Multidiszciplináris Tudományok, 14. kötet, 1. sz. (2024) pp. 46–59. **https://doi.org/10.35925/j.multi.2024.1.5**

COMPREHENSIVE INVESTIGATION OF THE EXPLICIT, POSITIVITY PRESERVING METHODS FOR THE HEAT EQUATION

Part 1

Husniddin Khayrullaev
PhD student, Institute of Physics and Electric Engineering, University of Miskolc
3515 Miskolc-Egyetemváros, e-mail: hxayrullayev@mail.ru

Endre Kovács
associate professor, Institute of Physics and Electric Engineering, University of Miskolc
3515 Miskolc-Egyetemváros, e-mail: endre.kovacs@uni-miskolc.hu

Abstract
In this paper-series, we investigate the performance of 12 explicit non-conventional algorithms. All of them have the convex combination property, thus they are unconditionally stable and preserve the positivity of the solution when they applied to the heat equation. In this part of the series, we construct several 2D systems to find how the errors depend on the time step size. Sweeps for other key parameters will be presented in the next part of the series.

Keywords: *explicit numerical methods, unconditional stability, heat equation, parabolic PDEs*

1. Introduction and the studied problem

In recent years, our research group developed new numerical algorithms for the heat conduction or diffusion equation and similar diffusion-reaction equations. The conduction of heat and many other diffusion-like phenomena can be modelled by the following partial differential equation (PDE), the heat equation:

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

where u is the temperature and α is the thermal diffusivity. However, the problem of heat conduction and diffusion arises in this very simple form very rarely. The general form of the heat equation is the following

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

where $u = u(\vec{r}, t)$ is the unknown function, while $k = k(\vec{r}, t)$, $c = c(\vec{r}, t)$, $\rho = \rho(\vec{r}, t)$ are the heat conductivity, the specific heat, and the (mass) density, respectively. The relation $\alpha = k/(c\rho)$ connects these non-negative quantities, but apart from this, the α, k, c, ρ are arbitrary functions. Moreover, several nonlinear equations, such as the Fisher, the Huxley, the Kardar-Parisi-Zhang and the FitzHugh-Nagumo (Agbavon and Appadu, 2020) equations, contain a diffusion term besides the nonlinear reaction term.

These equations have a few analytical solutions, old as well as new ones (Mátyás and Barna, 2022), but when the material properties are non-continuous functions of space, as in most cases in engineering, numerical integration is necessary to solve *Equation (2)*. Although many efficient methods are proposed and tested recently (Mbroh and Munyakazi, 2021; Ali et al., 2021), the numerical solution of the equation is still a challenging problem in the case of large systems. The usual finite difference methods are either explicit or implicit methods, and both groups have advantages and disadvantages. Explicit algorithms are easier to code and parallelize, moreover, they are faster for the same time step size than implicit methods, but they are considered as less reliable. This is because the majority of them are only conditionally stable, which means that if the time step size is below a certain threshold, sometimes called the CFL limit, then instabilities necessarily appear. Implicit methods require the solution of a system of algebraic equations at each time step, which can be slow and memory-demanding in the case of large matrices.

It is well known that the true solution of the heat or diffusion equation always follows the maximum and minimum principles (Holmes, 2007) (p. 87) reflecting the Second law of thermodynamics. Numerical schemes should also have this property or, at least, should preserve the positivity of the solution, because the modelled quantities, e.g. the temperature, concentrations of chemical species, size of populations or number of particles, are positive. However, most well-known finite difference or finite element methods do not necessarily produce non-negative solutions. Even if they usually yield positive values, sometimes the solution can go below zero and then it can start non-physical oscillations. That is why some scholars have investigated unconditionally positive schemes. The first example of these algorithms was the so-called UPFD (unconditionally positive finite-difference) method of Chen-Charpentier et al. (Chen-Charpentier and Kojouharov, 2013), developed for the advection-diffusion-reaction equation. This was then tested for several systems by (Appadu, 2017; Drljača and Savović, 2019). Then, the so-called enhanced unconditionally positive finite difference method (Ndou et al., 2022) was introduced, which employs proper orthogonal decomposition.

This work, consisting of two papers, can be considered as the continuation of our previous paper (Omle et al., 2022), in which we tested some recent numerical algorithms to solve the heat or diffusion equation. Since then, we have created numerical methods with third- and fourth-order accuracy (Kovács et al., 2024). In this work, we perform more extensive and systematic tests to examine how the performance of the methods changes and which of them is the best choice under various circumstances. All of the examined algorithms fulfil the maximum and minimum principles, and sometimes, for the sake of brevity, we call them positivity preserving methods.

In Part 1 of this paper-series, we describe the discretization and the 12 used numerical methods. Then we perform numerical experiments in stiff and non-stiff 2D systems to examine the numerical error as a function of the time step size and the running time. Part 2 will present five further numerical tests with a parameter sweep for the stiffness ratio and other key parameters and then our conclusions and recommendations about which numerical techniques should be used in different cases.

2. Description of the space-discretization

Let us first present the discretization in the case of a one-dimensional homogeneous system. In the case of an equidistant spatial grid with nodes $x_i, i=1,...,N$ fixed, where N denotes the number of grid points, the central difference formula is the most common starting point:

$$\frac{\partial^2}{\partial x^2} f(x_\mathrm{i},t_\mathrm{j}) \approx \frac{\dfrac{f(x_{\mathrm{i}+1},t_\mathrm{j}) - f(x_\mathrm{i},t_\mathrm{j})}{\Delta x} + \dfrac{f(x_{\mathrm{i}-1},t_\mathrm{j}) - f(x_\mathrm{i},t_\mathrm{j})}{\Delta x}}{\Delta x}.$$

Using this for the spatial derivative we obtain the semi-discretized form of *Equation (1)*

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

The system matrix *M* has the following elements:

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

thus is tridiagonal in the one-dimensional case. The matrix-form of equation-system (3) is:

$$\frac{d\vec{u}}{dt} = M\vec{u}. \tag{4}$$

We approach the discretization of *Equation (2)* by considering a one-dimensional, equidistant grid first, but with space-dependent material properties. In this case can write

$$c(x)\rho(x)\left.\frac{\partial u}{\partial t}\right|_x = \frac{1}{\Delta x}\left[k\left(x + \frac{\Delta x}{2}\right)\frac{u(x+\Delta x) - u(x)}{\Delta x} + k\left(x - \frac{\Delta x}{2}\right)\frac{u(x-\Delta x) - u(x)}{\Delta x}\right].$$

The next point is the change to cell variables, where the subscripts refer to whole cells:

$$\frac{du_i}{dt} = \frac{S}{c_i \rho_i S \Delta x}\left(k_{i,i+1}\frac{u_{i+1} - u_i}{\Delta x} + k_{i-1,i}\frac{u_{i-1} - u_i}{\Delta x}\right),$$

where u_i is the temperature of the cell indexed by *i*, $C_i = c_i m_i = c_i \rho_i V_i$ is the heat capacity, Δx is the length, *m* is the mass, *S* is the cross-section area, while $V_i = S_i \Delta x_i$ is the volume of the cell, respectively. The thermal resistance between the cell *i* and its arbitrary neighbour *j* is $R_{ij} = \dfrac{\Delta x}{k_{ij} S}$. The time-development of each cell-temperature is now determined by the equation:

$$\frac{du_i}{dt} = \frac{u_{i-1} - u_i}{R_{i,i-1} C_i} + \frac{u_{i+1} - u_i}{R_{i,i+1} C_i},$$

which can be straightforwardly generalized to two dimensions. Indeed, if we have a rectangular grid with $N_x \times N_y$ elements and the numbering of the cells is along the *x* direction from 1 to N_x, etc., then

$$\frac{du_i}{dt} = \sum_{j \in \left\{\substack{i-1,\, i+1, \\ i-Nx,\, i+Nx}\right\}} \frac{u_j - u_i}{R_{i,j} C_i}. \tag{5}$$

The equation system above can, of course, be written into a similar matrix form as (4). Further details on this manner of discretization can be found e.g. in (Nagy et al., 2021a). From this point we will use this capacity-resistivity model to simulate heat conduction in 2D. Random values will be generated for the heat capacities and the resistances with a log-uniform distribution using the formulas:

$$C_i = 10^{(a_C - b_C \times rand)},\ R_{x,i} = 10^{(a_{Rx} - b_{Rx} \times rand)},\ R_{y,i} = 10^{(a_{Ry} - b_{Ry} \times rand)}\ . \tag{6}$$

If one varies the *a* and *b* parameters, one can produce highly different systems. Here *rand* is a random number generated by the MATLAB uniformly in the (0,1) interval for each quantity. The running times will be measured using the tic-toc command of MATLAB. However, in this way, not the CPU time is measured which is necessary to the execution of the algorithms, but the elapsed time, which can be larger since other processes are running at the background, which can cause random fluctuations in the measured running times. To get rid of the effect of these, the calculations will be performed *Nrun* times subsequently, and then the average running times will be calculated.

In this work, the simplest zero Neumann boundary conditions will be assumed in all cases, which is equivalent to thermal isolation. Because of this, the matrix *M* has a zero eigenvalue due to the conservation of heat. All other eigenvalues are negative, which is necessary to fulfil the Second law of thermodynamics. Let us denote the smallest (largest) absolute value eigenvalues with λ_{MIN} (λ_{MAX}), where, of course, the zero eigenvalue is excluded. The stiffness ratio is usually given as $Sr = \lambda_{\mathrm{MAX}} / \lambda_{\mathrm{MIN}}$. For the standard explicit (Euler) method, the maximum possible time step size or CFL limit is $h_{\mathrm{MAX}}^{\mathrm{EE}} = \left|2 / \lambda_{\mathrm{MAX}}\right|$, and a similar threshold is valid for other conventional explicit methods.

3. Presentation of the tested methods and their properties

The time is discretized uniformly, i.e. $t \in \left[t^0, t_{\mathrm{fin}}\right]$ and $t^n = t^0 + nh$, $n = 1,...,T$, $hT = t_{\mathrm{fin}} - t^0$. The numerical methods for the general linear heat *Equation (2)* can be simply presented if we introduce the following notations

$$r_i = h\sum_{j \neq i} \frac{1}{C_i R_{ij}} = h m_{ii} \text{ and } A_i = h\sum_{j \neq i} \frac{u_j^n}{C_i R_{ij}} = h\sum_{j \neq i} m_{ij} u_j^n\ . \tag{7}$$

The first quantity is the generalization of the widely used mesh ratio, which, for a one-dimensional equidistant mesh with constant diffusivity, has the form $r = \frac{\alpha h}{\Delta x^2} = -\frac{m_{ii} h}{2} > 0,\ 0 < i < N-1$. Therefore, in this most simple case

$$r_i = 2r \text{ and } A_i = r\left(u_{i-1}^n + u_{i+1}^n\right), \text{ thus } \frac{A_i}{r_i} = \frac{u_{i-1}^n + u_{i+1}^n}{2}\ ,$$

while, in a 2D equidistant mesh

$$r_i = 4r \text{ and } A_i = r\left(u_{i-1}^n + u_{i+1}^n + u_{i-Nx}^n + u_{i+Nx}^n\right)\ ,$$

thus

$$\frac{A_i}{r_i} = \frac{u_{i-1}^n + u_{i+1}^n + u_{i-Nx}^n + u_{i+Nx}^n}{4}.$$

Due to the simplicity of these relations, we present only the generalized formula of the numerical algorithms.

3.1. The applied 12 convex-combination scheme for the diffusion equation

1. The UPFD method (Chen-Charpentier and Kojouharov, 2013) is a simple one-stage scheme. For *Equation (2)*, it is defined by the formula:

$$u_i^{n+1} = \frac{u_i^n + A_i}{1 + 2r_i}.$$

2. The constant neighbour (CNe) method (Kovács, 2020a; Kovács, 2020b) is a little bit similar, but contains the *r* quantities in the exponents:

$$u_i^{n+1} = u_i^n \cdot e^{-r_i} + \frac{A_i}{r_i}\left(1 - e^{-r_i}\right).$$

In the case of one-stage methods 1 and 2, the successive displacement implementation is used to enhance accuracy and speed. This means that when the program goes through the mesh points one by one, and calculates the new value u_i^{n+1}, it uses the already obtained value u_{i-1}^{n+1}, and, in 2D, u_{i-Nx}^{n+1} as well.

3. The LH-CNe (Nagy et al., 2021b) is member of the family of odd-even hopscotch methods, thus the space must be discretized by a special, so-called bipartite grid, where the cells are labelled as odd and even. It is necessary that all the nearest neighbours of the even nodes are odd, and vice versa, like in a checkerboard. The leapfrog-hopscotch (LH) is a space-time structure as follows: a half time step comes first for e.g. the even cells, when only the values at the beginning of the time step are used. After this, full-size time steps must be taken strictly alternately for the odd and even cells until we approach the end of the simulation. The last timestep have to be halved for the even cells to span exactly the same simulation time as for the odd cells. It is essential to use the latest values of the neighbours when a new value of u_i is calculated. In the case of the LH-CNe method investigated in this work, the CNe formula is employed in each stage with the appropriate (half or full) time step size. For the sake of programming simplicity and low running times, the number of cells in the *x* (horizontal) direction was always odd. With this convention, the vertical (*z*-directional) neighbours u_{i-Nx} and u_{i+Nx} of the odd cells are automatically even. Without this, the parity of each cell must be checked in each stage, which would increase running times.

4. The CpC method (Kovács et al., 2021) calculates new predictor values of the temperature with the CNe formula, but with a *h*/2 time step:

$$u_i^{\text{pred}} = u_i^n e^{-r_i/2} + \frac{A_i}{r_i}\left(1 - e^{-r_i/2}\right).$$

The second stage takes a full time step size corrector step using the CNe formula of point 2 again. Thus, the final values at the end of the time step are:

$$u_i^{n+1} = u_i^n e^{-r_i} + \frac{A_i^{\text{pred}}}{r_i}\left(1-e^{-r_i}\right),$$

where the predictor values provided by the first stage are used to obtain the A_i^{pred} quantities.

5. The linear-neighbour (LNe) method has also a predictor and a corrector stage (Kovács, 2020b). Its first stage uses the CNe method to obtain the new u_i^{pred} values valid at the final time of the actual time step. Using these predictor values we can calculate

$$A_i^{\text{pred}} = h\sum_{j\neq i}\frac{u_j^{\text{pred}}}{C_i R_{ij}}.$$

Then, in the second stage, the corrector values are obtained as follows:

$$u_i^{n+1} = u_i^n e^{-r_i} + \left(A_i - \frac{A_i^{\text{pred}} - A_i}{r_i}\right)\frac{1-e^{-r_i}}{r_i} + \frac{A_i^{\text{pred}} - A_i}{r_i}.$$

6. The values provided in the LNe corrector stage can be used to calculate A_i^{pred} again, thus we can repeat the predictor stage to refine the results. In this case, there are three stages altogether, thus the algorithm is abbreviated as LNe3 (Kovács, 2020b).

7–8. The LNe4 and the LNe5 algorithms are four- and five-stage algorithms, which are obtained by continuing the iteration explained in the previous point after the calculations of the LNe3 and the LNe4 scheme, respectively. Unfortunately, these iterations do not improve the order of convergence, but increase the accuracy of the results. The question we want to answer in this work is that in which cases it is worth to iterate.

9. The three-stage Constant-Linear-Quadratic neighbour (CLQ) method (Kovács et al., 2024) has two stages first, which are the same as those of the LNe method. However, the second, LNe stage values has to be calculated with not only a full, but a half time step size as well. Let us denote these values by u_i^{L} and $u_i^{\text{L}\frac{1}{2}}$, respectively. Using them we can calculate $A_i^{\text{pred,L}}$ and $A_i^{\text{pred,L}\frac{1}{2}}$ such as above, and then the quantities $S_i = 4A_i^{\text{pred,L}\frac{1}{2}} - A_i^{\text{pred,L}} - 3A_i$ and $W_i = 2\left(A_i^{\text{pred,L}} - 2A_i^{\text{pred,L}\frac{1}{2}} + A_i\right)$, where A_i is already calculated at the beginning of the first stage. The final temperatures at the end of the time step are given by the formula

$$u_i^{\text{Q}} = e^{-r_i}u_i^n + \frac{1-e^{-r_i}}{r_i}\left(\frac{2W_i}{r_i^2} - \frac{S_i}{r_i} + A_i\right) + \frac{W_i\left(1-2/r_i\right)+S_i}{r_i}.$$

10. The above obtained CLQ function values can be used to add one more stage, with which we have a four-stage algorithm called the CLQ2 method. To make it possible, the midpoint values must be calculated at the third stage:

$$u_i^{Q\frac{1}{2}} = e^{-r_i/2}u_i^n + \frac{1-e^{-r_i/2}}{r_i}\left(\frac{2W_i}{r_i^2} - \frac{S_i}{r_i} + A_i\right) + \frac{W_i}{4r_i} - \frac{W_i}{r_i^2} + \frac{S_i}{2r_i}\ .$$

In Stage 4, we use $A_i^{\text{pred},Q}$, $A_i^{\text{pred},Q\frac{1}{2}}$ to obtain the new values of *S* and *W* and then we repeat what we have done in third stage.

11–12. The iteration of point 10 is further repeated. In this way, we obtain the CLQ3 scheme (5 stages altogether), as well as the CLQ4 scheme (6 stages altogether) (Kovács et al., 2024).

The order of convergence is one for the UPFD and the CNe algorithms, two for the LH-CNe, CpC, and LNe-LNe5 algorithms, three for the CLQ and four for the CLQ2-4 methods. Their outstanding stability is the consequence of the convex combination property, which have been analytically proved in the original publications mentioned above.

Theorem 1. *If schemes 1–12 are applied to the spatially discretized linear heat Equation (3), then the new* u_i^{n+1} *temperature values are the convex combinations of the initial values* u_j^0. *The same is true for the methods 1–8 when are applied to the more general Equation (5).*

According to a large number of our numerical experiments, the convex combination property holds for the CLQ-CLQ4 algorithms in the case of *Equation (5)* as well, but this has not been proved analytically.

Corollary 1: The algorithms are not only preserve positivity, but satisfy the Maximum and Minimum principle (Holmes, 2007) (p. 87). It means that the extreme values of the function *u* occur among the initial or the boundary values, which is a physical property of heat conduction (without external heat sources). This also implies the stability of the algorithms for any time step sizes.

The corollary above means that the CFL limit does not affect the stability of the methods. However, it does affect their accuracy, and this is the main question we would like to numerically investigate in the remaining part of the paper.

4. Numerical experiments

We perform numerical case studies to test the above defined numerical methods. We calculate the error as the maximum of the absolute value of the difference between the reference temperature $u_{\mathrm{i}}^{\mathrm{ref}}$ and the temperature $u_{\mathrm{i}}^{\mathrm{num}}$ obtained by the studied numerical method at final time t_{fin}, the end of the examined time interval:

$$Error = \max_i \left| u_{\mathrm{i}}^{\mathrm{ref}}(t_{\mathrm{fin}}) - u_{\mathrm{i}}^{\mathrm{num}}(t_{\mathrm{fin}}) \right|\ .$$

The reference solution is a numerical solution obtained by applying the MATLAB ode15 solver with absolute and relative tolerances below 10^{-10}. We examine the errors of the numerical algorithms as a function of the time step size *h* and the running time. The results will be displayed in log-log diagrams. When the running times are measured, the calculations are performed *Nrun* > 1 times and the average of these *Nrun* running times are considered to reduce random fluctuations.

4.1. Case study 1: long-wave initial function

We solve the spatially discretized PDE (1) on a square-shaped system with $N_x = 41,\ N_y = 41$, and $t_{\mathrm{fin}} = 2.4$. All *a* and *b* exponents in *Equation (6)* are zero, which means $C = 1,\ R_x = R_z = 1$ for all cells, which yields $Sr = 1.36 \cdot 10^3,\ h_{\mathrm{MAX}}^{\mathrm{EE}} = 0.25$. The initial condition is the product of two cosine functions with wavelengths equivalent to the system size:

$$u(x, y, t = 0) = \big(\cos(2\pi x) - 1\big)\big(\cos(2\pi y) - 1\big) / 4\,. \tag{8}$$

The results of this numerical experiment are presented in *Figure 1* and *2*, where *Nrun* = 25 has been used.

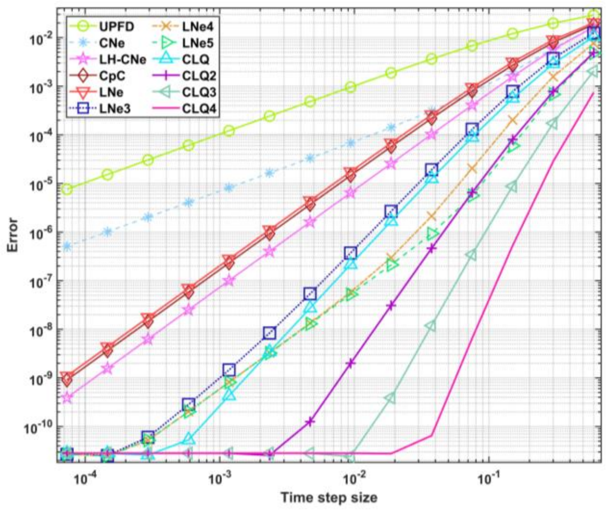

Figure 1. *The error as a function of time step size h for Case study 1*

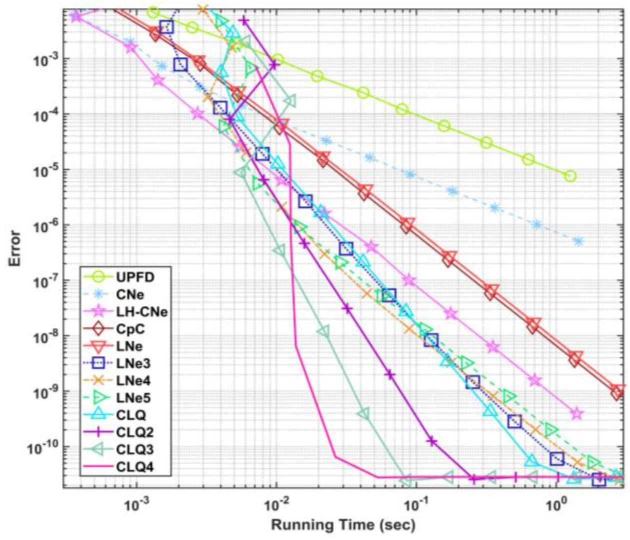

Figure 2. *The error as a function of the running time for Case study 1*

4.2. Case study 3 shortest wavelength initial function

Now $N_x = 21,\ N_y = 20$, and $t_{\text{fin}} = 0.2$. The initial condition is an alternating sequence of zeros and ones, which is a function with the shortest possible wavelength in the mesh:

$$u_i^0 = \text{ceil}\left\{(-1)^i / 2\right\}, \tag{9}$$

where the 'ceil' function returns the smallest integer value which is greater than or equal to the argument. The capacities and the resistances are again set to unity, thus $Sr = 365,\ h_{\text{MAX}}^{\text{EE}} = 0.25$. The results of this numerical experiment are presented in *Figure 3* and *4*, where *Nrun* = 150 has been used, because this system is relatively small.

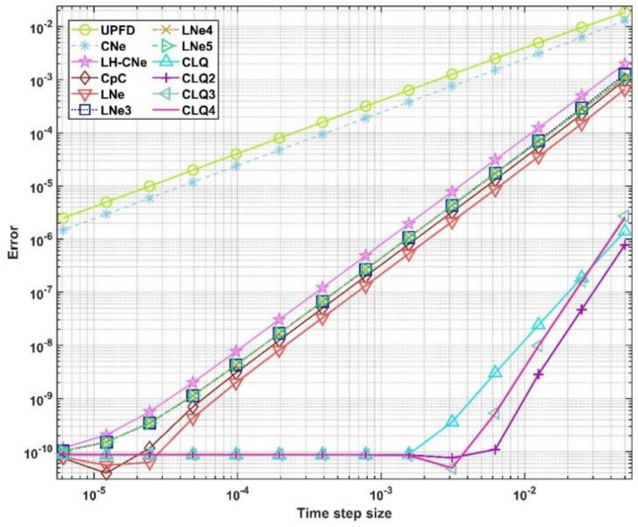

Figure 3. *The error as a function of time step size h for Case study 2*

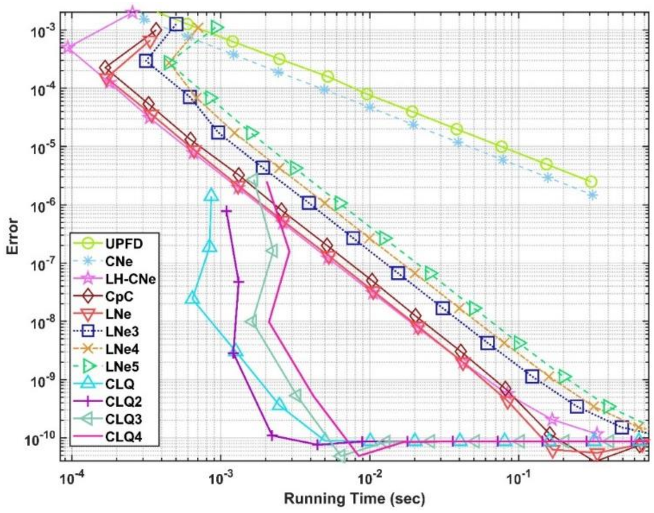

Figure 4. *The error as a function of the running time for Case study 2*

4.3. Case study 3: a very stiff system

Now $N_x = 91$, $N_y = 100$, and $t_{\text{fin}} = 0.2$. The initial condition is $u_i^0 = rand$ to ensure that short and long-wavelengths components are present. The distribution of the capacities and the resistances has a width of six order of magnitude: $a_C = a_{Rx} = a_{Ry} = 3$, $b_C = b_{Rx} = b_{Ry} = 6$. This yields a very large stiffness and low CFL limit: thus $Sr = 1.17 \cdot 10^{11}$, $h_{\text{MAX}}^{\text{EE}} = 1.4 \cdot 10^{-6}$. Since the system is quite large and therefore the fluctuations in the running times are relatively low, only *Nrun*=6 has been used. The results of this numerical experiment are presented in *Figure 5* and *6*.

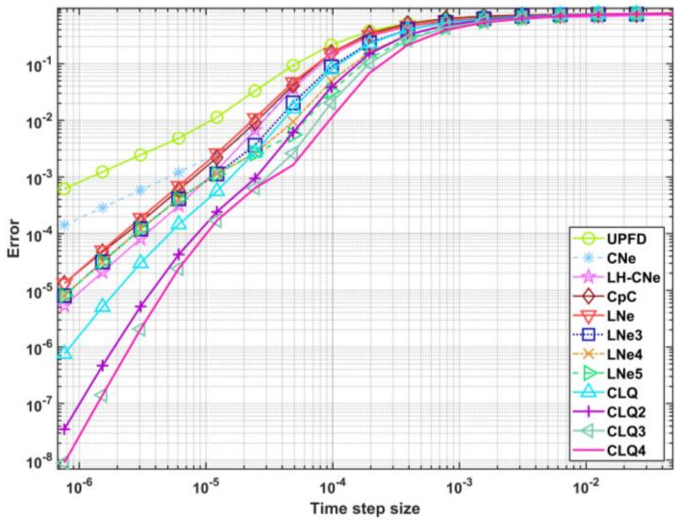

Figure 5. *The error as a function of time step size h for Case study 3*

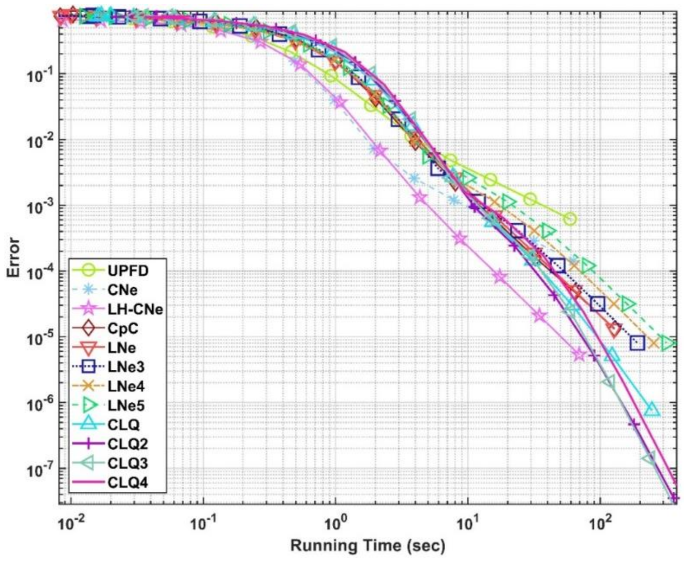

Figure 6. *The error as a function of the running time for Case study 3*

4.4. Case study 4: uniform anisotropy with random initial function

Now $N_x = 41$, $N_y = 41$, and $t_{\text{fin}} = 0.2$. The initial condition is function (9) again. The capacities are again set to unity, but the resistances in the *x* direction are 10 thousand times larger than in the *y* direction: $a_C = b_C = 0$, $a_{Rx} = 2$, $a_{Rz} = -2$, $b_{Rx} = b_{Ry} = 0$, thus $R_x \equiv 100$, $R_z \equiv 0.01$. This yields $Sr = 6.8 \cdot 10^6$ and $h_{\text{MAX}}^{\text{EE}} = 0.005$. The results of this numerical experiment are presented in *Figure 7* and *8*, where *Nrun* = 18 has been used.

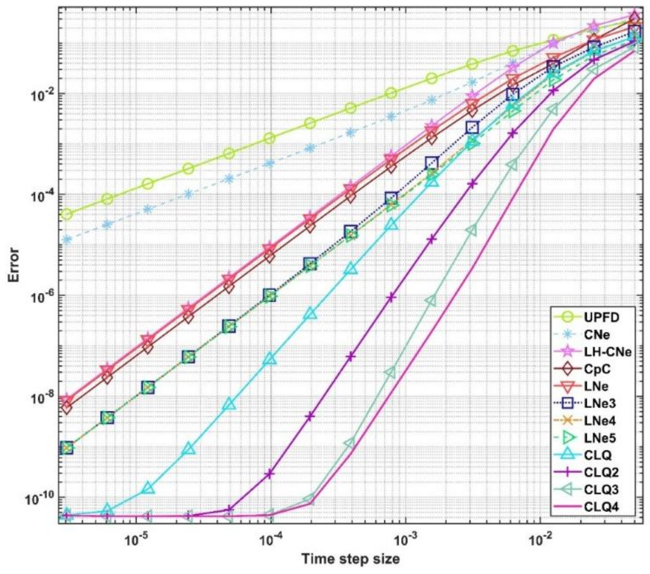

Figure 7. *The error as a function of time step size h for Case study 4*

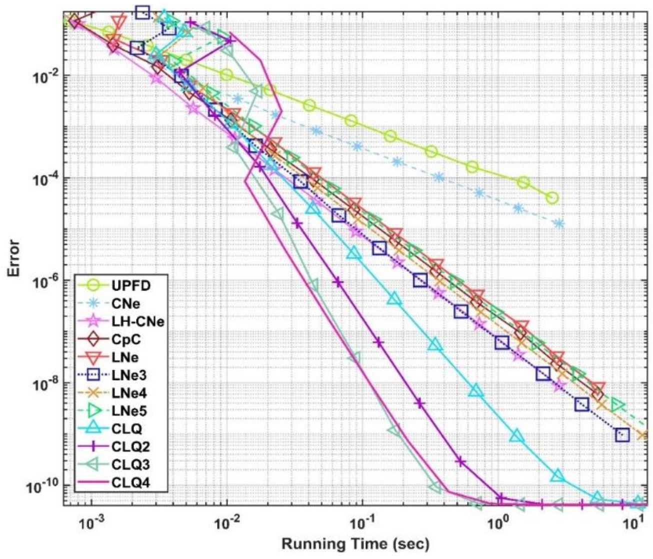

Figure 8. *The error as a function of the running time for Case study 4*

4.5. Case study 5: random anisotropy with smooth initial function

Now $N_x = N_y = 51$, , and $t_{\text{fin}} = 0.2$. The parameters of the system have a quite wide range again: $a_C = 3,\ b_C = 6,\ a_{Rx} = 2,\ b_{Rx} = 2,\ a_{Ry} = 0,\ b_{Ry} = 2$. One can see that the resistances in the *x* direction are two orders of magnitude larger than in the y direction, but now this is a random variable. We obtained that $Sr = 1.11 \cdot 10^{10}$, $h_{\text{MAX}}^{\text{EE}} = 1.84 \cdot 10^{-5}$. The initial condition is the same product of cosine functions as in (8). The results of this numerical experiment are presented in *Figure 9* and *10*, where *Nrun* = 20 has been used.

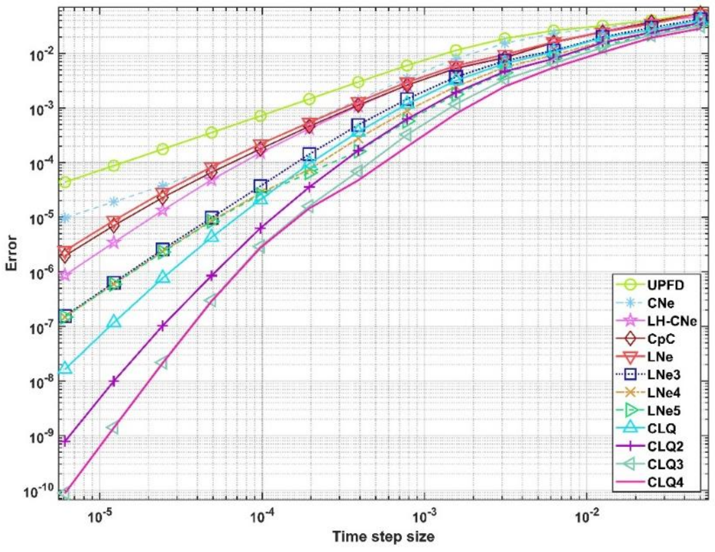

Figure 9. *The error as a function of time step size h for Case study 5*

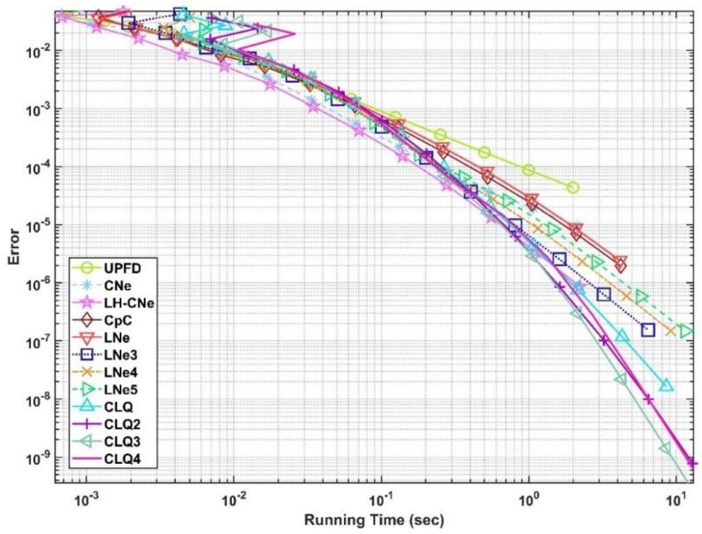

Figure 10. *The error as a function of the running time for Case study 5*

5. Discussion and conclusions

We studied 12 numerical algorithms which obey the Maximum and Minimum principle for arbitrary time step size when they are applied to the heat conduction equation. The numerical case studies presented here confirmed this statement, since the methods behaved well without the slightest sign of instability even for the stiff cases.

If a low or medium accuracy is required, generally the LH-CNe is the most efficient among the methods since it serves quite accurate results in very short time according to the running time measurements. If higher accuracy is required, the higher order CLQ family is the best choice, even if one time step is much slower than in the case of the low order algorithms. If the stiffness is very large, the differences between the methods in accuracy and efficiency is low due to order-reduction. If the initial temperature function is smooth, the multi-stage LNe methods can be as accurate as the higher order CLQ and CLQ2 methods for large time step sizes. On the other hand, when the very short wavelength initial function was used, there was a large gap between the accuracy of the first, the second and the higher order algorithms. In these cases, the LH-CNe scheme can lose its advantage in the low and medium accuracy region.

We note that in the Conclusion section of Part 2, a more detailed comparison of the properties and performance of the methods will be presented based on much more numerical experiments.

References

[1] Agbavon, K. M., and Appadu, A. R. (2020). Construction and analysis of some nonstandard finite difference methods for the FitzHugh–Nagumo equation. *Numer. Methods Partial Differ. Equ.*, 36(5), 1145–1169. **https://doi.org/10.1002/num.22468**

[2] Mátyás, L., and Barna, I. F. (2022). General self-similar solutions of diffusion equation and related constructions. *Romanian J. Phys.*, 67, 101.

[3] Mbroh, N. A., and Munyakazi, J. B. (2021). A robust numerical scheme for singularly perturbed parabolic reaction-diffusion problems via the method of lines. *Int. J. Comput. Math.*, no. May, **https://doi.org/10.1080/00207160.2021.1954621**

[4] Ali, I., Haq, S., Nisar, K. S., and Arifeen, S. U. (2021). Numerical study of 1D and 2D advection-diffusion-reaction equations using Lucas and Fibonacci polynomials. *Arab. J. Math.* **https://doi.org/10.1007/s40065-021-00330-4**

[5] Holmes, M. H. (2007). *Introduction to numerical methods in differential equations*. New York: Springer. **https://doi.org/10.1007/978-0-387-68121-4**

[6] Chen-Charpentier, B. M., and Kojouharov, H. V. (2013). An unconditionally positivity preserving scheme for advection-diffusion reaction equations. *Math. Comput. Model.*, 57(9–10), 2177–2185. **https://doi.org/10.1016/j.mcm.2011.05.005**

[7] Appadu, A. R. (2017). Performance of UPFD scheme under some different regimes of advection, diffusion and reaction. *Int. J. Numer. Methods Heat Fluid Flow*, 27(7), 1412–1429. **https://doi.org/10.1108/HFF-01-2016-0038**

[8] Drljača, B., and Savović, S. (2019). Unconditionally positive finite difference and standard explicit finite difference schemes for power flow equation. *Univ. Thought - Publ. Nat. Sci.*, 9(2), 75–78. **https://doi.org/10.5937/univtho9-23312**

[9] Ndou, N., Dlamini, P., and Jacobs, B. A. (2022). Enhanced unconditionally positive finite difference method for advection–diffusion–reaction equations. *Mathematics*, 10(15), 15. **https://doi.org/10.3390/math10152639**

[10] Omle, I., Askar, A. H., and Kovács, E. (2022). Systematic testing of explicit positivity preserving algorithms for the heat-equation. *J. Math. Comput. Sci.*, 12(0), 162. **https://doi.org/10.28919/10.28919/JMCS/7407**

[11] Kovács, E., Majár, J., and Saleh, M. (2024). Unconditionally positive, explicit, fourth order method for the diffusion- and Nagumo-type diffusion–reaction equations. *J. Sci. Comput.*, 2024 982, 98(2), 1–39. **https://doi.org/10.1007/S10915-023-02426-9**

[12] Nagy, Á., Saleh, M., Omle, I., Kareem, H., and Kovács, E. (2021a). New stable, explicit, shifted-hopscotch algorithms for the heat equation. *Math. Comput. Appl.*, 26(3), 61. **https://doi.org/10.3390/mca26030061**

[13] Kovács, E. (2020a). New stable, explicit, first order method to solve the heat conduction equation. *J. Comput. Appl. Mech.*, 15(1), 3–13. **https://doi.org/10.32973/jcam.2020.001**

[14] Kovács, E. (2020b). A class of new stable, explicit methods to solve the non-stationary heat equation. *Numer. Methods Partial Differ. Equ.*, 37(3), 2469–2489. **https://doi.org/10.1002/num.22730**

[15] Nagy, Á., Omle, I., Kareem, H., Kovács, E., Barna, I. F., and Bognar, G. (2021b). Stable, explicit, leapfrog-hopscotch algorithms for the diffusion equation. *Computation*, 9(8), 92. **https://doi.org/10.3390/computation9080092**

[16] Kovács, E., Nagy, Á., and Saleh, M. (2021). A set of new stable, explicit, second order schemes for the non-stationary heat conduction equation. *Mathematics*, 9(18), 2284. **https://doi.org/10.3390/math9182284**