlation. *AIP Adv.* **2021**, *11*, 1–12. [CrossRef]
10. Ndou, N.; Dlamini, P.; Jacobs, B.A. Enhanced Unconditionally Positive Finite Difference Method for Advection–Diffusion–Reaction Equations. *Mathematics* **2022**, *10*, 2639. [CrossRef]
11. 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**, *11*, 23–40. [CrossRef]
12. Zhang, J.; Zhao, C. Sharp error estimate of BDF2 scheme with variable time steps for molecular beam expitaxial models without slop selection. *J. Math.* **2021**, *41*, 1–19.
13. 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**, *99*, 1139–1158. [CrossRef]
14. Jalghaf, H.K.; Kovács, E.; Majár, J.; Nagy, Á.; Askar, A.H. Explicit stable finite difference methods for diffusion-reaction type equations. *Mathematics* **2021**, *9*, 3308. [CrossRef]
15. Singh, M.K.; Rajput, S.; Singh, R.K. Study of 2D contaminant transport with depth varying input source in a groundwater reservoir. *Water Sci. Technol. Water Supply* **2021**, *21*, 1464–1480. [CrossRef]
16. Ji, Y.; Zhang, H.; Xing, Y. New Insights into a Three-Sub-Step Composite Method and Its Performance on Multibody Systems. *Mathematics* **2022**, *10*, 2375. [CrossRef]
17. Essongue, S.; Ledoux, Y.; Ballu, A. Speeding up mesoscale thermal simulations of powder bed additive manufacturing thanks to the forward Euler time-integration scheme: A critical assessment. *Finite Elem. Anal. Des.* **2022**, *211*, 103825. [CrossRef]
18. Reguly, I.Z.; Mudalige, G.R. Productivity, performance, and portability for computational fluid dynamics applications. *Comput. Fluids* **2020**, *199*. [CrossRef]
19. Gagliardi, F.; Moreto, M.; Olivieri, M.; Valero, M. The international race towards Exascale in Europe. *CCF Trans. High Perform. Comput.* **2019**, *1*, 3–13. [CrossRef]
20. Appadu, A.R. Performance of UPFD scheme under some different regimes of advection, diffusion and reaction. *Int. J. Numer. Methods Heat Fluid Flow* **2017**, *27*, 1412–1429. [CrossRef]
21. Sanjaya, F.; Mungkasi, S. A simple but accurate explicit finite difference method for the advection-diffusion equation. *J. Phys. Conf. Ser.* **2017**, *909*, 1–5. [CrossRef]
22. Pourghanbar, S.; Manafian, J.; Ranjbar, M.; Aliyeva, A.; Gasimov, Y.S. An efficient alternating direction explicit method for solving a nonlinear partial differential equation. *Math. Probl. Eng.* **2020**, *2020*, 1–12. [CrossRef]
23. Al-Bayati, A.; Manaa, S.; Al-Rozbayani, A. Comparison of Finite Difference Solution Methods for Reaction Diffusion System in Two Dimensions. *AL-Rafidain J. Comput. Sci. Math.* **2011**, *8*, 21–36. [CrossRef]
24. Nwaigwe, C. An Unconditionally Stable Scheme for Two-Dimensional Convection-Diffusion-Reaction Equations. 2022. Available online: https://www.researchgate.net/publication/357606287_An_Unconditionally_Stable_Scheme_for_Two-Dimensional_Convection-Diffusion-Reaction_Equations (accessed on 14 September 2022).
25. Savović, S.; Drljača, B.; Djordjevich, A. A comparative study of two different finite difference methods for solving advection–diffusion reaction equation for modeling exponential traveling wave in heat and mass transfer processes. *Ric. Mat.* **2021**, *71*, 245–252. [CrossRef]
26. Liu, H.; Leung, S. An Alternating Direction Explicit Method for Time Evolution Equations with Applications to Fractional Differential Equations. *Methods Appl. Anal.* **2020**, *26*, 249–268. [CrossRef]
27. Saleh, M.; Nagy, Á.; Kovács, E. Part 1: Construction and investigation of new numerical algorithms for the heat equation. *Multidiszcip. Tudományok* **2020**, *10*, 323–338. [CrossRef]
28. Saleh, M.; Nagy, Á.; Kovács, E. Part 3: Construction and investigation of new numerical algorithms for the heat equation. *Multidiszcip. Tudományok* **2020**, *10*, 349–360. [CrossRef]
29. Nagy, Á.; Saleh, M.; Omle, I.; Kareem, H.; Kovács, E. New stable, explicit, shifted-hopscotch algorithms for the heat equation. *Math. Comput. Appl.* **2021**, *26*, 61. [CrossRef]
30. Nagy, Á.; Omle, I.; Kareem, H.; Kovács, E.; Barna, I.F.; Bognar, G. Stable, Explicit, Leapfrog-Hopscotch Algorithms for the Diffusion Equation. *Computation* **2021**, *9*, 92. [CrossRef]
31. Kovács, E.; Nagy, Á.; Saleh, M. A set of new stable, explicit, second order schemes for the non-stationary heat conduction equation. *Mathematics* **2021**, *9*, 2284. [CrossRef]
32. Kovács, E. A class of new stable, explicit methods to solve the non-stationary heat equation. *Numer. Methods Partial Differ. Equ.* **2020**, *37*, 2469–2489. [CrossRef]
33. Kovács, E.; Nagy, Á.; Saleh, M. A New Stable, Explicit, Third-Order Method for Diffusion-Type Problems. *Adv. Theory Simul.* **2022**, *5*, 2100600. Available online: https://onlinelibrary.wiley.com/doi/10.1002/adts.202100600 (accessed on 14 September 2022). [CrossRef]
34. Holmes, M.H. *Introduction to Numerical Methods in Differential Equations*; Springer: New York, NY, USA, 2007; ISBN 978-0387-30891-3.
35. Morton, K.W.; Mayers, D.F. *Numerical Solution of Partial Differential Equations*, 2nd ed.; Cambridge University Press: Cambridge, MA, USA, 2005; ISBN 978-0-521-60793-3.

36. Özişik, M.N. *Finite Difference Methods in Heat Transfer*; CRC Press: Boca Raton, FL, USA, 2017; ISBN 69781482243468.
37. Richtmyer, R.D.; Morton, K.W. *Difference Methods for Initial Value Problems*, 2nd ed.; Wiley: New York, NY, USA, 1967.
38. Mitchell, A.R.; Griffiths, D.F. *The Finite Difference Method in Partial Differential Equations*; Wiley: Chichester, UK, 1980; ISBN 0-471-27811-3.
39. 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* **2022**, *10*, 2813. [CrossRef]
40. Iserles, A. *A First Course in the Numerical Analysis of Differential Equations*; Cambridge University Press: Cambridge, MA, USA, 2009; ISBN 9788490225370.
41. Chen-Charpentier, B.M.; Kojouharov, H.V. An unconditionally positivity preserving scheme for advection-diffusion reaction equations. *Math. Comput. Model.* **2013**, *57*, 2177–2185. [CrossRef]
42. Kovács, E. New Stable, Explicit, First Order Method to Solve the Heat Conduction Equation. *J. Comput. Appl. Mech.* **2020**, *15*, 3–13. [CrossRef]
43. Barakat, H.Z.; Clark, J.A. On the solution of the diffusion equations by numerical methods. *J. Heat Transfer* **1966**, *88*, 421–427. [CrossRef]
44. Hirsch, C. *Numerical Computation of Internal and External Flows, Volume 1: Fundamentals of Numerical Discretization*; Wiley: Hoboken, NJ, USA, 1988.
45. Gourlay, A.R.; McGuire, G.R. General Hopscotch Algorithm for the Numerical Solution of Partial Differential Equations. *IMA J. Appl. Math.* **1971**, *7*, 216–227. [CrossRef]
46. Saleh, M.; Kovács, E. New Explicit Asymmetric Hopscotch Methods for the Heat Conduction Equation. *Comput. Sci. Math. Forum* **2021**, *2*, 22.
47. Ferziger, J.H.; Perić, M. *Computational Methods for Fluid Dynamics*; Springer: Berlin, Germany, 1996.
48. Mátyás, L.; Barna, I.F. General Self-Similar Solutions of Diffusion Equation and Related Constructions. *Rom. J. Phys.* **2022**, *67*, 101.