Multidiszciplináris tudományok, 10. kötet. (2020) 4 sz. pp. 349-360 **https://doi.org/10.35925/j.multi.2020.4.38**

CONSTRUCTION AND INVESTIGATION OF NEW NUMERICAL ALGORITHMS FOR THE HEAT EQUATION

Part 3

Mahmoud Saleh
PhD student, Institute of Physics and Electric Engineering, University of Miskolc
3515 Miskolc, Miskolc-Egyetemváros, e-mail: mhmodsalh84@gmail.com

Ádám Nagy
Teaching Assistant, Institute of Physics and Electric Engineering, University of Miskolc
3515 Miskolc, Miskolc-Egyetemváros, e-mail: fizadam@uni-miskolc.hu

Endre Kovács
Associate Professor, Institute of Physics and Electric Engineering, University of Miskolc
3515 Miskolc, Miskolc-Egyetemváros, e-mail: fizendre@uni-miskolc.hu

Abstract

This paper is the third part of a paper-series in which we create and examine new numerical methods for solving the heat conduction equation. Now we present additional numerical test results of the new algorithms which were constructed using the known, but non-conventional UPFD and odd-even hopscotch methods in Part 1. In Part 2 these methods were tested in one space dimension, while in this part of the series, we present numerical case studies for two and three space dimensions, as well as for inhomogeneous media.

Keywords: *explicit numerical methods, heat equation, parabolic PDEs, hopscotch method, UPFD*

1. Introduction and the generalized form of the studied equation

This paper is the last part of a longer paper-series on our new methods to simulate heat conduction phenomena. In the first part [1] we used the unconditionally positive finite-difference (UPFD) method of Chen-Charpentier et al. [2], and the odd-even hopscotch algorithm of Gordon [3], and Gourlay [4], [5] to construct and then to analytically examine new schemes. In Part 2 we investigated numerically the performance of these methods compared to the original UPFD and hopscotch methods in the simplest, materially homogeneous system, where heat conduction is modelled by the heat equation:

$$\frac{\partial u}{\partial t} = \alpha \Delta u + q \, , \tag{1}$$

where $u = u(x,t)$ is the unknown temperature function, α is the thermal diffusivity, and q is the intensity of external or internal heat sources, respectively.

In this part, we solve the more general form of the heat equation

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

where $u = u(\vec{r}, t)$ depends not only on *x*, but also on the *y* and *z* coordinates, $k = k(\vec{r})$, $c = c(\vec{r})$, $\rho = \rho(\vec{r})$ are the heat conductivity, the specific heat, and the mass-density, respectively. Now $q = q(\vec{r})$ and also k, c, ρ are in principle arbitrary functions of the space variables, except that $k, c, \rho,$ and $\alpha = k/(c\rho)$ are non-negative. In this case the general analytical solution of the problem does not exist. For the discretization of Eq. (2) we first consider a one dimensional, equidistant grid. Using the same central difference formula as in Part 1 we can write

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

At this point we change to cell variables, thus the subscripts refer to whole cells:

$$\frac{du_{\mathrm{i}}}{dt} = \frac{A}{c_{\mathrm{i}}\rho_{\mathrm{i}}A\Delta x}\left(k_{\mathrm{i,i+1}}\frac{u_{\mathrm{i+1}}-u_{\mathrm{i}}}{\Delta x} + k_{\mathrm{i-1,i}}\frac{u_{\mathrm{i\text{-}1}}-u_{\mathrm{i}}}{\Delta x}\right) + Q_{\mathrm{i}}\,,$$

where u_{i} is the temperature of the cell *i*, $C_i = c_i m_i = c_i \rho_i V_i$ is the heat capacity of the cell, Δx is the length, *m* is the mass, while $V_i = A_i \Delta x_i$ is the volume of the cell. We define two other quantities, the heat source term Q_{i} of the cell

$$Q_{\mathrm{i}} = \frac{1}{V_{\mathrm{i}}}\int_{V_i} q\,dV,$$

and the thermal resistance $R_{\mathrm{ij}} = \dfrac{\Delta x}{k_{\mathrm{ij}}A}$ between the cell *i* and its arbitrary neighbour *j*. If the grid is not equidistant, the distances between the cell-centres can be approximated as $d_{\mathrm{ij}} = \left(\Delta x_{\mathrm{i}} + \Delta x_{\mathrm{j}}\right)/2$ and the resistances can be estimated as $R_{\mathrm{ij}} \approx \dfrac{d_{\mathrm{ij}}}{k_{\mathrm{ij}}A_{\mathrm{ij}}}$. Using these quantities we obtain

$$\frac{du_{\mathrm{i}}}{dt} = \frac{u_{\mathrm{i\text{-}1}}-u_{\mathrm{i}}}{R_{\mathrm{i-1,i}}C_{\mathrm{i}}} + \frac{u_{\mathrm{i+1}}-u_{\mathrm{i}}}{R_{\mathrm{i+1,i}}C_{\mathrm{i}}} + Q_{\mathrm{i}}\ .$$

This can be straightforwardly generalised to obtain the ODE system for a general (possibly unstructured) grid in any number of space-dimensions, which gives the time derivative of each cell-variable independently of any coordinate-system:

$$\frac{du_{\mathrm{i}}}{dt} = \sum_{\mathrm{j}\neq\mathrm{i}}\frac{u_{\mathrm{j}}-u_{\mathrm{i}}}{R_{\mathrm{i,j}}C_{\mathrm{i}}} + Q_{\mathrm{i}}\ .$$

Formally we can write this equation system into the same matrix-form as Eq. (8) in Part 1. Now an off-diagonal $m_{\mathrm{ij}} = 1/\left(R_{\mathrm{ij}}C_{\mathrm{i}}\right)$ element of the matrix M can be nonzero only if the cells *i* and *j* are neighbours, i.e. there is direct heat conduction between them. The diagonal elements are the sum of the off-diagonal elements of the same row:

$$m_{\mathrm{ii}} = \sum_{\mathrm{j}\neq\mathrm{i}} \frac{-1}{R_{\mathrm{i,j}}C_{\mathrm{i}}} \doteq \frac{-1}{\tau_{\mathrm{i}}} \ ,$$

where we introduced the characteristic time or time-constant τᵢ of the cell *i*, which, for the simplest one-dimensional case, would take the form

$$\tau_{\mathrm{i}} = \frac{\Delta x^2}{2\alpha} \ \ (1 < i < N), \qquad \tau_1 = \tau_N = \frac{\Delta x^2}{\alpha} \, .$$

2. Description of the used numerical methods

Now we redefine Algorithms 1-6 in the generalized formalism. For the treatment of the heat sources, we follow the version in each case which have been proven to be the most accurate in Part 2. However, for the sake of simplicity, we omit the ‘a’ and ‘b’ letters referring to the position of the source term in the formulas.

Algorithm 1 (UPFD method).

$$u_i^{\mathrm{n+1}} = \frac{u_i^{\mathrm{n}} + h\sum_{\mathrm{j}\neq i} \frac{u_{\mathrm{j}}^{\mathrm{n}}}{C_i R_{\mathrm{ij}}}}{1 + \frac{h}{\tau_{\mathrm{i}}}} + Q_{\mathrm{i}} h \, .$$

Algorithm 2 (successive displacement UPFD method).

$$u_i^{\mathrm{n+1}} = \frac{u_i^{\mathrm{n}} + h\sum_{\mathrm{j}<i} \frac{u_{\mathrm{j}}^{\mathrm{n+1}}}{C_i R_{\mathrm{ij}}} + h\sum_{\mathrm{j}>i} \frac{u_{\mathrm{j}}^{\mathrm{n}}}{C_i R_{\mathrm{ij}}} + Q_{\mathrm{i}} h}{1 + \frac{h}{\tau_{\mathrm{i}}}} \, .$$

Now we try to describe the structure of the odd-even hopscotch-type algorithms when the number of space dimensions is more than one. Let us define a bipartite grid, where the nodes can be divided into two similar groups A and B such that nodes in group A are only nearest neighbours of nodes from group B and vice versa. For example, in a rectangular grid where the space indices are *i* and *j* in 2D and *i*, *j* and *k* in 3D, the cells where *i*+*j* or *i*+*j*+*k* is odd belong to subgrid A, and where these values are even belong to subgrid B. For simplicity we will use one letter *i* only instead of the index pairs or triples. At the first stage the new values of *u* are calculated only at the cells of subgrid A. During this stage only the values at the beginning of the time step are used. At the second stage, the node-values of B are calculated, using the values at subgrid A at the end of the time step which are already calculated at stage one. At the next time step the roles of subgrid A and B are interchanged. It is easy to see that if our grid is bipartite, the second stage is not implicit in reality (even if an implicit formula is used) because – similarly to Algorithm 2 – the necessary values $u_{i-1}^{\mathrm{n+1}}$ and $u_{i+1}^{\mathrm{n+1}}$ are already calculated at stage one and now only one unknown, namely $u_i^{\mathrm{n+1}}$ remained. Otherwise (e.g. for irregular grids) some implicitness cannot be avoided and the main advantage of this structure disappears.

Algorithm 3 (original odd-even Hopscotch).

Stage 1. Take a time step with the explicit Euler method for the cells of A:

$$u_i^{\mathrm{n+1}} = \left(1 - \frac{h}{\tau_i}\right) u_i^{\mathrm{n}} + h \sum_{\mathrm{j \in B}} \frac{u_j^{\mathrm{n}}}{R_{\mathrm{i,j}} C_{\mathrm{i}}} + hQ_{\mathrm{i}} .$$

Stage 2. Take a time step with the implicit Euler method for the remaining B cells:

$$u_i^{\mathrm{n+1}} = \frac{u_i^{\mathrm{n}} + h \sum_{\mathrm{j \in A}} \frac{u_{\mathrm{j}}^{\mathrm{n+1}}}{C_i R_{\mathrm{ij}}} + Q_{\mathrm{i}} h}{1 + \frac{h}{\tau_{\mathrm{i}}}} .$$

We note that when the summations are calculated, if cell *j* is not the neighbour of cell *i*, then R_{ij} will be automatically infinity, thus the summation must be carried out only over the neighbours. The second version of the odd-even hopscotch methods can be the combination of two explicit steps.

Algorithm 4 (explicit+explicit odd-even Hopscotch).

Stage 1. Take a time step with the explicit Euler method for the cells of A:

$$u_i^{\mathrm{n+1}} = \left(1 - \frac{h}{\tau_i}\right) u_i^{\mathrm{n}} + h \sum_{\mathrm{j \in B}} \frac{u_j^{\mathrm{n}}}{R_{\mathrm{i,j}} C_{\mathrm{i}}} + hQ_{\mathrm{i}} .$$

Stage 2. Take a time step with the explicit Euler method again for the remaining B cells:

$$u_i^{\mathrm{n+1}} = \left(1 - \frac{h}{\tau_i}\right) u_i^{\mathrm{n}} + h \sum_{\mathrm{j \in} A} \frac{u_j^{\mathrm{n+1}}}{R_{\mathrm{i,j}} C_{\mathrm{i}}} + hQ_{\mathrm{i}} .$$

Algorithm 5 (UPFD+Explicit Euler odd-even Hopscotch).

Stage 1. Take a time step with the UPDF method for the cells of A:

$$u_i^{\mathrm{n+1}} = \frac{u_i^{\mathrm{n}} + h \sum_{\mathrm{j \in B}} \frac{u_j^{\mathrm{n}}}{R_{\mathrm{i,j}} C_{\mathrm{i}}} + Q_{\mathrm{i}} h}{1 + \frac{h}{\tau_{\mathrm{i}}}} .$$

Stage 2. Take a time step with the explicit Euler method for the remaining B cells:

$$u_i^{\mathrm{n+1}} = \left(1 - \frac{h}{\tau_i}\right) u_i^{\mathrm{n}} + h \sum_{\mathrm{j \in A}} \frac{u_j^{\mathrm{n+1}}}{R_{\mathrm{i,j}} C_{\mathrm{i}}} + hQ_{\mathrm{i}} .$$

Algorithm 6 (UPFD+UPFD odd-even Hopscotch).

Stage 1. Same as Stage 1 in Algorithm 5.

Stage 2. Same as Stage 2 in Algorithm 3.

We remind the reader that when the calculations at the second stage start, the new values $u_{i-1}^{\mathrm{n+1}}$ and $u_{i+1}^{\mathrm{n+1}}$ of the neighbours are already calculated at the first stage, therefore the UPFD formula coincides with the implicit Euler scheme in the second stage.

In the remaining sections of this paper, we numerically investigate the most important properties of these methods.

3. Numerical experiments

We perform five numerical case studies to test the above defined numerical methods. We examine the error of the numerical calculations as a function of the time step size *h*. The solutions are examined and compared at final time $t_{\mathrm{fin}} = 0.1$. We define the (global) 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 methods at t_{fin}, the end of the examined time interval:

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

The reference solution is either the analytical solution of the PDE or a numerical solution obtained by applying a very accurate time integrator to the spatially discretized PDE, which will be specified later.

3.1. Case study 1

We solve PDE (1) with $\alpha = 1$, $k = 1, c = 1, \rho = 1$ and $q \equiv 0$ on the unit square $(x, y) \in [0,1] \times [0,1]$. The initial condition is the product of two sine functions with different wavelengths:

$$u(x, y, t = 0) = \sin(k_x \pi x) \sin(k_y \pi y),$$

where the wave numbers are fixed to $k_x = 1,\ k_y = 2$. The simplest zero Dirichlet boundary conditions are used

$$u(x = 0, y, t) = u(x = 1, y, t) = u(x, y = 0, t) = u(x, y = 1, t) = 0.$$

It is easy to check that the analytical solution of this problem is

$$u(x, t) = \sin(k_x \pi x) \sin(k_y \pi y)\, \mathrm{e}^{-\left(k_x^2 + k_y^2\right)\pi^2 t}.$$

We apply an equidistant grid to discretize the space variables. The cell number along axis *x* is set to $N_x = 50$. Similarly, the cell number along axis *y* is $N_y = 50$. Thus, we have a grid with total cell number $N = N_x \times N_y = 50 \times 50 = 2500$. We can calculate the cell lengths $(\Delta x, \Delta y)$ in both dimensions and using these values, the cell capacities (C) and the thermal resistances (R_x, R_y) can be given in the following way:

$$C = \Delta x \cdot \Delta y, \quad R_x = \frac{\Delta x}{\Delta y}, \quad R_y = \frac{\Delta y}{\Delta x}.$$

The results of this numerical experiment are presented in Fig 1 in a log-log diagram.

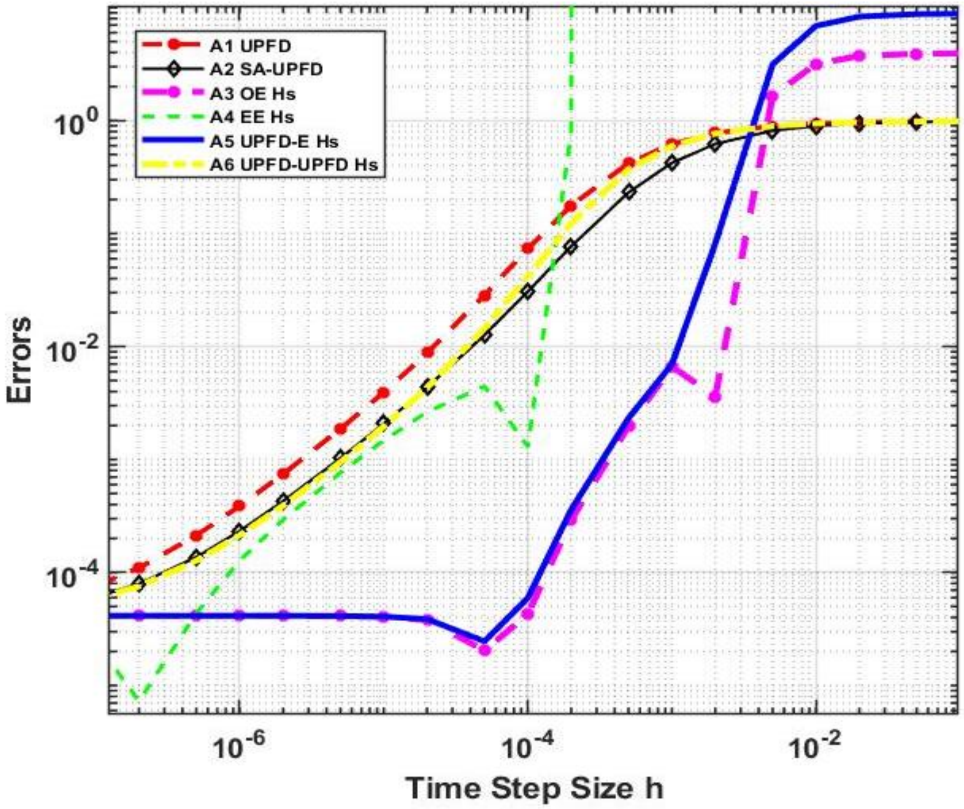

Figure 1. *The error as a function of time step size h for different Algorithms (A1-A6). A1 means Algorithm 1 (UPFD method), etc.*

3.2. Case study 2

We use the same grid as in the previous case study, but now the initial condition is a quadratic function:

$$u(x,y,t=0)=x^2+y^2/2\,.$$

Now it is easy to find the analytical solution

$$u(x,y,t)=x^2+y^2/2+3t$$

if we suppose the following time-dependent Dirichlet boundary conditions

$$u(x=0,y,t)=y^2/2+3t\,,\quad u(x=1,y,t)=1+y^2/2+3t$$
$$u(x,y=0,t)=x^2+3t\,,\quad u(x,y=1,t)=x^2+1/2+3t\,,$$

while the external heat source term is uniformly zero. The obtained error-functions are presented in Fig 2. As we mentioned in Part 2, Algorithm 3 gives the exact values apart from numerical round-off errors, thus the error function of this algorithm is not displayed.

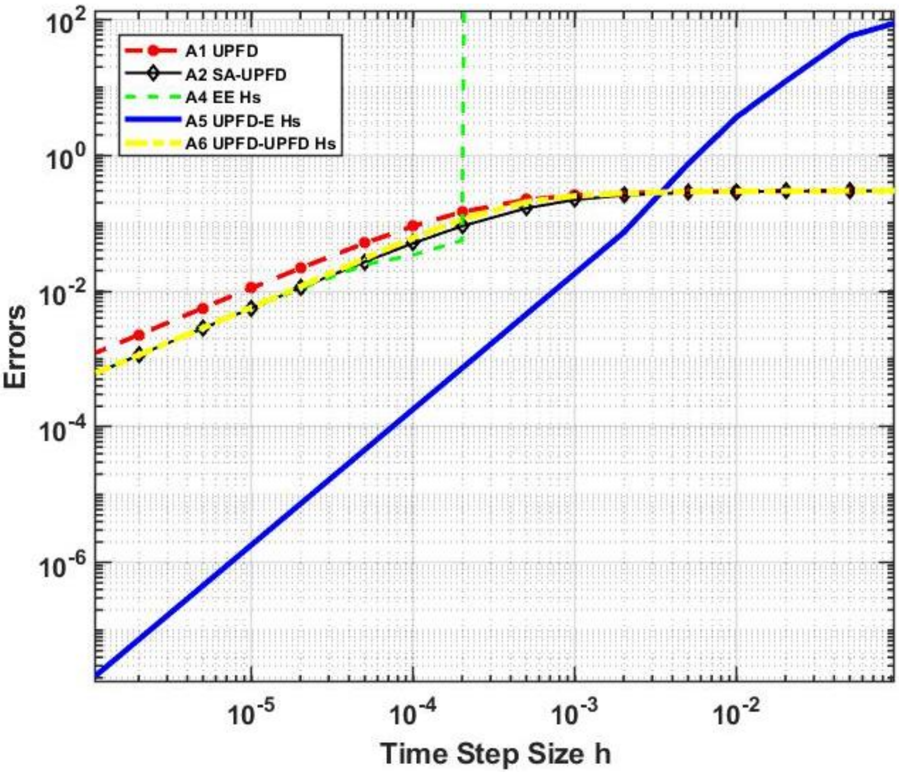

Figure 2. *The error as a function of time step size h for five different Algorithms. A1 means Algorithm 1 (UPFD method), etc.*

3.3. Case study 3

In this section a three dimensional rectangle-shaped system is examined with $N = N_x \times N_y \times N_z$ cells, where the edge of the system is thermally isolated regarding conductive type heat transfer, which means closed (zero Neumann) boundary conditions. We emphasize that the size and the shape of the cells are not necessarily identical. We set $N_x = 20,\ \ N_y = 25,\ \ N_z = 20$ thus the total number of cells is $N = 10000$. Different random values are given to the heat capacities and to the thermal resistances:

$$C_{\mathrm{i}} = 10^{(-1+2\cdot rand)},\ R_{\mathrm{x,i}} = 10^{(-1+2\cdot rand)}, R_{\mathrm{y,i}} = 10^{(-1+2\cdot rand)},\ R_{\mathrm{z,i}} = 10^{(-1+2 rand)}$$

where *rand* is a random number generated by the MATLAB uniformly in the (0,1) interval for each quantity. It means that the capacities (the resistances) follow a log-uniform distribution between 0.1 and 10. The initial temperatures have a uniform distribution between 0 and 1, i.e. $u_{\mathrm{i}}(0) = rand$, while the source-terms are zero everywhere. As the system is thermally isolated, the system matrix has a zero eigenvalue, all other eigenvalues are negative. If we denote the eigenvalues with the smallest (largest) absolute value with $\lambda_{MIN}(\lambda_{MAX})$, then the stiffness ratio can be calculated as

$$Sr = \frac{\lambda_{\mathrm{MAX}}}{\lambda_{\mathrm{MIN}}} = 1.4 \times 10^{5} \cdot$$

For the explicit Euler method, the maximum possible time step size is

$$h_{\mathrm{MAX}}^{\mathrm{EE}} = \left| \frac{2}{\lambda_{\mathrm{MAX}}} \right| = 0.016 \,.$$

Above this threshold instabilities necessarily appear for the Explicit Euler method.

To obtain a reference solution, we use the implicit ode15s solver of MATLAB, which is a variable-step, variable-order solver based on the numerical differentiation formulas (NDFs) of orders 1 to 5, where the letter s indicates that the codes were suggested to use in case of stiff systems. During this calculation, we apply stringent error tolerance (‘RelTol’ and ‘AbsTol’ were both 10^{-8}). The obtained errors as a function of the time step size are presented in Fig 3.

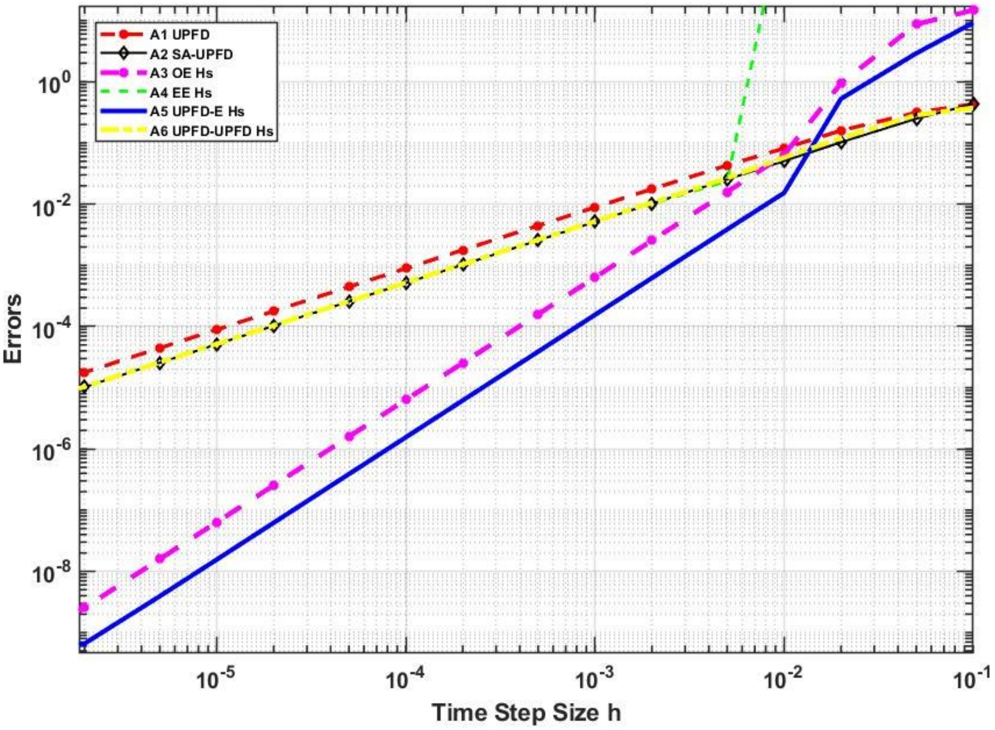

Figure 3. *The error as a function of time step size h for different Algorithms (A1-A6). A1 means Algorithm 1 (UPFD method), etc.*

3.4. Case study 4

Here everything is the same as in the previous case study, except the source-terms, which now have a uniform distribution between -5 and 5, $Q_i = -5 + 10 \cdot rand$ and, of course, new random values are generated to the appropriate quantities with the same distribution. Now the stiffness ratio is $Sr = 1.1 \times 10^5$ and $h_{\mathrm{MAX}}^{\mathrm{EE}} = 0.021$. The results can be seen in Fig 4.

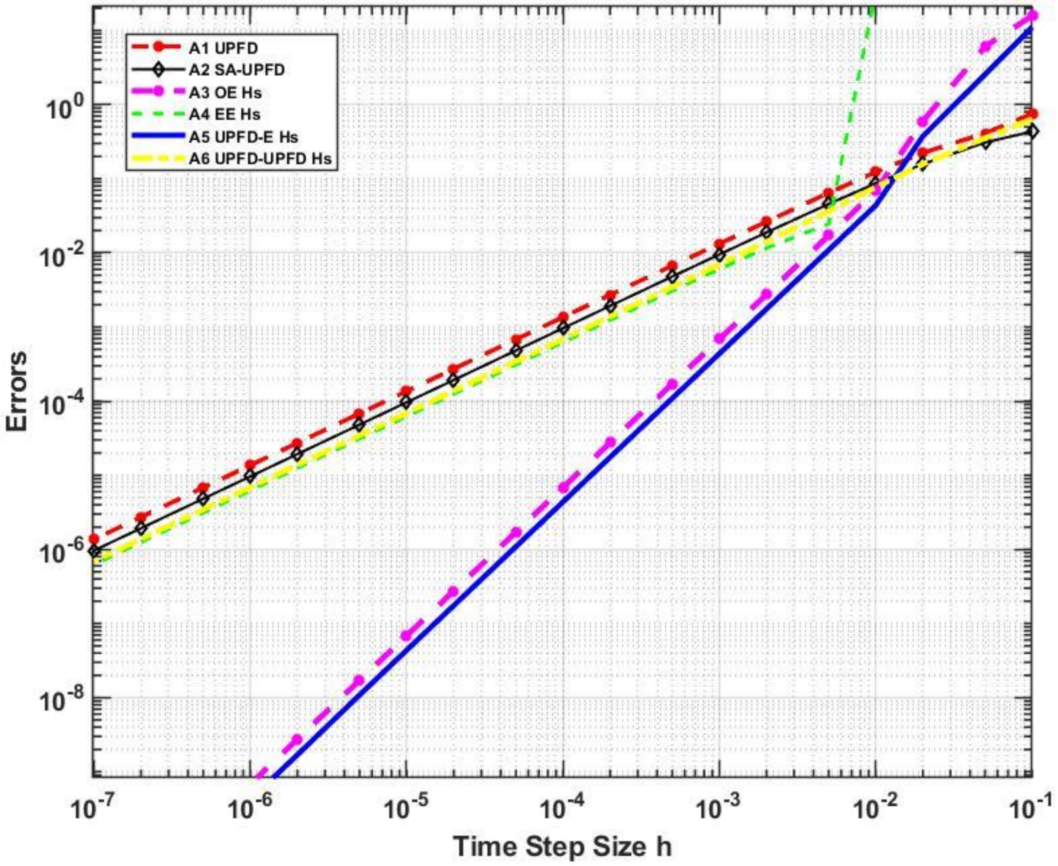

Figure 4. *The error as a function of time step size h for different Algorithms (A1-A6). A1 means Algorithm 1 (UPFD method), etc.*

3.5. Case study 5

In this case everything is the same as in the previous case study, except the distribution of the random numbers given to the heat capacities and to the thermal resistances is different:

$$C_{\mathrm{i}} = 10^{(-3+6\cdot rand)},\ R_{\mathrm{x,i}} = 10^{(-3+6\cdot rand)}, R_{\mathrm{y,i}} = 10^{(-3+6\cdot rand)},\ R_{\mathrm{z,i}} = 10^{(-3+6\cdot rand)}\ .$$

Now the stiffness ratio is much higher, $Sr = 4.21\times10^{10}$ and $h_{\mathrm{MAX}}^{\mathrm{EE}} = 1.26\times10^{-6}$. In Fig. 5 we again present the errors as a function of time step size *h*. One can see that much smaller time step sizes are necessary to obtain acceptable solutions.

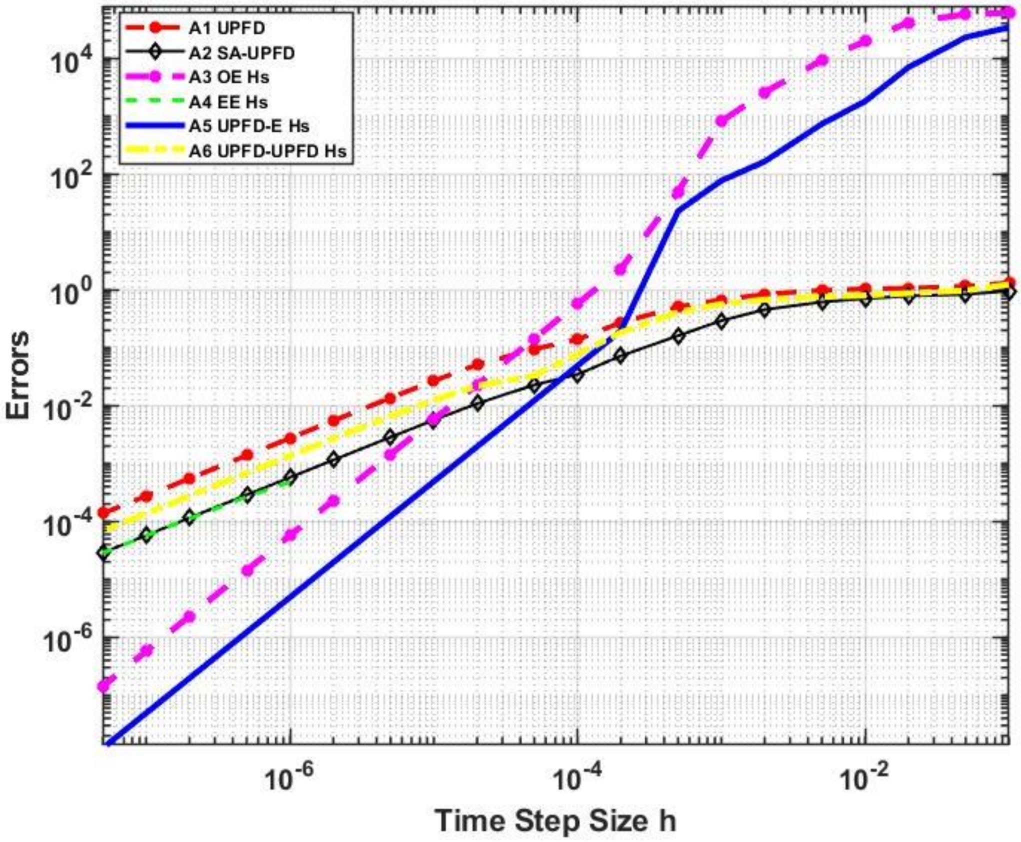

Figure 5. *The error as a function of time step size h for different Algorithms (A1-A6). A1 is for Algorithm 1 (UPFD method), etc.*

4. Summary and conclusions

In this paper-series, we used two known, but non-conventional algorithms, namely the UPFD and the odd-even hopscotch methods to construct new numerical algorithms for the solution of the heat equation. In Part 1 we defined and analysed the old and new methods in terms of convergence, stability and positivity in one space dimensional homogeneous media without the heat source term. We proved that all of the schemes are (at least) first order time integrators for the spatially discretized heat equation. Then we applied von Neumann stability analysis, which showed that Algorithms 1, 2 and 6 are unconditionally stable. This statement was also reinforced by proving that these algorithms guarantee the fulfilment of the Maximum and Minimum principles. The stability analysis was performed only for one time step of Algorithms 3, 4 and 5 and it indicated that these methods can be unstable for large time step sizes.

In Part 2 five numerical case studies were presented and different posibilities for handling the *q* source term were examined. Finally in Part 3 we extended the methods for 2 and 3 space dimensions and for systems where the material properties and the mesh spacings are not uniform. All numerical experiments demonstrate that the algorithms are unconditionally stable except for Algorithm 4, thus rigorous stability analysis should be carried out for two time steps for odd-even hopscotch type methods, as the roles of the odd and even nodes interchanges at each time step. Unfortunately performing this by the von Neumann method would be an order of magnitude more difficult and lengthy, therefore it is out of the scope of this paper. Nevertheless the obtained results are still useful, because they explain the inaccuracy of Algorithms 3 and 5 for large time step size, which are observed in the numerical experiments. A similar problem emerged with the order of convergence: we saw that these two algorithms are in fact second order, but this could be proved only by calculations incorporating two time steps.

The numerical case studies reinforced the analytical considerations that the algorithms which contain an explicit Euler formula do not preserve positivity of the solution, but all the others do. The Gauss-Seidel type A2 methods, based on the successive displacement of the data, as well as the hopscotch methods, have the advantage that in a computer implementation it is no longer necessary to allocate two arrays for $\vec{u}^{\,n}$ and $\vec{u}^{\,n+1}$. Instead, it is enough to use just a single array for $\vec{u}^{\,n}$, and perform all the updates on this [6]. However, Algorithm 2 has the disadvantage that, unlike the original and the hopscotch method, the computations cannot be straightforwardly parallelized. Furthermore, the calculated values depend on the numbering of the nodes. On the other hand, the odd-even hopscotch algorithms need a bipartite grid to be explicit, otherwise some degree of implicitness remains and the main advantage of this structure disappears. We summarize the properties of the algorithms in Table 1.

Table 1. *Properties of the algorithms*

	A1 UPFD	A2 SD UPFD	A3 OE Hs	A4 EE Hs	A5 UPDF- E Hs	A6 UPFD-UPFD Hs
analytical stability	✓	✓	✗	✗	✗	✓
numerical stability	✓	✓	✓	✗	✓	✓
numerical second order	✗	✗	✓	✗	✓	✗
positivity preserving	✓	✓	✗	✗	✗	✓
parallelizable	✓	✗	✓	✓	✓	✓
minimal memory	✗	✓	✓	✓	✓	✓
arbitrary grid	✓	✓	✗	✗	✗	✗

Finally, we can conclude that the original UPFD method performs the worst. This does not contradict to our expectations as it has been developed for the advection-diffusion-reaction equation, and it has been shown by Appadu that for diffusion-dominated cases this scheme does not perform very well [7]. In the case of the heat equation, we can advocate for the A2 (successive displacement UPFD) algorithm only if there is no possibility to parallelize or vectorize the code and the grid is irregular. If the grid can be divided into two appropriate subgrids, the hopscotch algorithms are always better, except for the A4 explicit-explicit one, which we wouldn't like to promote, because it has no advantage compared to the simplest Explicit Euler method or the other algorithms studied here. If the unconditional positivity is important, we think that Algorithm 6 is the best choice, otherwise the original odd-even hopscotch is the most advantageous for equidistant grids, and the newly discovered Algorithm 5 (UPFD+Explicit Euler odd-even hopscotch) for non-equidistant grids or inhomogeneous media.

References

[1] Saleh, M., Nagy, Á., Kovács, E.: *Construction and investigation of new numerical algorithms for the heat equation*, Multidisciplinary Sciences 2020, 10(4):323-338. **https://doi.org/10.35925/j.multi.2020.4.37**

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

[3] Gordon, P.: *Nonsymmetric difference equations*, J. Soc. Ind. Appl. Math., vol. 13, no. 3, (1965) pp. 667–673. **https://doi.org/10.1137/0113044**

[4] Gourlay, A. R.: *Hopscotch: a fast second-order partial differential equation solver*, IMA J. Appl. Math., vol. 6, no. 4, (1970) pp. 375–390. **https://doi.org/10.1093/imamat/6.4.375**

[5] Gourlay, A. R.: *General hopscotch algorithm for the numerical solution of partial differential equations*, IMA J. Appl. Math., vol. 7, no. 2, (1971) pp. 216–227. **https://doi.org/10.1093/imamat/7.2.216**

[6] Rycroft, C. H.: *Iterative methods for linear systems An example : a two dimensional Poisson problem*, Lect. Notes, (2009) pp. 1–20.

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