computation

Article

Stable, Explicit, Leapfrog-Hopscotch Algorithms for the Diffusion Equation

Ádám Nagy [1], Issa Omle [1], Humam Kareem [1,2], Endre Kovács [1,*], Imre Ferenc Barna [3] and Gabriella Bognar [4]

[1] Institute of Physics and Electrical Engineering, University of Miskolc, 3515 Miskolc, Hungary; adam.nagy0310@gmail.com (Á.N.); issa.j.omle@gmail.com (I.O.); 20310@uotechnology.edu.iq (H.K.)
[2] Department of Mechanical Engineering, University of Technology, Baghdad 10066, Iraq
[3] Wigner Research Center for Physics, 1051 Budapest, Hungary; barna.imre@wigner.hu
[4] Institute of Machine and Product Design, University of Miskolc, 3515 Miskolc, Hungary; v.bognar.gabriella@uni-miskolc.hu
* Correspondence: kendre01@gmail.com

Abstract: In this paper, we construct novel numerical algorithms to solve the heat or diffusion equation. We start with 10^5 different leapfrog-hopscotch algorithm combinations and narrow this selection down to five during subsequent tests. We demonstrate the performance of these top five methods in the case of large systems with random parameters and discontinuous initial conditions, by comparing them with other methods. We verify the methods by reproducing an analytical solution using a non-equidistant mesh. Then, we construct a new nontrivial analytical solution containing the Kummer functions for the heat equation with time-dependent coefficients, and also reproduce this solution. The new methods are then applied to the nonlinear Fisher equation. Finally, we analytically prove that the order of accuracy of the methods is two, and present evidence that they are unconditionally stable.

Keywords: odd-even hopscotch method; diffusion equation; heat equation; explicit time-integration; stiff equations; unconditional stability

Citation: 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. https://doi.org/10.3390/computation9080092

Academic Editor: Ali Cemal Benim

Received: 31 July 2021
Accepted: 13 August 2021
Published: 20 August 2021

1. Introduction

This paper can be considered to be a continuation of our previous work [1–4], in which we developed novel numerical algorithms to solve the heat or diffusion equation. The phenomenon of diffusion is described by the heat or diffusion equation, which, in its simplest form, is the following linear parabolic partial differential equation (PDE):

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \tag{1}$$

where u is the concentration or temperature, and α is the diffusion coefficient or thermal diffusivity in the case of particle or heat transport, respectively. If the medium in which the diffusion takes place is not homogeneous, one can use a more general form

$$c\rho \frac{\partial u}{\partial t} = \nabla(k\nabla u) \tag{2}$$

where, in the case of conductive heat transfer, $k = k\left(\vec{r}, t\right)$, $c = c\left(\vec{r}, t\right)$, and $\rho = \rho\left(\vec{r}, t\right)$ are the heat conductivity, specific heat, and mass density, respectively. The $\alpha = k/(c\rho)$ relation connects the four nonnegative quantities.

The heat equation, either the simplest version (1) or similar but more difficult equations, has an increasing number of analytical solutions. Unfortunately, most consider only systems with relatively simple geometries, such as a cylindrical inclusion in an infinite medium [5]. Moreover, parameters such as the diffusion coefficient or the heat conductivity are typically taken as constants, or a fixed and relatively simple function of the space variables [6] at best. This means that, for intricate geometries, and in the case of space- and time-dependent coefficients in general, numerical calculations cannot be avoided. However, approaches such as the finite difference (FDM) or finite element (FEM) methods require the full spatial discretization of the system; thus, they are computationally demanding. We explained in our previous papers [3,4,7], and also demonstrate in this paper, that the widely used conventional solvers, either explicit or implicit, have serious difficulties. The explicit methods are usually conditionally stable, so when the stiffness of the problem is high, very small time step sizes have to be used. By comparison, when the number of cells is large, which is almost always the case in two or three dimensions, the implicit solvers become very slow and use a significant amount of memory. This information highlights the fact that finding effective numerical methods is still important.

One of the very few easily parallelizable explicit and unconditionally stable methods is the two-stage odd-even hopscotch (OEH) algorithm [3,8–10]. In our previous papers [3,4], we showed that this method is robust and powerful for spatially homogeneous grids but, in the case of stiff systems, it can be disastrously inaccurate for large time step sizes. We constructed and tested new hopscotch combinations and found [1–4] that some of them behave much better, not only for large, but also for medium and small time step sizes. In this paper, we extend our research by further modifying the underlying space and time structure as explained below.

Our current work was inspired by the well-known leapfrog method [11] used by the molecular dynamics community to solve Newton's equations of motion. In their books, Hockney and Eastwood [12] (p. 28) and, later, Frenkel and Smit [13] (p. 75) introduced this method in the following form:

$$\begin{aligned} v(t+\Delta t/2) &= v(t-\Delta t/2) + F(x(t))\Delta t/m \\ x(t+\Delta t) &= x(t) + v(t+\Delta t/2)\Delta t \end{aligned}$$

In Figure 1, a diagram of this method is presented, and one can immediately understand why the method is called leapfrog.

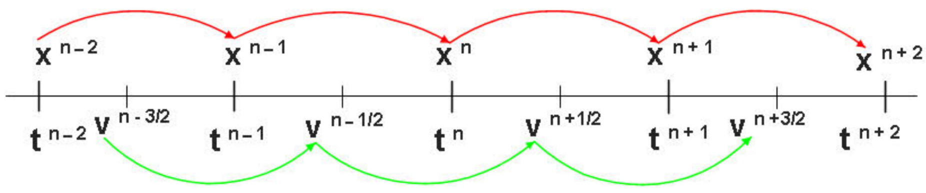


Figure 1. Diagram of the original leapfrog algorithm.

The leapfrog scheme was successfully adapted and also generalized to hyperbolic PDEs [14], but for parabolic equations it is unstable. This can be addressed by a modification to obtain the Dufort–Frankel method [15] (Subsection 8.3). In all of these publications, the nodes of the spatial mesh are treated equivalently, unlike in the OEH method. Verwer and Sommeijer [16] presented a composite method for the advection-diffusion problem, where the leapfrog scheme was used for the horizontal advection part, and the Du Fort Frankel and Crank–Nicolson methods are used for the other parts. It can be easily determined that their method is significantly different from ours. In addition, we have been unable to find any combinations of the leapfrog and hopscotch structures in the literature that are at least a little similar to ours.

The remainder of this paper is organized as follows. In Section 2 we introduce the new algorithms, both for the simplest, one-dimensional, equidistant mesh, and also for general, arbitrary meshes. In Section 3, first we very briefly present the results of the numerical tests for the first assessment of the 10^5 combinations to obtain a manageable number of methods. Then, in Sections 3.3 and 3.4, two numerical experiments are presented for two space dimensional stiff systems consisting of 10,000 cells. We choose the top five combinations with the highest accuracy and compare them to selected other methods. In Section 3.5, we reproduce an analytical solution from the top five methods using a non-equidistant mesh for verification. In Section 3.6, we construct a new analytical solution for time-dependent diffusivity and then, in Section 3.7, the top five methods are verified again by comparing their numerical results to this solution. In Section 3.8, the convergence and stability properties of these methods are analytically investigated. In Section 4 we summarize the conclusions and our research plan for the future.

2. The New Methods

To implement the OEH algorithm, we need to construct a bipartite grid, in which the set of nodes (or cells) is divided into two similar subsets with odd and even space indices, such that all nearest neighbors of odd nodes are even, and vice versa. In one space dimension, where the space variable is discretized as usual, $x_i = i\Delta x,\ i = 0, \dots, N-1$, it is straightforward. Let us fix the time discretization for the remainder of the paper to $t_n = t_0 + nh,\ n = 1, \dots, T,\ T = (t_{\text{fin}} - t_0)/h$. In the original OEH method, odd and even time steps can also be distinguished: in odd time steps the odd nodes are treated in the first stage, and only the u values belonging to the beginning of the time step are used. Then, in the second stage, when the even nodes are calculated, the new values of their odd neighbors, which have just been calculated, are used. In even time steps, the roles of the nodes are interchanged, as shown in Figure 2a. The original OEH method always applies the standard explicit Euler formula in the first stage and the implicit Euler formula in the second stage. However, in each stage of this structure, the latest available u values of the neighbors are used; thus, when the second stage calculations begin, the new values of the neighbors u_{i-1}^{n+1} and u_{i+1}^{n+1} are known, which makes the implicit formula effectively explicit. Therefore, it is obvious that the OEH method is fully explicit, and the previous u values do not need to be stored at all, which means that one array is sufficient to store the variable u in the memory. We emphasize that all of our algorithms described in this paper have this property, irrespective of the modifications. In one of our previous papers [1], we applied the UPFD method in stage one and the explicit Euler in stage two, which we then called the UPFD + Explicit Euler odd-even Hopscotch method. However, this can simply be the called the reversed hopscotch method, as one only needs to change the order of the two formulas in the code of the original OEH to obtain the code of this method. Then, we modified the space and time structures and constructed the so called shifted-hopscotch method [4].

In the case of the new leapfrog-hopscotch method, the calculation starts with taking a half-sized time step for the odd nodes using the initial u_i^0 values, which is symbolized by a green box containing the number 0 in Figure 2b. Then, full time steps are taken for the even and odd nodes strictly alternately until the end of the last timestep (orange box in Figure 1), which must also be halved for odd nodes to reach exactly the same time point as the even nodes. In Figure 1, the number "0" means that this stage is performed only once and is not repeated. By comparison, stages 1–4 are repeated $T/2$ times (including the last, halved stage 4), so T must be an even integer. Thus, finally, we have five different stages 0…4 in which five different formulas can be applied to maximize the efficiency. Let us turn our attention to these formulas.

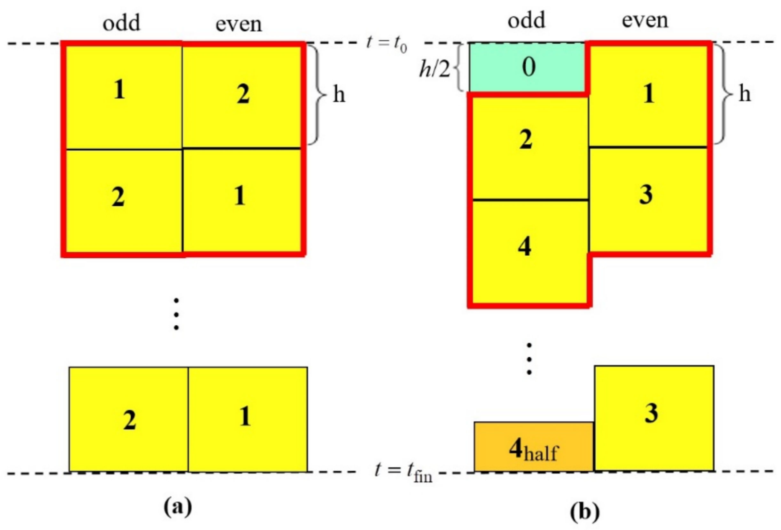


Figure 2. The diagram of the odd-even hopscotch structures for two nodes. The numbers in the boxes represent the different formulas, whereas the areas surrounded by thick red lines are the repeating units. (**a**) The original OEH algorithm. (**b**) The new leapfrog-hopscotch structure. The green and orange boxes are non-repeating stages.

It is relatively common [17] (p. 112) to spatially discretize Equation (1) in one dimension by applying the central difference formula to obtain a system of ordinary differential equations (ODEs) for nodes $i = 1, \ldots, N-2$:

$$\frac{du_i}{dt} = \alpha \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2}. \tag{3}$$

Suppose now that one needs to solve Equation (2) when the material properties are not constants but functions of the space variables and the mesh is possibly unstructured. We have shown in our previous papers [3,7] (based on, e.g., Chapter 5 of the book [18]) that Equation (3) can be generalized to arbitrary grids consisting of cells of various shapes and properties:

$$\frac{du_i}{dt} = \sum_{j \neq i} \frac{u_j - u_i}{R_{i,j} C_i}, \tag{4}$$

where the heat capacity and the inverse thermal resistance can be calculated approximately as:

$$C_i = c_i \rho_i V_i \text{ and } 1/R_{ij} \approx k_{ij} A_{ij} / d_{ij}, \tag{5}$$

respectively, where V_i is the volume of the cell, and A_{ij} and d_{ij} are the surface between the cells and the distance between the cell centers, respectively. Figure 3 can help the reader to understand these quantities. Although the grid should be topologically rectangular to maintain the explicit property of the OEH-type methods, we emphasize that the geometrical shape of the cells is not necessarily rectangular.

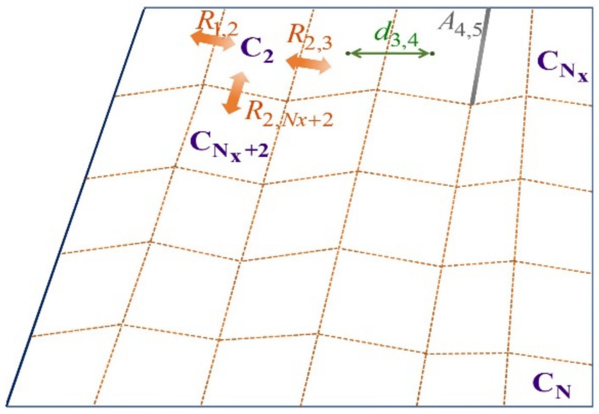


Figure 3. Visualization of the generalized variables. The red arrows symbolize conductive (heat) transport between cells with heat capacities C_i and C_j through the resistances R_{ij}.

Equation systems (3) and (4) can be written into a condensed matrix-form:

$$\frac{d\vec{u}}{dt} = M\vec{u}. \tag{6}$$

The matrix M is tridiagonal in the one-dimensional case of Equation (3) with the following elements:

$$m_{ii} = -\frac{2\alpha}{\Delta x^2}\ (1 < i < N),\ m_{i,i+1} = \frac{\alpha}{\Delta x^2}\ (1 \le i < N),\ m_{i,i-1} = \frac{\alpha}{\Delta x^2}\ (1 < i \le N). \tag{7a}$$

In the general case of Equation (4), the nonzero elements of the matrix can be given as:

$$m_{ij} = \frac{1}{R_{i,j}C_i}\ ,\ m_{ii} = -\sum_{j\neq i} m_{ij}. \tag{7b}$$

Let us introduce the characteristic time or time constant τ_i of cell i, which is always a non-negative quantity:

$$\tau_i \doteq -(m_{ii})^{-1}. \tag{8}$$

Using this we introduce the following notations:

$$r_i = \frac{h}{\tau_i} \text{ and } A_i = h\sum_{j\neq i} m_{ij}u_j^n = h\sum_{j\neq i}\frac{u_j^n}{C_iR_{ij}}. \tag{9}$$

Here, r_i is the generalization of $r = \frac{\alpha h}{\Delta x^2} = -\frac{m_{ii}h}{2}$, the usual mesh ratio. We recall the so-called theta method:

$$u_i^{n+1} = u_i^n + r\left[\theta\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right) + (1-\theta)\left(u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}\right)\right], \tag{10}$$

where $\theta \in [0,1]$. If $\theta = 1$, this scheme is the explicit (Euler) or forward-time central-space (FTCS) scheme. Otherwise the theta method is implicit, and for $\theta = 0$, $1/2$ one has the implicit (Euler) and the Crank–Nicolson methods, respectively [17]. However, we remind the reader that in our leapfrog-hopscotch scheme, the neighbors are always taken into account at the same, most recent time period; thus, we can express the new value of the u variable in the following brief form in the 1D case:

$$u_i^{\mu+1} = \frac{(1-2r\theta)u_i^{\mu} + r\left(u_{i-1}^{\mu+1/2} + u_{i+1}^{\mu+1/2}\right)}{1+2r(1-\theta)}, \tag{11}$$

and in the general case:

$$u_i^{\mu+1} = \frac{(1-r_i\theta)u_i^{\mu} + A_i}{1+r_i(1-\theta)}. \tag{12}$$

Here, $\mu = n$ for the even cells, and $\mu \in \left\{n,\ n-\frac{1}{2},\ n-1\right\}$ for the odd cells, depending on where we are in the leapfrog-hopscotch structure. For easier understanding, see Figure 1, in addition to Example 1 below.

The other formula we use is derived from the constant-neighbor (CNe) method, which was introduced in our previous papers [7,19]. For the leapfrog-hopscotch method in the one-dimensional case, this can be written as follows:

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

whereas for general grids it is:

$$u_i^{\mu+1} = u_i^{\mu} \cdot e^{-r_i} + \frac{A_i}{r_i}\left(1-e^{-r_i}\right), \tag{14}$$

and for halved time steps, r_i and A_i must be divided by 2, obviously.

Remark 1. *In case of the theta method for $\theta = 0$ (Formulas (11) and (12)) and the CNe methods (Formulas (13) and (14)), the new u_i^{n+1} values are the convex combinations of the old u_j^n values. Indeed, in (11) the coefficients $(1-2r)/(1+2r)$, $r/(1+2r)$, and $r/(1+2r)$ are nonnegative and their sum is one. This property can be similarly shown for (12)–(14).*

Here, briefly outline the notation of the individual combinations. The five numbers in a bracket are the values of the parameter θ for each of the stages $0, \ldots, 4$, whereas the letter "C" means the CNe constant neighbor method. For example, L2 (1/4, 1/2, C, 1/2, 1/2) means the following algorithm, which will be selected among the top five algorithms in Section 3.3, and is also denoted L2.

Example 1. ***Algorithm L2*** *(1/4, 1/2, C, 1/2, 1/2), general from.*

Stage 0. Take a half time step with the (12) formula with θ = 1/4 for odd cells:

$$u_i^{1/2} = \frac{(1-{r_i}/8)u_i^0 + A_{i,\text{half}}}{1+{r_i}/2(1-1/4)},\ A_{i,\text{half}} = \frac{h}{2}\sum_{j\neq i}\frac{u_j^0}{C_iR_{ij}}.$$

Repeat the following four stages for $n = 0, \ldots, T-3$:

Stage 1. Take a full time step with the (12) formula with θ = 1/2 for even cells:

$$u_i^{n+1} = \frac{(1-{r_i}/2)u_i^n + A_i}{1+r_i(1-1/2)},\ A_i = h\sum_{j\neq i}\frac{u_j^{n+1/2}}{C_iR_{ij}}.$$

Stage 2. Take a full time step with the (14) formula for odd cells:

$$u_i^{n+3/2} = u_i^{n+1/2} \cdot e^{-r_i} + \frac{A_i}{r_i}\left(1-e^{-r_i}\right),\ A_i = h\sum_{j\neq i}\frac{u_j^{n+1}}{C_iR_{ij}}.$$

Stage 3. Take a full time step with the (12) formula with θ = 1/2 for even cells:

$$u_i^{n+2} = \frac{(1-{r_i}/2)u_i^{n+1} + A_i}{1+r_i(1-1/2)},\ A_i = h\sum_{j\neq i}\frac{u_j^{n+3/2}}{C_iR_{ij}}.$$

Stage 4. If $n + 2 < T - 1$, then take a full time step with $\theta = 1/2$ for odd cells:

$$u_i^{n+5/2} = \frac{(1 - r_i/2)u_i^{n+3/2} + A_i}{1 + r_i(1 - 1/2)},\ A_{i,} = h\sum_{j\neq i} \frac{u_j^{n+2}}{C_i R_{ij}},$$

else take a half time step with the (12) formula with $\theta = 1/2$ for odd cells:

$$u_i^{n+2} = \frac{(1 - r_i/4)u_i^{n+3/2} + A_{i,\text{half}}}{1 + r_i/2(1 - 1/2)},\ A_{i,\text{half}} = \frac{h}{2}\sum_{j\neq i} \frac{u_j^{n+2}}{C_i R_{ij}}.$$

Based on this example, all other combinations can be easily constructed. In our previous paper [4], the best shifted-hopscotch combination was S4 (0, 1/2, 1/2, 1/2, 1), and, in the category of the positivity-preserving algorithms, S1 (C, C, C, C, C). These will be tested against the new leapfrog-hopscotch algorithms in the following section.

3. Results

3.1. On the Methods and Circumstances of the Investigation

In Sections 3.2–3.4, two space dimensional, topologically rectangle-structured lattices with $N = N_x \times N_z$ cells are investigated (see Figure 3 for visualization). We consider the system as thermally isolated (zero Neumann boundary conditions), which is implemented by the omission of those terms of the sum in Equation (4), which have infinite resistivity in the denominator due to the isolated boundary. Let us denote by λ_{MIN} and λ_{MAX} the eigenvalues of the system matrix M with the (nonzero) smallest and the largest absolute values, respectively. Now, $\lambda_{\text{MAX}}/\lambda_{\text{MIN}}$ gives the stiffness ratio of the system, and the maximum possible time step size for the FTCS (explicit Euler) scheme is exactly given by $h_{\text{MAX}}^{\text{FTCS}} = |2/\lambda_{\text{MAX}}|$, above which the solutions are expected to diverge due to instability. This threshold is frequently called the CFL limit and also holds for the second-order explicit Runge–Kutta (RK) method [20].

In Sections 3.2–3.4, the reference solution is calculated by the ode15s built-in solver of MATLAB, with very strict error tolerance and, therefore, high precision. In Section 3.5, Section 3.6, and Section 3.8 the reference solution is an analytical solution. The (global) numerical error is the absolute difference of the numerical solutions u_j^{num} produced by the examined method and the reference solution u_j^{ref} at final time t_{fin}. We use these individual errors of the nodes or cells to calculate the maximum error:

$$\text{Error}(L_\infty) = \max_{1\le j\le N}\left|u_j^{\text{ref}}(t_{\text{fin}}) - u_j^{\text{num}}(t_{\text{fin}})\right|, \tag{15}$$

the average error:

$$\text{Error}(L_1) = \frac{1}{N}\sum_{1\le j\le N}\left|u_j^{\text{ref}}(t_{\text{fin}}) - u_j^{\text{num}}(t_{\text{fin}})\right|, \tag{16}$$

and the so-called energy error:

$$\text{Error}(Energy) = \sum_{1\le j\le N} C_j\left|u_j^{\text{ref}}(t_{\text{fin}}) - u_j^{\text{num}}(t_{\text{fin}})\right|, \tag{17}$$

which, in the case of heat conduction, yields the error in terms of energy. However, for different algorithms, these errors depend on the time step size in different ways. If we want to evaluate the overall performance of the new methods compared to the original OEH method, we consider a fixed value of the final time t_{fin}, and first calculate the solution with a large time step size (typically $t_{\text{fin}}/4$), then repeat the whole calculation for subsequently halved time step sizes S times until h reaches a very small value (typically around 2×10^{-6}).

Then, the aggregated relative error (ARE) quantities are calculated for each type of error defined above. For instance, in the case of the L_∞ error, it has the following form:

$$\mathrm{ARE}(L_\infty) = \frac{1}{S}\sum_{i=1}^{S}\left(\log(\mathrm{Error}(L_\infty))_{\mathrm{OEH}} - \log(\mathrm{Error}(L_\infty))_{\mathrm{examined}}\right). \tag{18}$$

We can also take the simple average of these errors,

$$\mathrm{ARE} = \frac{1}{3}(\mathrm{ARE}(L_\infty) + \mathrm{ARE}(L_1) + \mathrm{ARE}(\mathit{Energy})), \tag{19}$$

to obtain one single quantity for a method applied on a concrete problem, which will help us to select the best combinations. For example, if ARE = 2, then the examined method produces smaller errors by two orders of magnitude than the OEH method.

For the simulations where running times are measured, we used a desktop computer with an Intel Core i5-9400 CPU, 16.0 GB RAM with the MATLAB R2020b software, in which there was a built-in tic-toc function to measure the total running time of the tested algorithms.

3.2. Preliminary Tests

The following nine values of the parameter theta are substituted into the formula given in (12): $\theta \in \{0, 1/5, 1/4, 1/3, 1/2, 2/3, 3/4, 4/5, 1\}$. Thus, with the CNe Formula (14), we have 10 different formulas. There are five stages in the leapfrog-hopscotch structure and, by inserting these 10 formulas in all possible combinations, we obtain 10^5 = 100,000 different algorithm combinations in total. We wrote MATLAB code which constructs and tests all of these combinations in a highly automatized manner under the following circumstances.

The generalized Equation (4) is solved in the case of four different small systems with $N_x \times N_z = 2\times 2, 2\times 6, 4\times 4,$ and 3×5, to ensure the total running times are manageable despite the large number of combinations. Here, and also in Sections 3.3 and 3.4, randomly generated cell capacities and thermal resistances following a log-uniform distribution

$$C_i = 10^{(\alpha_C - \beta_C \times rand)}, R_{x,i} = 10^{(\alpha_{Rx} - \beta_{Rx} \times rand)}, R_{z,i} = 10^{(\alpha_{Rz} - \beta_{Rz} \times rand)} \tag{20}$$

have been used. The parameters $\alpha_C, \beta_C, \alpha_{Rx}, \beta_{Rx}, \alpha_{Rz}, \beta_{Rz}$ of the distribution of the mesh-cells data were chosen to construct test problems with various stiffness ratios and $h_{\mathrm{MAX}}^{\mathrm{FTCS}}$, for example, $\alpha_C = 1, 2,$ or $3, \beta_C = 2, 4,$ or 6. The (pseudo-)random number, *rand*, is generated by MATLAB for each quantity with a uniform distribution in the unit interval (0, 1). We also generate different random values for the initial conditions $u_i(0) = rand$. The final times were set to $t_{\mathrm{fin}} = 0.1$ and once to $t_{\mathrm{fin}} = 10$. The purpose of this latter, larger number is to let instabilities manifest themselves in order to exclude the unstable combinations. The computer program calculated the aggregated relative error (ARE) quantities and then sorted the algorithms in descending order according to this quantity to obtain a ranking list of the 10^5 algorithm combinations. Finally, we manually checked the top of these lists for all of the four small systems to select the best 20 combinations, which have the following short form:

$$\begin{gathered}(\mathrm{C}, \mathrm{C}, \mathrm{C}, \mathrm{C}, \mathrm{C}), (1/4, 1/2, \mathrm{C}, 1/2, 1/2), (1/4, 1/2, \mathrm{C}, 1/2, 1/2), (1/4, 1/2, 1/2, 1/2, \mathrm{C}),\\ (1/4, 1/2, \mathrm{C}, 1/2, \mathrm{C}), (\mathrm{C}, 1/2, \mathrm{C}, 1/2, \mathrm{C}), (\mathrm{C}, 1/4, 1/2, \mathrm{C}, 1/2), (\mathrm{C}, 1/2, \mathrm{C}, 1/2, 1/2),\\ (\mathrm{C}, 1/2, \mathrm{C}, \mathrm{C}, 1/2), (\mathrm{C}, 1/3, 1/2, 2/3, \mathrm{C}), (1/4, \mathrm{C}, 1/4, \mathrm{C}, 3/4), (0, \mathrm{C}, 1/2, 1/2, 1/2, 1/2)\\ (\mathrm{C}, 1/2, 1/2, 1/2, 1/2), (1/4, \mathrm{C}, 1/4, 1/2, 3/4), (0, 1/2, 1/2, 1/2, 1/2), (1, 1/2, 1/2, 1/2, 1/2),\\ (1/4, 1/2, 1/2, 1/2, 1/2), (1/3, 1/2, 1/2, 1/2, 1/2), (1/2, 1/2, 1/2, 1/2, 1/2), (1/5, 1/2, 1/2, 1/2, 1/2).\end{gathered} \tag{21}$$

In the next two subsections, we start only with these combinations. We note that there is an "odd one out" in the list above, i.e., the (C, C, C, C, C) combination, which was typically not at the top of the ranking lists above. We include it into the top 20 and, later, in the top five, because it is the best among those combinations which preserve positivity of the solution; more precisely, it always follows the maximum and minimum principles as the true solution [17] (p. 87), which is proven in Section 3.8.

We stress that, until this point, we have not stated anything precisely. The only purpose of these preliminary tests on the small systems is the reduction of the large number of algorithm combinations by eliminating those which are probably not worthy of more careful investigation.

3.3. Comparison with Other Methods for a Large, Moderately Stiff System

We examine a grid similar to the one in Figure 2 with an isolated boundary, but where the sizes are fixed to $N_x = 100$ and $N_z = 100$; thus, the total number of cells is 10,000, and the final time is $t_{\text{fin}} = 0.1$. The exponents introduced above were set to the following values:

$$\alpha_C = 2,\ \beta_C = 4,\ \alpha_{Rx} = \alpha_{Rz} = 1,\ \beta_{Rx} = \beta_{Rz} = 2, \tag{22}$$

which means, e.g., that log-uniformly distributed values between 0.01 and 100 were given to the capacities. The generated system can be characterized by its stiffness ratio and $h_{\text{MAX}}^{\text{FTCS}}$ values, which are 3.1×10^7 and 7.3×10^{-4}, respectively. The performance of the new algorithms is compared with the following, widely used MATLAB solvers:

- ode15s, a first- to fifth-order (implicit) numerical differentiation formulas with variable-step and variable order (VSVO), developed for solving stiff problems;
- ode23s, a second-order modified (implicit) Rosenbrock formula;
- ode23t, applies implicit trapezoidal rule using a free interpolant;
- ode23tb, a combination of backward differentiation formula and trapezoidal rule;
- ode45, a fourth-/fifth-order explicit Runge–Kutta–Dormand–Prince formula;
- ode23, a second-/third-order explicit Runge–Kutta–Bogacki–Shampine method;
- ode113, a first- to 13th-order VSVO Adams–Bashforth–Moulton solver.

In the case of the MATLAB solvers, we could not directly set the time step size, but changed the tolerances instead, starting from a large value, such as ‘AbsTol’ = ‘RelTol’ $\doteq$ ‘Tol’ $=10^3$, towards a small minimum value, for example, ‘Tol’ $=10^{-5}$. In addition to these solvers, we also used the following methods for comparison purposes; the original UPFD, the CNe, the two- and three-stage linear-neighbor (LNe and LNe3) methods [7], and the Dufort–Frankel (DF) algorithm [17] (p. 120):

$$u_i^{n+1} = \frac{(1 - r_i)u_i^{n-1} + 2A_i}{1 + r_i} \tag{23}$$

are explicit unconditionally stable schemes. The DF scheme is a two-step method which is not self-starting; thus, we calculated the first time step by two subsequent UPDF steps with halved time step sizes. The Heun method, also called the explicit trapezoidal rule, is one of the most common second-order RK scheme [21]. Finally, the widely used Crank–Nicolson scheme (abbreviated here as CrN) is an implicit scheme:

$$\underbrace{\left(I - \frac{h}{2}M\right)}_{A}\vec{u}^{\,n+1} = \underbrace{\left(I + \frac{h}{2}M\right)}_{B}\vec{u}^{\,n},$$

where I is the $N \times N$ unit matrix. The A and B matrixes here are time-independent; thus, it is sufficient to calculate them only once, before the first time step. We implement this scheme in two different ways: one is to calculate $Y = A^{-1}B$ before the first time step and then each time step is just a simple matrix multiplication $\vec{u}^{\,n+1} = Y\vec{u}^{\,n}$, which is denoted by “CrN invert”. The other is the $\vec{u}^{\,n+1} = \text{linsolve}\left(A, B\vec{u}^{\,n}\right)$ command of MATLAB, which we denote by “CrN lins”.

We plotted the L_∞, L_1, and energy errors as a function of the effective time step size h_{EFF}, and based on this (and on similar data that are presented in the next subsection), we selected the following top five combinations from those listed in (21) and discarded the reminder:

L1 (C, C, C, C, C),
L2 (0, 1/2, 1/2, 1/2, 1/2),
L3 (1/5, 1/2, 1/2, 1/2, 1/2),
L4 (1/4, 1/2, C, 1/2, 1/2),
L5 (1/5, 1/2, C, 1/2, 1/2).

One can see that L2–3 and L4–5 are highly similar; only the zeroth stage is different. We tried to optimize the value of theta for this zeroth stage by calculating the errors for a few fixed time step sizes for $\theta = k/N_\theta, k = 0, ..., N_\theta$, where $N_\theta = 1000$. In the case of the method L2–3 (θ, 1/2, 1/2, 1/2, 1/2), we found that the errors are larger for large values of theta and the smallest when $\theta \approx 0$. In case of the L4–5 (θ, 1/2, C, 1/2, 1/2) combination, we found that there is a nontrivial minimum of the error function at around $\theta \approx 0.2$, but its position slightly depends on the time step size, as shown in Figure 4. We can conclude that the L4 combination is usually slightly more accurate than that of L5.

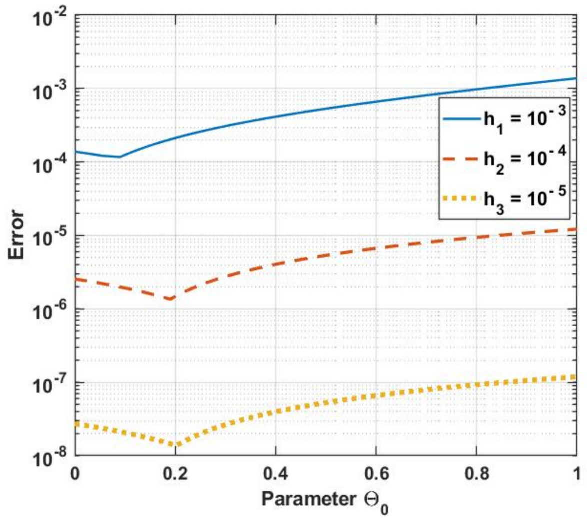


Figure 4. Errors as a function of the parameter θ_0 for the first (moderately stiff) system, in the case of the method L4–5 (θ, 1/2, C, 1/2, 1/2).

In Figure 5, we present the energy error as a function of the time step size for these top five combinations and other methods, and Figure 6 shows the L_1 errors vs. the total running times. Furthermore, Table 1 presents some results that were obtained by our numerical schemes and the "ode" routines of MATLAB. One can see that, as expected, the implicit Crank–Nicolson scheme is the most accurate, but the accuracy of the L2 method, in addition to the L5 and S4 methods, approaches it, and is similar to that of the Heun method below the CFL limit.

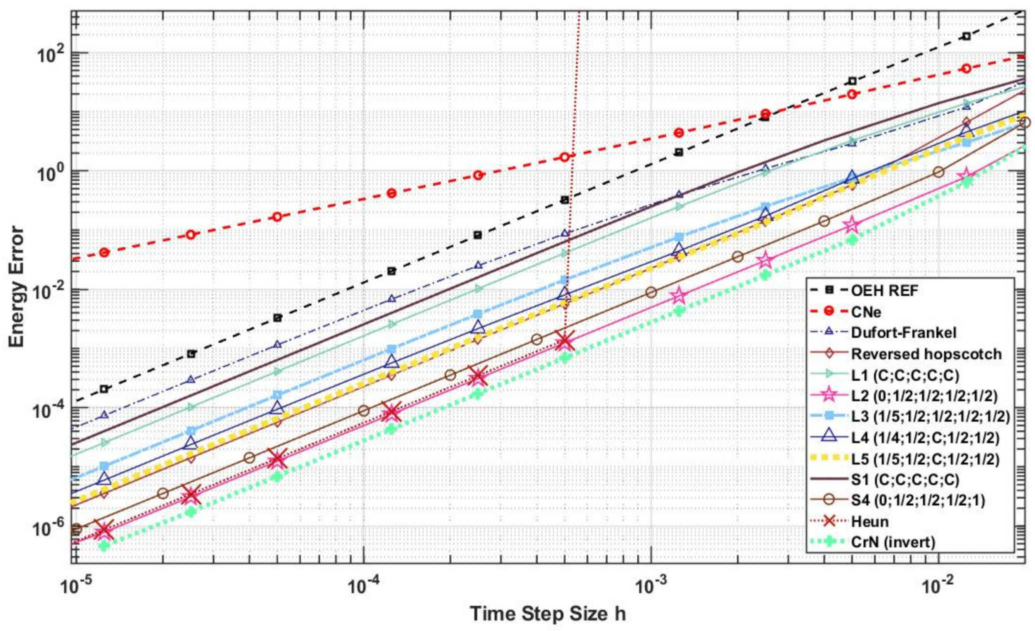


Figure 5. Energy errors as a function of the time step size for the first (moderately stiff) system, in the case of the original OEH method (OEH REF), the new algorithms L1–L5, and several other methods.

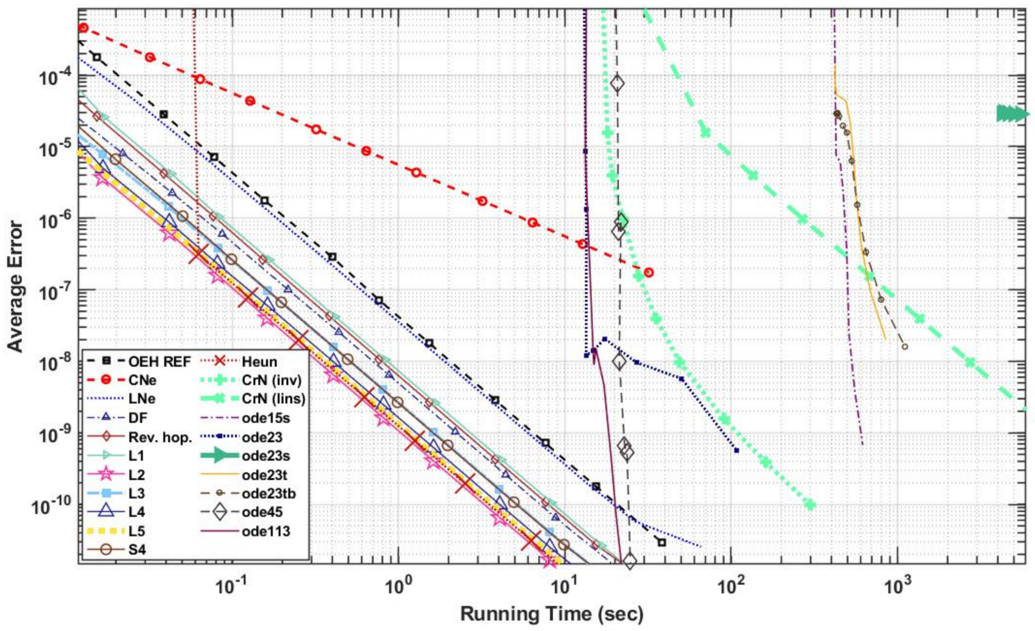


Figure 6. Errors as a function of the running time for the first (moderately stiff) system, in the case of the original OEH method (OEH REF), the new algorithms L1–L5, and several other methods.

Table 1. Comparison of different leapfrog-hopscotch algorithms with other methods for the moderately stiff system of ten thousand cells.

Numerical Method	**Error**(L_∞)	**Error**(L_1)	**Energy Error**	**Running Time (s)**
ode45, Tol $= 10^{-3}$	5.18×10^{-5}	1.01×10^{-8}	4.26×10^{-5}	2.12×10^{1}
ode23, Tol $= 10^{-5}$	3.85×10^{-6}	2.06×10^{-8}	1.03×10^{-4}	1.73×10^{1}
ode113, Tol $= 10^{-4}$	6.77×10^{-5}	1.49×10^{-8}	6.42×10^{-5}	1.54×10^{1}
ode15s, Tol $= 10^{2}$	1.18×10^{-2}	1.05×10^{-3}	5.27×10^{0}	$4.03 \times 10^{+2}$
ode23s, Tol $= 10^{2}$	4.11×10^{-4}	2.90×10^{-5}	1.46×10^{-1}	4.23×10^{3}
ode23t, Tol $= 10^{-7}$	1.32×10^{-6}	9.32×10^{-8}	4.68×10^{-4}	6.88×10^{2}
ode23tb, Tol $= 10^{3}$	4.18×10^{-4}	2.95×10^{-5}	1.48×10^{-1}	4.36×10^{2}
CNe, $h = 5 \times 10^{-4}$	3.33×10^{-3}	1.78×10^{-4}	1.71×10^{0}	3.19×10^{-2}
LNe, $h = 5 \times 10^{-5}$	2.99×10^{-6}	8.33×10^{-8}	6.84×10^{-4}	6.52×10^{-1}
LNe3, $h = 1.25 \times 10^{-4}$	8.78×10^{-6}	2.11×10^{-7}	2.50×10^{-3}	2.61×10^{-1}
DF, $h = 2.5 \times 10^{-6}$	1.04×10^{-8}	2.59×10^{-10}	2.98×10^{-6}	6.7×10^{0}
Rev. hop., $h = 5 \times 10^{-6}$	4.23×10^{-8}	4.27×10^{-10}	5.75×10^{-7}	3.82×10^{0}
L1, $h = 1.25 \times 10^{-4}$	9.06×10^{-6}	2.63×10^{-7}	2.56×10^{-3}	1.68×10^{-1}
L2, $h = 2.5 \times 10^{-4}$	1.16×10^{-5}	1.58×10^{-7}	3.11×10^{-4}	8.19×10^{-2}
L3, $h = 1.25 \times 10^{-6}$	5.19×10^{-10}	1.11×10^{-11}	2.99×10^{-7}	1.62×10^{-1}
L4, $h = 1.25 \times 10^{-5}$	3.05×10^{-8}	6.20×10^{-10}	5.98×10^{-6}	1.62×10^{0}
L5, $h = 5 \times 10^{-6}$	3.51×10^{-9}	8.01×10^{-11}	6.80×10^{-7}	4.07×10^{0}
S1, $h = 2.5 \times 10^{-6}$	3.63×10^{-9}	1.05×10^{-10}	1.03×10^{-6}	9.93×10^{0}
S4, $h = 1.25 \times 10^{-5}$	6.61×10^{-8}	6.68×10^{-10}	8.92×10^{-7}	1.97×10^{0}
Heun, $h = 1.25 \times 10^{-4}$	2.79×10^{-7}	1.97×10^{-8}	8.68×10^{-5}	2.5×10^{-1}
CrN (inv), $h = 5 \times 10^{-3}$	3.67×10^{-4}	1.59×10^{-5}	6.93×10^{-2}	1.80×10^{1}
CrN (lins), $h = 5 \times 10^{-3}$	3.67×10^{-4}	1.59×10^{-5}	6.93×10^{-2}	7.03×10^{1}

3.4. Comparison with Other Methods for a Large, Very Stiff System

In the second case study, new values were set for the α and β exponents:

$$\alpha_C = 3,\ \beta_C = 6,\ \alpha_{Rx} = 3,\ \alpha_{Rz} = 1,\ \beta_{Rx} = \beta_{Rz} = 4.$$

This means that the width of the distribution of the capacities and thermal resistances were increased and a non-negligible anisotropy appeared because, on average, the resistances in the x direction are two orders of magnitude larger than in the z direction. Note that the geometric mean of the C and R quantities is still equal to one. With this modification, we gained a system with a much higher stiffness ratio, 2.5×10^{11}, and the CFL limit for the standard FTCS was $h_{MAX}^{EE} = 1.6 \times 10^{-6}$, which, we stress again, also holds for the Heun method. All other parameters and circumstances are the same as in the previous subsection. In Figures 7 and 8, the energy and the L_∞ errors are presented as a function of the time step size and the total running time, respectively. As expected, the implicit methods gained a slight advantage compared to the previous, less stiff case, but the new L2 method outperforms all other examined methods provided that not only the accuracy, but also the speed, is taken into account. In Table 2, we report the data related to this numerical experiment, and in Table 3 we summarize the ARE error quantities, defined in Equation (19), of the explicit stable methods for both case studies.

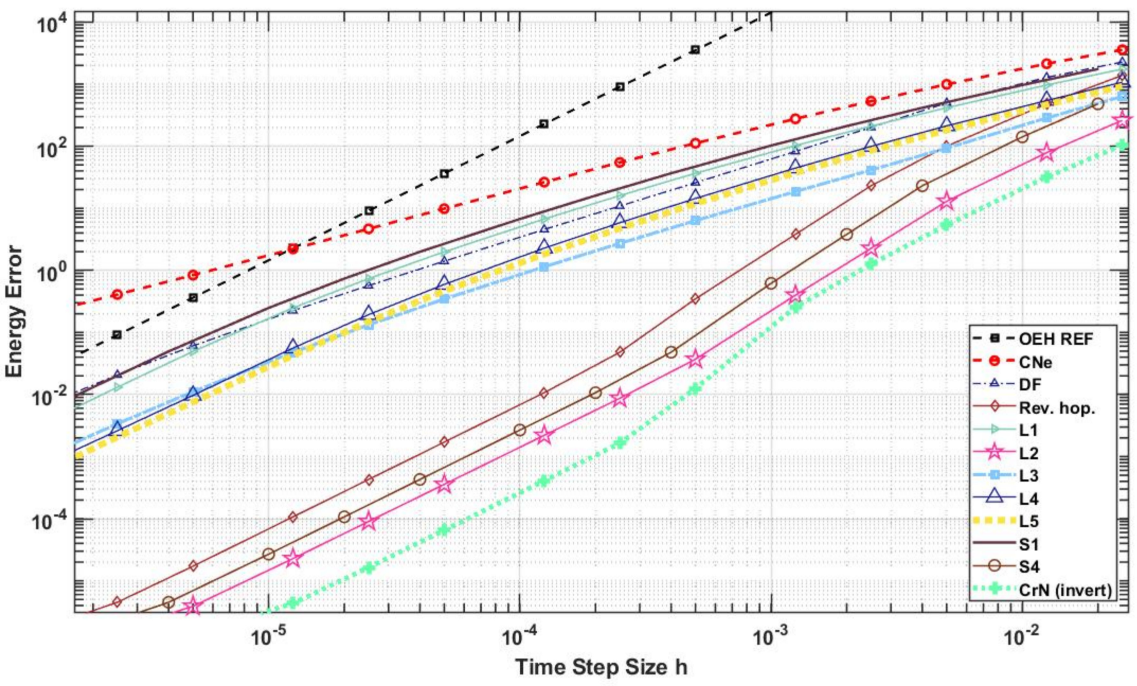


Figure 7. Energy errors as a function of the time step size for the second (very stiff) system, in the case of the original OEH method (OEH REF), the new algorithms L1–L5, and some other methods.

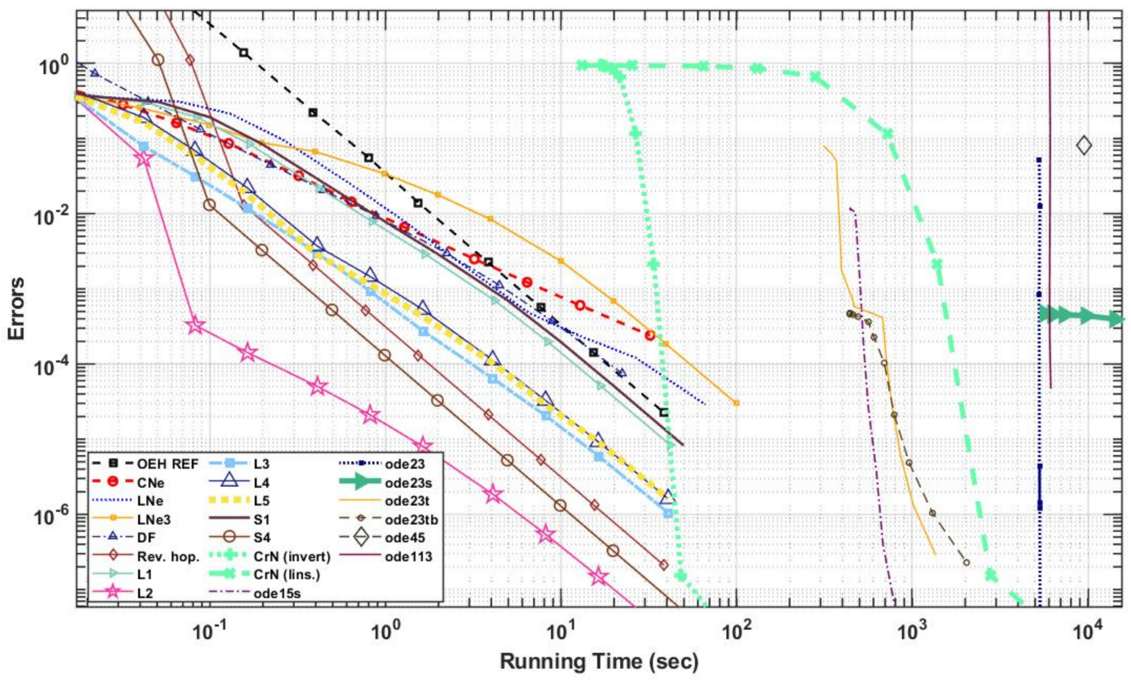


Figure 8. Errors as a function of the running time for the second (very stiff) system, in the case of the original OEH method (OEH REF), one stage CNe method, the new algorithms L1–L5, and several other methods.

Table 2. Comparison of different algorithms for the very stiff system of ten thousand cells.

Numerical Method	Error L_∞	Error L_1	Energy Error	Running Time (s)
ode45, Tol $= 10^0$	8.13×10^{-2}	1.52×10^{-5}	7.03×10^{-2}	9.48×10^3
ode23, Tol $= 10^{-2}$	1.25×10^{-2}	2.34×10^{-6}	1.08×10^{-2}	5.33×10^3
ode113, Tol $= 10^{-4}$	4.07×10^{-5}	8.84×10^{-9}	4.08×10^{-5}	6.15×10^3
ode15s, Tol $= 10^{-1}$	1.05×10^{-2}	7.19×10^{-4}	3.65×10^0	4.72×10^2
ode23s, Tol $= 10^{-3}$	3.12×10^{-4}	1.57×10^{-5}	7.89×10^{-2}	2.48×10^4
ode23t, Tol $= 10^{-5}$	2.58×10^{-5}	1.31×10^{-6}	6.55×10^{-3}	7.07×10^2
ode23tb, Tol $= 10^{-7}$	1.04×10^{-6}	5.29×10^{-8}	2.65×10^{-4}	1.30×10^3
CNe, $h = 1.25 \times 10^{-4}$	8.62×10^{-2}	1.32×10^{-3}	2.64×10^1	1.28×10^{-1}
LNe, $h = 5 \times 10^{-5}$	2.35×10^{-2}	1.37×10^{-4}	2.39×10^0	6.52×10^{-1}
LNe3, $h = 2.5 \times 10^{-5}$	1.79×10^{-2}	3.81×10^{-5}	7.26×10^{-1}	1.98×10^0
DF, $h = 1.25 \times 10^{-5}$	3.74×10^{-4}	3.75×10^{-7}	6.47×10^{-3}	8.9×10^0
Rev. hop., $h = 1.25 \times 10^{-5}$	1.31×10^{-4}	3.55×10^{-7}	1.07×10^{-4}	1.54×10^0
L1, $h = 1.25 \times 10^{-5}$	2.92×10^{-3}	1.46×10^{-5}	2.48×10^{-1}	1.67×10^0
L2, $h = 5 \times 10^{-6}$	1.85×10^{-6}	5.59×10^{-9}	3.89×10^{-6}	4.08×10^0
L3, $h = 2.5 \times 10^{-6}$	2.04×10^{-5}	1.52×10^{-7}	3.38×10^{-3}	8.18×10^0
L4, $h = 1.25 \times 10^{-3}$	4.36×10^{-1}	3.59×10^{-3}	4.44×10^1	1.65×10^{-2}
L5, $h = 5 \times 10^{-7}$	1.61×10^{-6}	6.76×10^{-9}	9.48×10^{-5}	4.08×10^1
S1, $h = 5.5 \times 10^{-5}$	2.20×10^{-2}	1.25×10^{-4}	1.99×10^0	4.94×10^{-1}
S4, $h = 2.5 \times 10^{-6}$	1.31×10^{-6}	3.55×10^{-9}	1.71×10^{-6}	9.87×10^0
Heun, $h = 1.25 \times 10^{-6}$	2.83×10^{-11}	1.39×10^{-12}	7.77×10^{-8}	2.52×10^1
CrN (inv), $h = 1.25 \times 10^{-4}$	1.56×10^{-7}	7.91×10^{-9}	4.08×10^{-4}	4.83×10^1
CrN (lins), $h = 1.25 \times 10^{-4}$	1.56×10^{-7}	7.91×10^{-9}	4.08×10^{-4}	2.78×10^3

Table 3. ARE (average relative error) quantities of different explicit stable algorithms.

Numerical Method	ARE (Mod. Stiff)	ARE (Very Stiff)
CNe	−1.242	1.264
LNe	0.699	1.804
LNe3	0.924	1.859
Dufort-Frankel	0.615	1.575
Reversed hopscotch	0.928	2.570
L1 (C, C, C, C, C)	0.985	1.885
L2 (0, 1/2, 1/2, 1/2, 1/2)	1.745	3.988
L3 (1/5, 1/2, 1/2, 1/2, 1/2)	1.326	2.451
L4 (1/4, 1/2, C, 1/2, 1/2)	1.486	2.315
L5 (1/5, 1/2, C, 1/2, 1/2)	1.573	2.385
S1 (C, C, C, C, C)	0.985	1.885
S4 (0, 1/2, 1/2, 1/2,1)	1.509	3.184

3.5. Verification 1. Comparison to Analytical Results Using a Non-Uniform Mesh

We consider a very recent [22] nontrivial analytical solution of Equation (1), given on the whole real number line for positive values of t as follows:

$$u^{\text{exact}} = \frac{x}{t^{5/2}}\left(1 - \frac{x^2}{6\alpha t}\right)e^{-\frac{x^2}{4\alpha t}}, \tag{24}$$

where, for the sake of simplicity, we use the value $\alpha = 1$. In our last paper [4], we reproduced this solution by prescribing the Dirichlet boundary conditions calculated using the analytical solution at the two ends of the interval. Now, we do not use this kind of information, but construct a large-scale non-equidistant spatial grid according to the following procedure. First, we define the coordinates of the cell borders by the formula:

$$x_j = x_{j-1} + \Delta x_{j-1}, x_0 = 0, \Delta x_0 = 0.01, \Delta x_j = \Delta x_0 \exp\left(\gamma j^4\right), j = 1, ..., 1000.$$

where $\gamma = 10^{-11}$. Thus, we have a relatively dense system of nodes close to the origin, which becomes increasingly less dense as one moves further from the origin, and towards $+5922.3$, which is the right boundary of the mesh. Then the cell centers are easily calculated:

$$X_j = X_{j-1} + \frac{\Delta x_j}{2}, X_0 = 0, j = 1, ..., 1000.$$

It is straightforward to reflect this structure to the origin to create the mirror image of the mesh at the negative side of the x axis, thus obtaining 2000 cells in total. Now, at the vicinity of the origin we have small cells with diameter 0.01, which are increasing as we move further from the origin, first very slowly, then increasingly rapidly until they reach $\Delta x_{\pm 1000} = 211.6$. The resistances and the cell capacities can then be calculated using Formula (5) in Section 2:

$$C_i = \Delta x_i \ , \ i = 1, ..., 2000 \text{ and } R_i = X_{i+1} - X_i \ , \ i = 1, ..., 1999$$

We took into account the zero Neumann boundary conditions, as in the previous two subsections, which is a good approximation because the values of the initial function are extremely close to zero far from the origin. The stiffness ratio is 5.7×10^{11} for this mesh, and $h_{\text{MAX}}^{\text{FTCS}} = 5 \times 10^{-5}$. As in our last paper [4], and also in the next subsection, we reproduce the analytical solution in the finite time interval $t \in [t_0, \ t_{\text{fin}}]$, where $t_0 = 0.5, t_{\text{fin}} = 1$. Obviously, the generalized Formulas (12) and (14) must be used. In Figure 9, the L_∞ errors as a function of the time step size h are presented for the case of the u solution for the top five leapfrog-hopscotch algorithms, a first-order "reference-curve" for the original CNe method, and the Heun method. These results verify not only the second-order convergence of the numerical methods, but also the procedure of generalizing the calculations to non-uniform grids (see Equations (4) and (5) above). One can also see that the L2 and L3 algorithms reach the minimum error (determined by the space discretization) for larger values of h than the CFL limit for the Heun method.

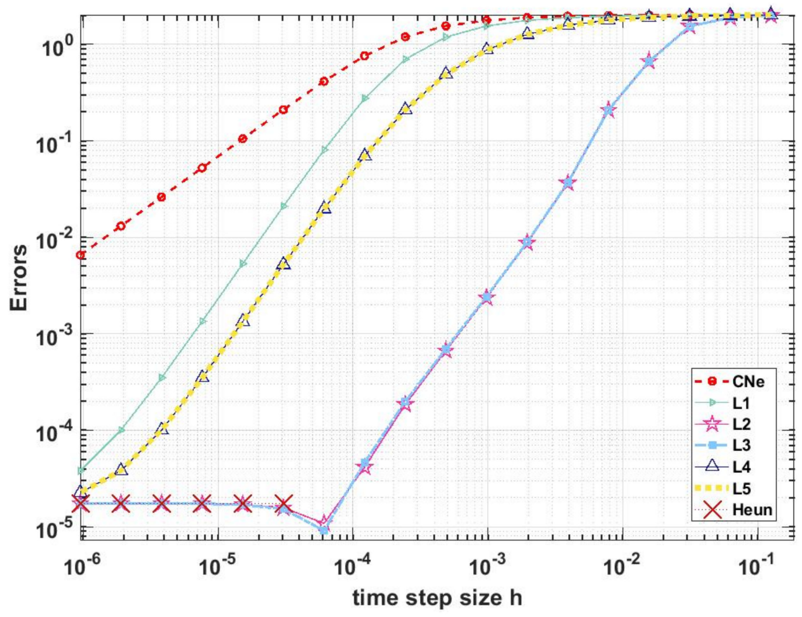


Figure 9. The L_∞ errors as a function of time step size h for the space-dependent mesh to reproduce the exact solution given in (24).

3.6. New Analytical Solution with Time-Dependent Diffusion Coefficient

Now we investigate the analytical solutions of the diffusion equation where the diffusion constant has a power-law time dependence:

$$\frac{\partial u(x,t)}{\partial t} = \hat{\alpha} t^n \frac{\partial^2 u(x,t)}{\partial x^2}. \tag{25}$$

To avoid confusion, we use the "hat" notation for the "generalized diffusion coefficient", $\hat{\alpha}$, which is considered constant and has the dimension of $t^{-n}\alpha$. We apply the self-similar Ansatz, first introduced by Sedov [23], with the form

$$u(x,t) = t^{-a} f(\eta), \eta = \frac{x}{t^b},$$

where $a, b \in \mathbb{R}$ are the self-similar exponents describing the decay and the spreading of the solution in time and space. These properties make this Ansatz physically extraordinarily relevant. In the previous decade, we applied this Ansatz to numerous physical systems described by partial differential equations, most with hydrodynamical origins [24,25].

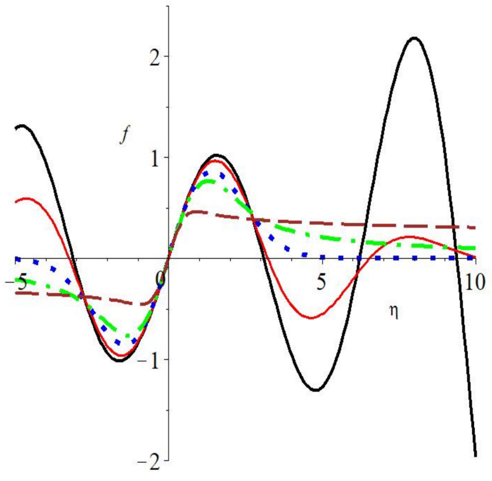
The constraint among the exponents is defined via $a + 1 = a + 2b - n$. Therefore, a can have arbitrary real values but $b = (1+n)/2$ must be fulfilled. These relations dictate the reduced ODE of:

$$-af(\eta) - \frac{1+n}{2}\eta \frac{df(\eta)}{d\eta} = \hat{\alpha}\frac{d^2 f(\eta)}{d\eta^2}.$$

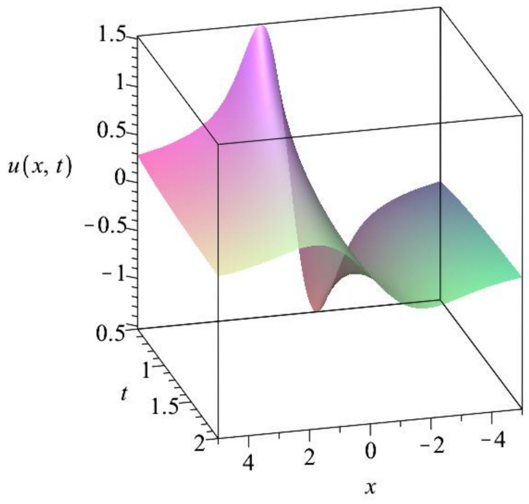
The solution can be evaluated with the Maple 12 software and reads:

$$f(\eta) = \eta e^{-\frac{(1+n)\eta^2}{4\hat{\alpha}}}\left(c_1 M\left[\frac{1+n-a}{1+n}, \frac{3}{2}, \frac{(1+n)\eta^2}{4\hat{\alpha}}\right] + c_2 U\left[\frac{1+n-a}{1+n}, \frac{3}{2}, \frac{(1+n)\eta^2}{4\hat{\alpha}}\right]\right), \tag{26}$$

where M and U are the Kummer functions [26], whereas c_1 and c_2 are arbitrary real constants. Note that the larger the exponent n, the quicker the spreading of the solutions. Figure 10 shows numerous shape functions for various values of n.



(**a**)



(**b**)

Figure 10. The M part of the solutions in the case of time-dependent diffusion coefficients of Equation (25). (**a**) Shape functions for different n exponents of the diffusion parameter. The black, red, blue dotted, green dashed-dotted, and brown dashed lines are for $n = -1.1,\ -0.8,\ 0,\ 1,$ and 10, respectively. (**b**) Illustration of the solution given in Equation (28) for $\hat{\alpha} = 1$.

In the next subsection we address the first term on the right-hand side of (26); thus, we give this part of the solution $(c_1 = 1, c_2 = 0)$ as a function of the x and t variables:

$$u(x,t) = t^{-a}\frac{x}{t^b}e^{-\frac{(1+n)x^2}{4\hat{\alpha}t^{2b}}}M\left[\frac{1+n-a}{1+n}, \frac{3}{2}, \frac{(1+n)x^2}{4\hat{\alpha}t^{2b}}\right]. \tag{27}$$

3.7. Verification 2. Comparison to Analytical Results with Time-Dependent Material Properties

We examine Equation (25) and take the solution given in (27) for $a = 1, n = 1$ as a reference solution:

$$u_2^{exact} = \frac{x}{t^2}e^{-\frac{x^2}{2\hat{\alpha}t^2}}M\left[\frac{1}{2}, \frac{3}{2}, \frac{x^2}{2\hat{\alpha}t^2}\right], \tag{28}$$

whose 3D graph can be seen in Figure 10b. We reproduce this solution in finite space and time intervals $x \in [x_1, x_2]$ and $t \in [t_0, t_{\text{fin}}]$, where $x_1 = -5, x_2 = 5$, while $t_0 = 0.5$, $t_{\text{fin}} = 1$, and $\hat{\alpha} = 1$. The space and time intervals are discretized as usual: $x_j = x_1 + j\Delta x$, $j = 0, ..., 1000, \Delta x = 0.01$ and $t_n = t_0 + nh, n = 1, ..., T, T = (t_{\text{fin}} - t_0)/h$; thus, we can return to the equidistant theta and CNe Formulas (11) and (13). For this grid, $h_{\text{MAX}}^{\text{FTCS}} = 5 \times 10^{-5}$

again, and the stiffness ratio is only 4.1×10^5. We use the analytical solution to prescribe the appropriate Dirichlet boundary conditions at the two ends of the interval. We take into consideration the continuous change of the thermal diffusivity by recalculating the mesh ratio at each time step by multiplying its original value by the physical time taken at the middle of the actual time step, i.e., $r_n = \left(t_n + \frac{h}{2}\right)\frac{\alpha h}{\Delta x^2}$ is used at the nth time step in Formulas (11) and (13). The values of the Kummer M function were calculated by the hypergeom function of MATLAB. The L_∞ errors as a function of the time step size h, which are presented in Figure 11, show that our methods, particularly L2 and L3, can also easily solve this problem. Figure 11 is highly similar to Figure 10, and we also obtained similar error functions for several other values of the parameters $t_0, t_{\text{fin}}, x_{1,2}, \Delta x, \ldots$ for both problems.

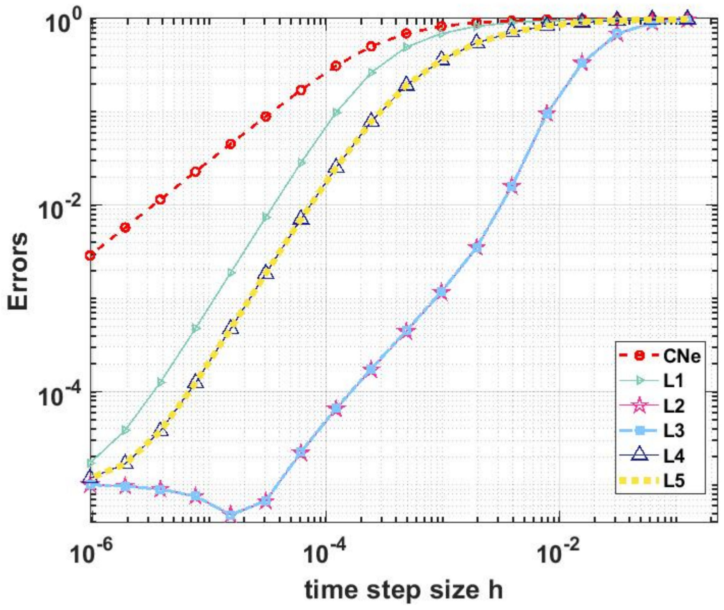


Figure 11. The L_∞ errors as a function of time step size h for the problem with a time-dependent coefficient.

We also present the graphs of the initial function $u(x, t = 0.5)$, the reference analytical solution $u^{\text{ref}}(x, t_{\text{fin}})$, and a corresponding numerical solution for $h = 7.8 \times 10^{-3}$ in Figure 12. We intentionally chose such a large time step size (more than two orders of magnitude larger than $h_{\text{MAX}}^{\text{FTCS}}$), for which the L_∞ error is 0.096, to demonstrate the robustness of the algorithms, because now one can see that there is nothing unphysical in the numerical solution. We observed that, for even larger time step sizes, the numerical solution is approaching the initial function, which means that increasing the time step size is roughly equivalent to the slowing of the numerical diffusion process. The second conclusion we can draw is that the CNe formula is less effective than the $\theta = 1/2$ formula if we have small, nearly equally sized cells directly adjacent to one another.

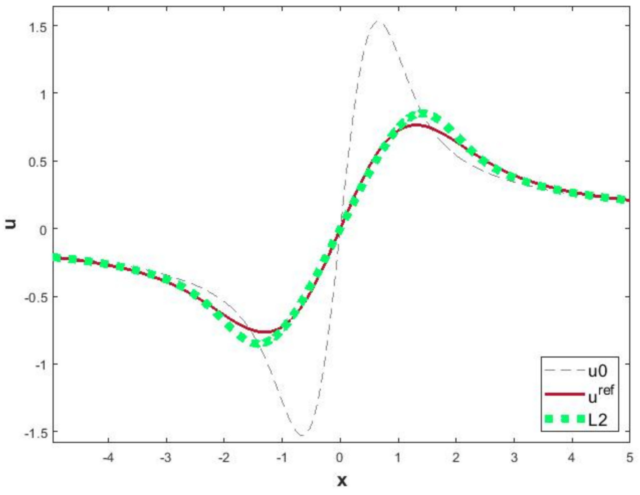


Figure 12. The graphs of the u functions for time-dependent coefficients. Here, u_0 is the initial function $u(x,\ t = 0.5)$, $\mathrm{u}^{\mathrm{ref}}$ is the analytical solution at the final time $t_{\mathrm{fin}} = 1$, and L2 is the corresponding numerical solution served by the L2 (0, 1/2, 1/2, 1/2, 1/2) algorithm for $h = 7.8 \times 10^{-3}$.

3.8. Solution for the Nonlinear Fisher Equation

Here, we examine a nonlinear reaction-diffusion equation, the so-called Fisher equation [27], in the following form:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + \beta u(1-u). \tag{29}$$

This equation is also known as the Kolmogorov–Petrovsky–Piskunov or Fisher-KPP equation [28,29], and was originally developed to describe the spread of advantageous gene variants in space. We found the following analytical solution for $\alpha = 1$ in the literature [30,31]:

$$u^{\mathrm{exact}}(x,t) = \left(1 + e^{\sqrt{\frac{\beta}{6}}x - \frac{5}{6}\beta t}\right)^{-2}. \tag{30}$$

As in Section 3.7, we use this solution to gain the initial condition:

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

and the Dirichlet boundary conditions at the ends of the interval:

$$u(x = x_1, t) = \left(1 + e^{\sqrt{\frac{\beta}{6}}x_1 - \frac{5}{6}\beta t}\right)^{-2}, \text{ and } u(x = x_2, t) = \left(1 + e^{\sqrt{\frac{\beta}{6}}x_2 - \frac{5}{6}\beta t}\right)^{-2}.$$

In principle, the nonlinear term $\beta u(1-u)$ can be incorporated into the schemes in many different ways. We chose the following semi-implicit treatment:

$$u_i^{\mathrm{new}} = u_i^{\mathrm{pred}} + \beta u_i^{\mathrm{pred}}(1 - u_i^{\mathrm{new}})h,$$

which can be rearranged into a fully explicit form:

$$u_i^{\mathrm{new}} = \frac{1+\beta h}{1+\beta h u_i^{\mathrm{pred}}} u_i^{\mathrm{pred}}, \tag{31}$$

where u_i^{pred} is the most recent value of u_i after performing the previous stage calculations which describe the diffusion term. The operation (31) is performed in two extra, separate stages, where a loop goes through all of the nodes (both odd and even). The first extra stage is after Stage 1, whereas the second is after (the original) Stage 4, at integer time levels, at the same moments in which the boundary conditions are recalculated.

With this procedure, we observed that the new methods are very stable for Fisher's equation and behave very similarly to the linear case. We solved Equation (29) for several different values of the parameters β, x_1, x_2, t_{fin}, and Δx. We present the L_∞ errors as a function of the time step size for $\beta = 6, x_1 = 0, x_2 = 4$, $t_{\text{fin}} = 1$, and $\Delta x = 0.01$ in Figure 13, but we must note that very similar curves were produced for other values of the parameters.

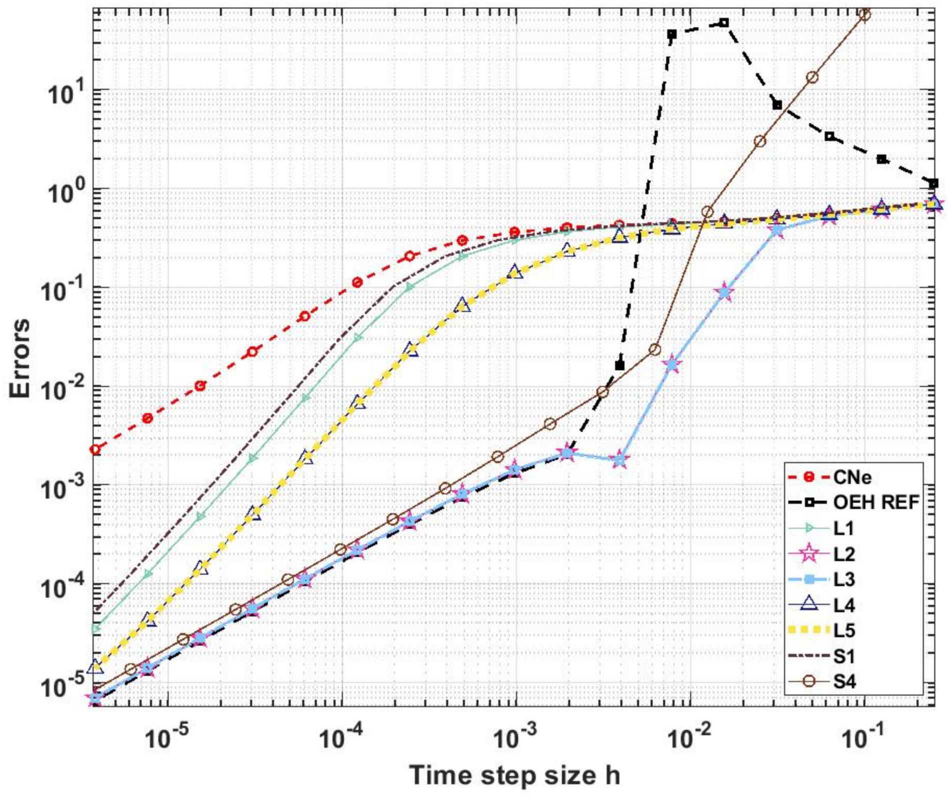


Figure 13. The L_∞ errors as a function of time step size h for the numerical solutions of Fisher's equation for $\alpha = 1$ and $\beta = 2$.

We underline that the goal of this subsection is not the systematic investigation of the new methods in the case of nonlinear equations, which is planned to be performed in our future works. Here, we only show that these leapfrog-hopscotch methods can be successfully applied to nonlinear equations, even if the coefficient of the nonlinear term is large.

3.9. Analytical Investigations

We start with the analysis of the convergence properties of the five most advantageous methods in the one-dimensional case for constant values of mesh ratio r, that is, when the equidistant spatial discretization is fixed.

Theorem 1. *The order of convergence of the L1...L5 leapfrog-hopscotch algorithms is two for the system (6) of linear ODEs:*

$$\frac{d\vec{u}}{dt} = M\vec{u},\ \vec{u}(t=0) = \vec{u}^{\,0}, \tag{32}$$

where M is introduced in (7) and $\vec{u}^{\,0}$ *is an arbitrary initial vector.*

The proof of Theorem 1 is presented in Appendix A.

When analyzing the stability of the methods, we distinguish two types of algorithms. In the first type, only the CNe and the $\theta = 0$ formulas are present, in which the new u_i^{n+1} values are the convex combinations of the old u_j^n values, see Remark 1. We also recall a lemma [32] (p. 28) stating that a convex combination of convex combinations is again a convex combination. These lemmas imply the following:

Corollary 2. *The final u_i^T values are the convex combinations of the initial u_j^0 values in the case of all leapfrog-hopscotch algorithms containing only the CNe and the $\theta = 0$ formulas.*

This means that all of these algorithms, and particularly L1 (C, C, C, C, C), are positivity preserving (more precisely, they satisfy the minimum and maximum principle), which is a significantly stronger property than mere unconditional stability.

By comparison, the L2–L5 algorithms contain the theta-formula for $\theta \neq 0$; thus, they are not positivity preserving. We examine the stability through the eigenvalues of their amplification matrixes. Because Stage 0 is not repeated, it cannot affect stability. The smallest repeating unit in the L2 (0, 1/2, 1/2, 1/2, 1/2) and L3 (1/5, 1/2, 1/2, 1/2, 1/2) combinations consists of only two stages containing the $\theta = \frac{1}{2}$ formula; therefore, it is sufficient to construct them in a matrix form. If we denote the matrices of the $\theta = \frac{1}{2}$ formula acting on the odd and even nodes by H_o and H_e, respectively, then these matrices have the following elements:

$$\begin{gathered} H_o(2i,2i) = H_e(2i+1,2i+1) = 1\,,\ \ H_o(2i+1,2i+1) = H_e(2i,2i) = \tfrac{1-r}{1+r}\,, \\ H_o(2i+1,2i+2) = H_o(2i-1,2i) = H_e(2i,2i\pm 1) = \tfrac{r}{1+r}\,,\ i = 1,\ldots,N-1 \end{gathered} \tag{33}$$

and periodic boundary conditions are taken into account while calculating the first and the last row. Now, the amplification matrix $H_2 = H_o H_e$ has the following elements:

$$\begin{gathered} H_2(2i,2i) = \tfrac{1-r}{1+r}\,,\ \ H_2(2i,2i\pm 1) = \tfrac{r}{1+r}\,,\ \ H_2(2i+1,2i+1) = \tfrac{1-r}{1+r} + \tfrac{2r^2}{(1+r)^2}\,, \\ H_2(2i+1,2i+1\pm 1) = \tfrac{r(1-r)}{(1+r)^2}\,,\ \ H_2(2i+1,2i+1\pm 2) = \tfrac{r^2}{(1+r)^2}\,,\ i = 1,\ldots,N-1 \end{gathered}$$

Using the Maple software we analytically calculated the absolute values of the eigenvalues of this matrix for several values of N and found that none of them exceeds 1, which implies stability for an arbitrary time step size. As an illustration, we present these in Figure 14 as a function of the mesh ratio r, for N = 20.

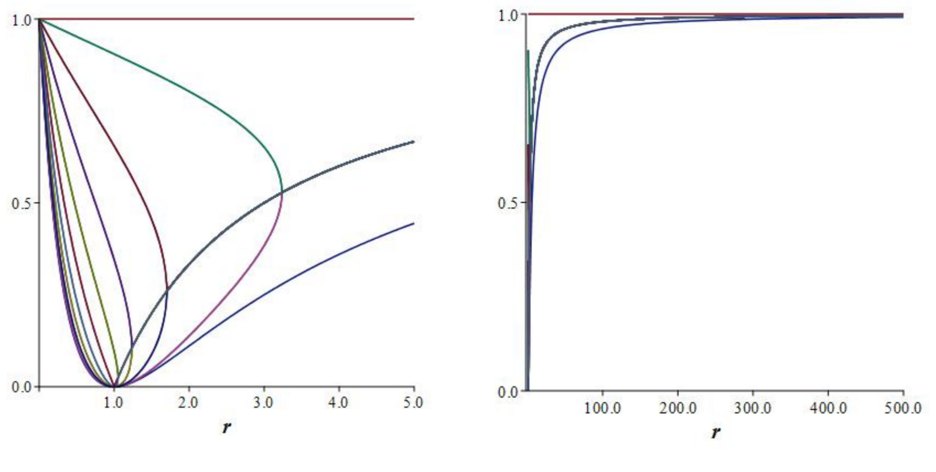


Figure 14. The graphs of the eigenvalues of the amplification matrix H_2 as a function of the mesh ratio r for N = 20. (**a**) $r \in [0,5]$. (**b**) $r \in [1,500]$.

We also constructed the amplification matrix of the L4 (1/4, 1/2, C, 1/2, 1/2) and L5 (1/5, 1/2, C, 1/2, 1/2) combinations by the multiplication of the appropriate four matrices of Stages 1–4 in a similar manner, and obtained essentially the same results. These calculations provide strong evidence that these algorithms are unconditionally stable.

4. Discussion and Summary

The present paper is the natural continuation of our previous work on explicit algorithms for solving the time-dependent diffusion equation. In the current work, we combined the hopscotch space structure with leapfrog time integration. By applying the well-known theta method with nine different values of θ and the recently invented CNe method, we constructed 10^5 combinations. Via subsequent numerical experiments, this huge number was decreased by excluding the combinations that underperformed and, finally, only the top five of these were retained. We used two two-dimensional stiff systems containing 10,000 cells with completely discontinuous random parameters and initial conditions, and presented the results of these five algorithms, in addition to those of several other methods, for these systems. For the purpose of verification, two analytical solutions were used. One was reproduced using a non-equidistant grid. The second was a completely new solution containing the Kummer *M* function for time-dependent thermal diffusivity, presented here for the first time. We also solved the nonlinear Fisher equation and demonstrated that our algorithms can handle the nonlinearity of this equation without losing stability. Then, we provided exact proof that the top five new methods have second-order convergence for fixed spatial discretization, and also showed by analytical calculations that they have very good stability properties.

We found that, in general, the L2 (0, 1/2, 1/2, 1/2, 1/2) combination is the most competitive. This combination yields accurate results orders of magnitude faster than the well-optimized MATLAB routines, and it is more accurate than all of the other examined explicit and stable methods. Although, unlike the L1 (C, C, C, C, C) algorithm, L2 is not positivity preserving, it is also surprisingly robust for relatively large time step sizes, which is not that case for the original OEH algorithm. Moreover, this new L2 algorithm is easy to implement and requires an even smaller amount of function evaluation and computer memory than the conventional explicit second-order Runge–Kutta methods, such as the Heun method. We can conclude that it combines has the most important advantages of the standard explicit and the implicit methods.

Our next aim is to search for more accurate algorithms for nonlinear parabolic equations. We plan to construct new analytical solutions under circumstances in which the heat conducting coefficients depend on space, and use these to test both the new and old methods. These results will be of significant interest in the simulation of real problems, such as in the field of engineering, the heat transfer by conduction, and the radiation in elements of machine tools.

Author Contributions: Conceptualization and methodology, E.K.; software, Á.N. and I.O.; validation, E.K., H.K. and I.O.; formal analysis, I.F.B. (analytical solutions), E.K. (analytical proofs); investigation, Á.N., I.O. and H.K.; resources, E.K. and G.B.; data curation, Á.N.; writing—original draft preparation, E.K. and I.F.B.; writing—review and editing, G.B., Á.N.; visualization, Á.N. and H.K.; supervision, E.K. and G.B.; project administration, E.K. and Á.N.; funding acquisition, G.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Project No. 129257 implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the K_18 funding scheme.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is available at the following link https://github.com/Drendre/Leapfrog-hopscotch-data (accessed on 16 August 2021).

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

Appendix A

Proof of Theorem 1. We start from the Taylor series of u_i:

$$u_i(t+h) \approx u_i + hu_i{}' + \frac{h^2}{2}u_i{}'' \tag{A1}$$

where all u terms on the right-hand side are taken at time t. Let us substitute the spatially discretized ODE:

$$u_i{}' = \alpha\frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2}, \tag{A2}$$

to obtain:

$$\begin{aligned} &u_i(t+h) \approx u_i + h\alpha\tfrac{u_{i-1}-2u_i+u_{i+1}}{\Delta x^2} + \tfrac{h^2}{2}\alpha\tfrac{u_{i-1}{}'-2u_i{}'+u_{i+1}{}'}{\Delta x^2} = \\ &u_i + r(u_{i-1} - 2u_i + u_{i+1}) + \tfrac{r}{2}h\left(\alpha\tfrac{u_{i-2}-2u_{i-1}+u_i}{\Delta x^2} + \alpha\tfrac{u_{i-1}-2u_i+u_{i+1}}{\Delta x^2} + \alpha\tfrac{u_i-2u_{i+1}+u_{i+2}}{\Delta x^2}\right) \end{aligned} \tag{A3}$$

After simplification one obtains:

$$u_i(t+h) \approx u_i + r(u_{i-1} - 2u_i + u_{i+1}) + \frac{r^2}{2}[6u_i - 4(u_{i-1} + u_{i+1}) + (u_{i-2} + u_{i+2})], \tag{A4}$$

Thus, for a double time step:

$$u_i^{n+2} \approx \left(1 - 4r + 12r^2\right)u_i^n + \left(2r - 8r^2\right)\left(u_{i-1}^n + u_{i+1}^n\right) + 2r^2\left(u_{i-2}^n + u_{i+2}^n\right). \tag{A5}$$

We show that the numerical solution is identical to (A5). We use series expansions up to the second order and, for the sake of brevity, we omit the higher-order terms; thus, for example, we write $e^{-x} \approx 1 - x + x^2/2$, etc. Starting with the L1 combination, (13) can be written as:

$$u_i^{n+1} = \left(1 - 2r + 2r^2\right)u_i^n + \left(r - r^2\right)\left(u_{i-1}^{n+1/2} + u_{i+1}^{n+1/2}\right) \tag{A6}$$

Let us calculate the value of u at the next time step:

$$\begin{aligned} &u_i^{n+2} = (1-2r+2r^2)u_i^{n+1} + (r-r^2)\left(u_{i-1}^{n+1+1/2} + u_{i+1}^{n+1+1/2}\right) = \\ &(1-2r+2r^2)\left[(1-2r+2r^2)u_i^n + (r-r^2)\left(u_{i-1}^{n+1/2} + u_{i+1}^{n+1/2}\right)\right] + (r-r^2)\left(u_{i-1}^{n+1+1/2} + u_{i+1}^{n+1+1/2}\right) \approx \\ &(1-4r+8r^2)u_i^n + (r-3r^2)\left(u_{i-1}^{n+1/2} + u_{i+1}^{n+1/2}\right) + (r-r^2)\left(u_{i-1}^{n+1+1/2} + u_{i+1}^{n+1+1/2}\right) . \end{aligned} \tag{A7}$$

We also repeat the same calculation for the left neighbors:

$$\begin{aligned} &u_{i-1}^{n+1/2} = (1-2r+2r^2)u_{i-1}^{n-1/2} + (r-r^2)\left(u_{i-2}^n + u_i^n\right) \\ &u_{i-1}^{n+1+1/2} = (1-2r+2r^2)u_{i-1}^{n+1/2} + (r-r^2)\left(u_{i-2}^{n+1} + u_i^{n+1}\right) = \\ &(1-2r+2r^2)\left[(1-2r+2r^2)u_{i-1}^{n-1/2} + (r-r^2)\left(u_{i-2}^n + u_i^n\right)\right] + \\ &(r-r^2)\left[(1-2r+2r^2)u_{i-2}^n + (r-r^2)\left(u_{i-3}^{n+1/2} + u_{i-1}^{n+1/2}\right) + (1-2r+2r^2)u_i^n + (r-r^2)\left(u_{i-1}^{n+1/2} + u_{i+1}^{n+1/2}\right)\right] = \\ &(1-4r+8r^2)u_{i-1}^{n-1/2} + (r-3r^2)\left(u_{i-2}^n + u_i^n\right) + (r-3r^2)\left(u_{i-2}^n + u_i^n\right) + r^2\left(u_{i-3}^{n+1/2} + 2u_{i-1}^{n+1/2} + u_{i+1}^{n+1/2}\right) = \\ &(1-4r)u_{i-1}^{n-1/2} + 2r\left(u_{i-2}^n + u_i^n\right) + O\left(r^2\right) . \end{aligned} \tag{A8}$$

The right neighbors can be treated similarly:

$$\begin{aligned} u_{i+1}^{n+1/2} &= (1-2r+2r^2)u_{i+1}^{n-1/2} + (r-r^2)\left(u_{i+2}^n + u_i^n\right) \\ u_{i+1}^{n+1+1/2} &= (1-4r)u_{i+1}^{n-1/2} + 2r\left(u_{i+2}^n + u_i^n\right) + O\left(r^2\right) . \end{aligned} \tag{A9}$$

Substituting (A8) and (A9) back to (A7) we obtain:

$$
\begin{aligned}
&u_i^{n+2} = (1-4r+8r^2)u_i^n + (r-3r^2)\left(u_{i-1}^{n+1/2}+u_{i+1}^{n+1/2}\right) \\
&+(r-r^2)\left[(1-4r)\left(u_{i+1}^{n-1/2}+u_{i-1}^{n-1/2}\right)+2r\left(u_{i+2}^n+2u_i^n+u_{i-2}^n\right)\right] \approx \\
&(1-4r+8r^2)u_i^n + (r-3r^2)\left(u_{i-1}^{n+1/2}+u_{i+1}^{n+1/2}\right)+(r-5r^2)\left(u_{i+1}^{n-1/2}+u_{i-1}^{n-1/2}\right)+2r^2\left(u_{i+2}^n+2u_i^n+u_{i-2}^n\right) = \\
&(1-4r+12r^2)u_i^n + (r-3r^2)\left(u_{i-1}^{n+1/2}+u_{i+1}^{n+1/2}\right)+(r-5r^2)\left(u_{i+1}^{n-1/2}+u_{i-1}^{n-1/2}\right)+2r^2\left(u_{i+2}^n+u_{i-2}^n\right) .
\end{aligned}
\tag{A10}
$$

The first and the last term on the right-hand side of (A10) is already equivalent to the appropriate terms in (A5). The second and third terms containing the values of the first neighbors need some more manipulation as follows:

$$
\begin{aligned}
&\left.\begin{aligned}
u_{i-1}^{n-1/2} &= u_{i-1}^n - \tfrac{r}{2}\left(u_{i-2}^n-2u_{i-1}^n+u_i^n\right)+O(r^2) \\
u_{i-1}^{n+1/2} &= u_{i-1}^n + \tfrac{r}{2}\left(u_{i-2}^n-2u_{i-1}^n+u_i^n\right)+O(r^2)
\end{aligned}\right\}\pm \\
&u_{i-1}^{n-1/2}+u_{i-1}^{n+1/2} = 2u_{i-1}^n + O(r^2) \\
&u_{i-1}^{n+1/2}-u_{i-1}^{n-1/2} = r\left(u_{i-2}^n-2u_{i-1}^n+u_i^n\right)+O(r^2) .
\end{aligned}
\tag{A11}
$$

Now, we can cope with the second and third term on the right-hand side of (A10):

$$
\begin{aligned}
&(r-3r^2)\left(u_{i-1}^{n+1/2}+u_{i+1}^{n+1/2}\right)+(r-5r^2)\left(u_{i+1}^{n-1/2}+u_{i-1}^{n-1/2}\right) = \\
&(r-4r^2)\left(u_{i-1}^{n+1/2}+u_{i+1}^{n+1/2}+u_{i-1}^{n-1/2}+u_{i+1}^{n-1/2}\right)+r^2\left(u_{i-1}^{n+1/2}+u_{i+1}^{n+1/2}-u_{i-1}^{n-1/2}-u_{i+1}^{n-1/2}\right) = \\
&(2r-8r^2)\left(u_{i-1}^n+u_{i+1}^n\right)+O(r^3) .
\end{aligned}
\tag{A12}
$$

Substituting these back into (A10), one obtains (A5), which completes the proof for the L1 algorithm.

For the theta method, we use the power series expansion $(1+x)^{-1} = 1-x+x^2-\ldots$ to find, for the denominator of Equation (11), that $(1+2r(1-\theta))^{-1} \approx 1-2r(1-\theta)+4r^2(1-\theta)^2$; thus, rather than (A6), we now have:

$$
\begin{aligned}
u_i^{n+1} = &\left[(1-2r\theta)u_i^n + r\left(u_{i-1}^{n+1/2}+u_{i+1}^{n+1/2}\right)\right]\left\{1-2r(1-\theta)+4r^2(1-\theta)^2\right\} \approx \\
&(1-2r+4r^2(1-\theta))u_i^{\mu} + (r-2r^2(1-\theta))\left(u_{i-1}^{n+1/2}+u_{i+1}^{n+1/2}\right) .
\end{aligned}
\tag{A13}
$$

For $\theta = 1/2$, this is the same as (A6); thus, the proof presented above for the L1 method is also valid for the L2…L5 combinations. One can also see that for arbitrary combinations of values of theta, the second-order property does not hold. □

References

1. Saleh, M.; Nagy, Á.; Kovács, E. Construction and investigation of new numerical algorithms for the heat equation: Part 1. *Multidiszcip. Tudományok* **2020**, *10*, 323–338. [CrossRef]
2. Saleh, M.; Nagy, Á.; Kovács, E. Construction and investigation of new numerical algorithms for the heat equation: Part 2. *Multidiszcip. Tudományok* **2020**, *10*, 339–348. [CrossRef]
3. Saleh, M.; Nagy, Á.; Kovács, E. Construction and investigation of new numerical algorithms for the heat equation: Part 3. *Multidiszcip. Tudományok* **2020**, *10*, 349–360. [CrossRef]
4. 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.
5. António, J.; Tadeu, A.; Godinho, L.; Simões, N. Benchmark solutions for three-dimensional transient heat transfer in two-dimensional environments via the time fourier transform. *Comput. Mater. Contin.* **2005**, *2*, 1–11. [CrossRef]
6. Zoppou, C.; Knight, J. Analytical solution of a spatially variable coefficient advection–diffusion equation in up to three dimensions. *Appl. Math. Model.* **1999**, *23*, 667–685. [CrossRef]
7. Kovács, E. A class of new stable, explicit methods to solve the non-stationary heat equation. *Numer. Methods Partial. Differ. Equ.* **2021**, *37*, 2469–2489. [CrossRef]
8. Gordon, P. Nonsymmetric Difference Equations. *J. Soc. Ind. Appl. Math.* **1965**, *13*, 667–673. [CrossRef]
9. Gourlay, A.R. Hopscotch: A Fast Second-order Partial Differential Equation Solver. *IMA J. Appl. Math.* **1970**, *6*, 375–390. [CrossRef]
10. 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]

11. Leapfrog Integration. Available online: https://en.wikipedia.org/wiki/Leapfrog_integration (accessed on 11 July 2021).
12. Hockney, R.W.; Eastwood, J.W. *Computer Simulation Using Particles*; Taylor & Francis: Boca Raton, FL, USA, 1989.
13. Frenkel, D.; Smit, B.; Ratner, M.A. Understanding Molecular Simulation: From Algorithms to Applications. *Phys. Today* **1997**, *50*, 66. [CrossRef]
14. Iserles, A. Generalized Leapfrog Methods. *Ima. J. Numer. Anal.* **1986**, *6*, 381–392. [CrossRef]
15. Hirsch, C. *Numerical Computation of Internal and External Flows, Volume 1: Fundamentals of Numerical Discretization*; Wiley: Hoboken, NJ, USA, 1988.
16. Verwer, J.G.; Sommeijer, B.P. Stability Analysis of an Odd—Even-Line Hopscotch Method for Three-Dimensional Advection—Diffusion Problems. *SIAM J. Numer. Anal.* **1997**, *34*, 376–388. [CrossRef]
17. Holmes, M.H. *Introduction to Numerical Methods in Differential Equations*; Springer: New York, NY, USA, 2007.
18. Munka, M.; Pápay, J. *4D Numerical Modeling of Petroleum Reservoir Recovery*; Akadémiai Kiadó: Budapest, Hungary, 2001.
19. Kovács, E. New stable, explicit, first order method to solve the heat conduction equation. *J. Comput. Appl. Mech.* **2020**, *15*, 3–13. [CrossRef]
20. Muñoz-Matute, J.; Calo, V.M.; Pardo, D.; Alberdi, E.; Van Der Zee, K.G. Explicit-in-time goal-oriented adaptivity. *Comput. Methods Appl. Mech. Eng.* **2018**, *347*, 176–200. [CrossRef]
21. Heun's Method—Wikipedia. Available online: https://en.wikipedia.org/wiki/Heun%27s_method (accessed on 30 July 2021).
22. Mátyás, L.; Barna, I.F. General self-similar solutions of diffusion equation and related constructions. *arXiv* **2021**, arXiv:2104.09128v1.
23. Sedov, L.I. *Similarity and Dimensional Methods in Mechanics*; CRC Press: Boca Raton, FL, USA, 2018.
24. Barna, I.F.; Kersner, R. Heat conduction: A telegraph-type model with self-similar behavior of solutions. *J. Phys. A Math. Theor.* **2010**, *43*, 375210. [CrossRef]
25. Barna, I.F.; Bognár, G.; Guedda, M.; Mátyás, L.; Hriczó, K. Analytic self-similar solutions of the kardar-parisi-zhang interface growing equation with various noise terms. *Math. Model. Anal.* **2020**, *25*, 241–256. [CrossRef]
26. Olver, F.W.J.; Lozier, D.W.; Boisvert, R.F.; Clark, C.W. *NIST Handbook of Mathematical Functions*; Cambridge University Press: New York, NY, USA, 2011.
27. Fisher, R.A. The wave of advance of advantageous genes. *Ann. Eugen.* **1937**, *7*, 355–369. [CrossRef]
28. Kolmogorov, A.; Petrovskii, I.; Piscunov, N. A study of the equation of diffusion with increase in the quantity of matter, and its application to a biological problem. *Bull. Mosc. Univ. Math. Mech.* **1937**, *1*, 1–25.
29. Li, Y.; Van Heijster, P.; Marangell, R.; Simpson, M.J. Travelling wave solutions in a negative nonlinear diffusion–reaction model. *J. Math. Biol.* **2020**, *81*, 1495–1522. [CrossRef] [PubMed]
30. Bastani, M.; Salkuyeh, D.K. A highly accurate method to solve Fisher's equation. *Pramana. J. Phys.* **2012**, *78*, 335–346. [CrossRef]
31. 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.* **2019**, *2019*, 146. [CrossRef]
32. Hiriart-Urruty, J.-B.; Lemaréchal, C. *Fundamentals of Convex Analysis*; Springer: Berlin, Germany, 2001.