Multidiszciplináris tudományok, 10. kötet. (2020) 4 sz. pp. 323-338 **https://doi.org/10.35925/j.multi.2020.4.36**

CONSTRUCTION AND INVESTIGATION OF NEW NUMERICAL ALGORITHMS FOR THE HEAT EQUATION

Part 1

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

In this paper-series, we use two known, but non-conventional algorithms, the UPFD and the odd-even hopscotch method, to construct new schemes for the numerical solution of the heat equation. In this part of the series, we examine the algorithms analytically. We exactly prove that all the methods are first order time integrators, three of them preserve positivity of the solutions and we deduce important information about the convergence and accuracy of the methods. Numerical case studies will be presented in the next two part of the series.

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

1. Introduction and the studied problem

Heat transfer analysis has a fundamental importance in the design of heat transfer equipment such as boilers, condensers, radiators, heaters, furnaces, refrigerators, heat exchangers and solar collectors. Heat transfer in these devices can be studied and predicted either by direct experimental measurements or by modelling, which usually means solving the mathematical equations which describe the system. The experimental way has the advantage that we examine the actual physical system, and the required physical quantities are determined by measurement, within the limits of experimental error. On the one hand, experiments are usually expensive, time-consuming, and sometimes even impractical. On the other hand, the analytical solution of mathematical equations – especially partial differential equations (PDEs) – which describes realistic systems, is possible only in very limited cases. However, more and more complicated engineering problems can be solved with numerical simulation at a very little cost very quickly, therefore the interest towards this approach is continuously rising [1], [2].

It is well known that heat conduction and many other diffusion-like phenomena are modelled by the following partial differential equation (PDE), the so-called heat equation:

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

where $u = u(x,t)$ is the temperature, α is the thermal diffusivity, and q is the intensity of heat sources (due to e.g. infrared radiation coming from outside of the system) respectively. Eq. (1) is the prototype of second-order linear parabolic PDEs, of which the general analytic solution is well-known. 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

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

where $u = u(\vec{r},t)$ is the unknown function, $k = k(\vec{r},t)$, $c = c(\vec{r},t)$, $\rho = \rho(\vec{r},t)$ are the heat conductivity, the specific heat, and the mass-density, respectively. We will use the fact that $\alpha = k/(c\rho)$ is a non-negative function. Apart from non-negativity, the $\alpha, k, c, \rho,$ and q functions are arbitrary in principle and there is no hope to give analytical solutions in general. We note that many intensively studied non-linear equations like the Fisher, the Kardar-Parisi-Zhang and the FitzHugh-Nagumo [3] equations contain a diffusion-type term as well. We also would like to dispel the possible misconception that simple equations are always solved only analytically and the more difficult ones are only numerically. In fact even the numerical solution of Eq. (1) is still investigated [4], [5], and, on the other hand, even the nonlinear PDEs have some analytical solutions [6]–[10]. However, these latter ones are always valid in special circumstances only, e.g. for specific initial conditions and homogeneous material properties of the media, thus numerical solutions remain necessary and widespread.

Although there are plenty of numerical methods to solve Eq. (1) or (2), for large systems the numerical solution still poses a nontrivial problem. The finite difference methods are usually classified as explicit or implicit methods, and certainly both categories have advantages and disadvantages. Explicit methods are easier to code and parallelize and much faster for the same time step size than implicit methods, but usually only conditionally stable and therefore they are considered as less reliable. In case of implicit methods, the solution of a system of algebraic equations is required at each time step, which can be extremely time consuming for huge matrices. Because of these reasons, we are interested in those explicit methods which are unconditionally stable or at least have improved stability properties. We are convinced that these methods deserve more attention than they currently receive. One of the most well-known example for these methods are the odd-even hopscotch algorithm, which was published first by Gordon [11], then reformulated and analysed by Gourlay [12], [13]. The other example we deal with is the unconditionally positive finite-difference (UPDF) method of Chen-Charpentier et al. [14], developed for the advection-diffusion-reaction equation. In Part 1 we explain for $q \equiv 0$ how it is possible to create new combinations using these two methods and analyse the properties of the old and new methods such as convergence, stability and positivity. In Part 2 several numerical case studies will be presented for one dimensional systems and different posibilities for handling the q source term will be examined. Finally in Part 3 we extend the methods for 2 and 3 space dimensions and for systems where the material properties and the mesh spacings are not uniform.

2. Description of the used numerical methods

Throughout this paper, we take the spatial grid x_i, $i = 1,...,N$ fixed, where N denotes the number of grid points. We use the central difference formula

$$\frac{\partial^2}{\partial x^2} f(x_i, t_j) \approx \frac{\dfrac{f(x_{i+1}, t_j) - f(x_i, t_j)}{\Delta x} + \dfrac{f(x_{i-1}, t_j) - f(x_i, t_j)}{\Delta x}}{\Delta x}$$

for the discretization of the spatial derivative and the simplest discretization of the time derivative, by which we obtain:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2}, \tag{3}$$

where the superscript *n* refers to the time level. Now if one consider the *u* values in the right hand side of (3) at the *n*-th time level, one obtains the explicit Euler (or forward time central space, FTCS) scheme, which can be written as

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

where $r = \dfrac{2\alpha h}{\Delta x^2} > 0$ and $h = \Delta t$. If, on the other hand these *u* values are taken at the $(n+1)$ -th time level, the implicit Euler scheme is obtained:

$$(1+r)u_i^{n+1} = u_i^n + \frac{r}{2}\left(u_{i-1}^{n+1} + u_{i+1}^{n+1}\right). \tag{5}$$

The intermediate option, proposed in [14], is when only the value of the actual node is taken at the $(n+1)$ -th time level

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

which can be rearranged into the following explicit form:

Algorithm 1 (UPFD method).

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

When the program goes through the mesh points one by one, it is also possible to use the already obtained u_{i-1}^{n+1} when the new value u_i^{n+1} at the next node is calculated. With this modification we obtain a one-stage "pseudo-implicit" method:

Algorithm 2 (successive displacement UPFD method).

$$u_i^{n+1} = \frac{u_i^n + \dfrac{r}{2}\left(u_{i-1}^{n+1} + u_{i+1}^n\right)}{1+r}. \tag{7}$$

This successive displacement UPFD method relates to the original one as the Jacobi method for systems of linear algebraic equations to the Gauss–Seidel method [5, p. 306] and also similar to the

successive over-relaxation (SOR) method. The Gauss-Seidel type methods, based on the successive displacement of the data, has 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 [16]. However, it has the disadvantage that, unlike in the original method, the computations cannot be straightforwardly parallelized. Furthermore, the calculated values depend on the numbering of the nodes. This type of logic was applied in the numerical solution of the heat equation by Ghaffar et al [17], but with different concrete formulas, which were derived by direct copying of the original (linear algebra) Gauss-Seidel and SOR algorithms.

For understanding the structure of the odd-even hopscotch algorithm, 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, like in a checkerboard. At the first stage the new values of *u* are calculated only at the point of subgrid A, where the space index of the nodes are odd. During this stage only the values at the beginning of the time step are used. At the second stage, the remaining node-values (where the space indices are even) 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. This logic is visualized in Fig 1.

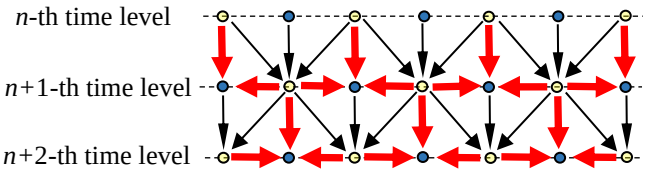

Figure 1. *The stencil of the hopscotch algorithm.*
Thin black arrows (thick red arrows) indicate operations at Stage 1 (Stage 2).

Now for each stage, we can use three formulas, namely the explicit, the implicit and the UPFD, thus altogether we have $3^2=9$ possibilities. As we want to construct explicit methods, we exclude the implicit formula at the first stage, thus 6 possibilities remain. However, at the second stage, the new values u_{i-1}^{n+1} and u_{i+1}^{n+1} of the neighbours are already calculated at the first stage, therefore the UPFD scheme (6) coincides with the implicit scheme (5). We remind that if the old values u_{i-1}^{n} and u_{i+1}^{n} were used, the original UPFD method would be obtained. So finally, we have four different possibilities and we are going to examine all of them. Let us start with the original, well known one.

Algorithm 3 (original odd-even hopscotch).

Stage 1. Take a time step with the explicit Euler method for every other nodes, e.g. where $n+i$ is odd:

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

Stage 2. Take a time step with the implicit Euler method for the remaining nodes, e.g. where $n+i$ is even:

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

As it was already referred to, this second stage is not implicit in reality, because – similarly to Algorithm 2 – the necessary values u_{i-1}^{n+1} and u_{i+1}^{n+1} are already calculated at stage one and now only one unknown, namely u_i^{n+1} remained.

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 every other nodes, e.g. where $n+i$ is odd:

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

Stage 2. Take a time step with the explicit Euler method again for the remaining nodes, e.g. where $n+i$ is even:

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

Now we combine the UPFD method with the explicit Euler formula in the odd-even hopscotch structure.

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

Stage 1. Take a time step with the UPDF method for every other nodes, e.g. where $n+i$ is odd:

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

Stage 2. Take a time step with the explicit Euler method for the remaining nodes, e.g. where $n+i$ is even:

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

Finally we apply the UPFD formula at both stage:

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

Stage 1. Take a time step with the UPDF method for every other nodes, e.g. where $n+i$ is odd:

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

Stage 2. Take a time step with the implicit Euler (which, in this case, is the same as the UPDF) method for the remaining nodes, e.g. where $n+i$ is even:

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

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

3. Convergence of the methods

In this section, we examine the convergence properties of the methods as time-integrators for the spatially discretized systems.

Theorem 1. The Algorithms 1-6 are at least first order numerical methods for the

$$\frac{d\vec{u}}{dt} = M\vec{u}\,, \quad \vec{u}(t=0) = \vec{u}^0 \tag{8}$$

linear ODE initial value problem, where the matrix M is obtained by discretizing the Laplace operator Δ in (1) by the central difference formula as in (3), while $\vec{u}^0$ is an arbitrary vector and $\alpha \in \mathbb{R}$ is an arbitrary nonzero scalar.

Proof. As we consider a one-dimensional system, M is a tridiagonal matrix with the following elements:

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

The exact solution of the initial value problem (8) can be written as follows:

$$\vec{u}(t) = \mathrm{e}^{Mt}\vec{u}^0 = \left(1 + Mt + M^2\frac{t^2}{2} + M^3\frac{t^3}{3!} + \ldots\right)\vec{u}^0 . \tag{9}$$

The zeroth and first order terms in the exact solution at $t=h$ are the following:

$$u_i^{\mathrm{n}+1} = u_i^{\mathrm{n}}\left(1-r\right) + \frac{r}{2}\left(u_{i-1}^{\mathrm{n}} + u_{i+1}^{\mathrm{n}}\right). \tag{10}$$

We will deduce that the expressions for the variable u obtained by any of the presented algorithms coincides with (10) up to first order in r.

Remark 1. As M is tridiagonal, M^2 is pentadiagonal in the bracket in the right hand side of (9). Hence if we examine the exact solution only up to first or second order, the value u_{i}^{n} depends on the first neighbours $u_{i-1}^{\mathrm{n}}, u_{i+1}^{\mathrm{n}}$ or the second neighbours $u_{i-2}^{\mathrm{n}}, u_{i+2}^{\mathrm{n}}$, respectively. So if a formula does not contain the values of the first or second neighbour, then it cannot represent a first or second order method.

Remark 2. The original explicit and the implicit Euler method are well-known first order methods, thus we don't need to examine their convergence-properties.

Proof for Algorithm 1 (UPFD method). The power series of the denominator is $\left(1+x\right)^{-1} = 1 - x + x^2 - \ldots$, thus one can write

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

Proof for Algorithm 2 (successive displacement UPFD method). Applying (7) twice we obtain

$$u_i^{n+1} = \frac{u_i^n + \frac{r}{2}\left(u_{i-1}^{n+1} + u_{i+1}^n\right)}{1+r} = \frac{u_i^n + \frac{r}{2}\left(\frac{u_{i-1}^n + \frac{r}{2}\left(u_{i-2}^n + u_i^n\right)}{1+r} + u_{i+1}^n\right)}{1+r}, \tag{12}$$

thus

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

which is indeed the same as (10) up to first order of *r*.

Remark 3. All the remaining algorithms follow the odd-even Hopscotch structure. At Stage 1 the u_i^{n+1} values are calculated by the explicit Euler method or by the UPFD method. Because of Remark 2 and the calculation in (11) we don't need to examine the first stage, only the second one.

Proof for Algorithm 3 (original odd-even hopscotch). The value of the neighbours of node *i* at the first stage has the values:

$$u_{i-1}^{n+1} = u_{i-1}^n + r\left(\frac{u_{i-2}^n + u_i^n}{2} - u_{i-1}^n\right) \quad \text{and} \quad u_{i+1}^{n+1} = u_{i+1}^n + r\left(\frac{u_{i+2}^n + u_i^n}{2} - u_{i+1}^n\right), \tag{13}$$

thus, at the second stage, one can write

$$\begin{aligned} u_i^{n+1} &= \frac{1}{1+r}\left[u_i^n + \frac{r}{2}\left(u_{i-1}^{n+1} + u_{i+1}^{n+1}\right)\right] = \\ &= \frac{1}{1+r}\left[u_i^n + \frac{r}{2}\left(u_{i-1}^n + r\left(\frac{u_{i-2}^n + u_i^n}{2} - u_{i-1}^n\right) + u_{i+1}^n + r\left(\frac{u_{i+2}^n + u_i^n}{2} - u_{i+1}^n\right)\right)\right]. \end{aligned} \tag{14}$$

It is easy to see that

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

Proof for Algorithm 4 (explicit+explicit hopscotch). Using (13) we obtain at the second stage:

$$\begin{aligned} u_i^{n+1} &= u_i^n + r\left(\frac{u_{i-1}^{n+1} + u_{i+1}^{n+1}}{2} - u_i^n\right) = \\ &= \left(1-r\right)u_i^n + \frac{r}{2}\left(u_{i-1}^n + r\left(\frac{u_{i-2}^n + u_i^n}{2} - u_{i-1}^n\right) + u_{i+1}^n + r\left(\frac{u_{i+2}^n + u_i^n}{2} - u_{i+1}^n\right)\right). \end{aligned} \tag{15}$$

It is also easy to see that

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

Proof for Algorithm 5 (UPFD+Explicit Euler odd-even Hopscotch). At Stage 1 we obtain

$$u_{i-1}^{n+1} = \frac{u_{i-1}^n + \frac{r}{2}\left(u_{i-2}^n + u_i^n\right)}{1+r} \quad \text{and} \quad u_{i+1}^{n+1} = \frac{u_{i+1}^n + \frac{r}{2}\left(u_i^n + u_{i+2}^n\right)}{1+r}, \tag{16}$$

which gives at Stage 2 that

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

therefore

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

Proof for Algorithm 6 (UPFD-UPFD odd-even Hopscotch). Using (16) for the values of Stage 1 we obtain for Stage 2

$$u_i^{n+1} = \frac{u_i^n + \frac{r}{2}\left(u_{i-1}^{n+1} + u_{i+1}^{n+1}\right)}{1+r} = \frac{1}{1+r}\left[u_i^n + \frac{r}{2(1+r)}\left(u_{i-1}^n + \frac{r}{2}\left(u_{i-2}^n + u_i^n\right) + u_{i+1}^n + \frac{r}{2}\left(u_i^n + u_{i+2}^n\right)\right)\right] \tag{18}$$

thus

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

which is again the same as (10) up to first order of *r*.

4. Stability of the methods

In this section we examine the stability of the methods by von Neumann stability analysis. For a linear PDE, the exact solution must satisfy the discretized equation exactly, thus the error must also satisfy the same discretized equation. So to perform the analysis, one needs to replace the u_i^n values with the errors ε_i^n into the finite difference formulas of the algorithms such as (6). If the boundary conditions are periodic, the error ε_i^n can be decomposed into a Fourier series as

$$\varepsilon_i^n = \sum_m E_m(t) e^{Ik_m x}$$

or, in an other form,

$$\varepsilon_{i+j}^{\mathrm{n}} = \sum_m E_m(t) e^{Ik_m(x+j\Delta x)},$$

where $j \in \mathbb{Z}$ is the distance from node *i* in grid spacings, $E_m(t)$ is the amplitude of the *m*th harmonic $e^{Ik_m x}$ in the Fourier series of the error and $I = \sqrt{-1}$ [2, p 284-286]. (We note that for other kind of boundary conditions, the finite Fourier integral can be used.) Omitting the *m* index and using the notation $s = k\Delta x$ we can deduce

$$\begin{aligned} &\varepsilon_i^{\mathrm{n}} = E(t)e^{Ikx},\ \varepsilon_i^{\mathrm{n+1}} = E(t+\Delta t)e^{Ikx},\ \varepsilon_{i\pm1}^{\mathrm{n}} = E(t)e^{Ikx\pm\Delta x}, \\ &\frac{\varepsilon_{i-1}^{\mathrm{n}} + \varepsilon_{i+1}^{\mathrm{n}}}{2} = \frac{E(t)\left(e^{Ik(x+\Delta x)} + e^{Ik(x-\Delta x)}\right)}{2} = E(t)e^{Ikx}\frac{e^{Is}+e^{-Is}}{2} = E(t)e^{Ikx}\cos s, \\ &\frac{\varepsilon_{i-2}^{\mathrm{n}} + 2\varepsilon_i^{\mathrm{n}} + \varepsilon_{i+2}^{\mathrm{n}}}{2} = E(t)e^{Ikx}\frac{\left(e^{-2Is}+e^{2Is}+2\right)}{2} = E(t)e^{Ikx}(\cos 2s + 1) = 2E(t)e^{Ikx}\cos^2 s\,. \end{aligned} \tag{19}$$

Finally, substituting these into the formulas defining the algorithms such as (6) and (7), the amplification factor $G = E(t+h)/E(t)$ must be calculated. If the absolute value of this factor exceeds 1, the errors due to e. g. the finite precision of the computer arithmetic will start to grow without limits.

Calculation for Algorithm 1. The discretized equation (6) immediately gives

$$\varepsilon_i^{\mathrm{n+1}} = \frac{\varepsilon_i^{\mathrm{n}} + \frac{r}{2}\left(\varepsilon_{i-1}^{\mathrm{n}} + \varepsilon_{i+1}^{\mathrm{n}}\right)}{1+r}.$$

Substituting formulas listed in (19) and simplifying with e^{Ikx} we can obtain:

$$E(t+\Delta t) = \frac{E(t) + rE(t)\cos s}{1+r}$$

Now the amplification factor is the following simple expression:

$$|G| = \left|\frac{E(t+\Delta t)}{E(t)}\right| = \left|\frac{1+r\cos k\Delta x}{1+r}\right| \le 1,$$

therefore it is obvious that this algorithm is unconditionally stable.

Calculation for Algorithm 2 (UPFD successive displacement modification). Using (12) we can write

$$\varepsilon_i^{\mathrm{n+1}} = \frac{\varepsilon_i^{\mathrm{n}} + \frac{r}{2}\left(\frac{\varepsilon_{i-1}^{\mathrm{n}} + \frac{r}{2}\left(\varepsilon_{i-2}^{\mathrm{n}} + \varepsilon_i^{\mathrm{n}}\right)}{1+r} + \varepsilon_{i+1}^{\mathrm{n}}\right)}{1+r}$$

Using formulas listed in (19) and we can obtain after simplification:

$$E(t+\Delta t)=\frac{E(t)}{1+r}\left(1+\frac{r}{2}e^{Is}\right)+E(t)\frac{r}{2}\frac{e^{-is}+\frac{r}{2}\left(e^{-2Is}+1\right)}{\left(1+r\right)^2}$$

The amplification factor can be calculated as follows

$$G=\frac{1}{1+r}\left\{1+\frac{r}{2}e^{Is}+\frac{r/2}{1+r}\left[e^{-Is}+\frac{r}{2}\left(e^{-2Is}+1\right)\right]\right\}=$$
$$\frac{1}{1+r}\left\{1+\frac{r}{2}\cos s+I\frac{r}{2}\sin s+\frac{r/2}{1+r}\left[\cos s-I\sin s+\frac{r}{2}\left(\cos 2s-I\sin 2s+1\right)\right]\right\}$$

Because of the asymmetry, this is a complex-valued function. We need only the magnitude of this *G* function, thus we calculate its absolute value square first:

$$\left|G\right|^2=\frac{1}{\left(1+r\right)^2}\left[1+\frac{r+r^2/2}{1+r}\cos s+\frac{r^2}{1+r}\frac{\cos^2 s}{2}\right]^2+\frac{1}{\left(1+r\right)^2}\left[\frac{r}{2}\sin s-\frac{r/2}{1+r}\left(\sin s-r\sin s\cos s\right)\right]^2=$$
$$=\frac{1}{\left(1+r\right)^4}\left[1+r+rS+\frac{r^2}{2}S+r^2\frac{S^2}{2}\right]^2+\frac{1-S^2}{\left(1+r\right)^4}\left[\frac{r^2}{2}(1+S)\right]^2$$

The square root of this expression (the *G* function) is presented in Fig. 2. One can see that it is always between zero and one, therefore we can state that Algorithm 2 is also unconditionally stable.

Calculation for Algorithm 3 (Original Hopscotch). Here we emphasize that we examine only one time step. Later we will explain the nontrivial significance of this distinction. Formula (14) yields

$$\varepsilon_i^{n+1}=\frac{1}{1+r}\left[\varepsilon_i^n+\frac{r}{2}\left(\varepsilon_{i-1}^n+r\left(\frac{\varepsilon_{i-2}^n+\varepsilon_i^n}{2}-\varepsilon_{i-1}^n\right)+\varepsilon_{i+1}^n+r\left(\frac{\varepsilon_{i+2}^n+\varepsilon_i^n}{2}-\varepsilon_{i+1}^n\right)\right)\right]=$$
$$\frac{1}{1+r}\left[\varepsilon_i^n+r\left(1-r\right)\frac{\varepsilon_{i-1}^n+\varepsilon_{i+1}^n}{2}+r^2\frac{\varepsilon_{i-2}^n+\varepsilon_i^n+\varepsilon_{i+2}^n}{4}\right]$$

Using formulas (19) we obtain:

$$E(t+\Delta t)e^{Ikx}=\left[E(t)e^{Ikx}+r\left(1-r\right)E(t)e^{Ikx}\cos s+r^2E(t)e^{Ikx}\cos^2 s\right]\left(1+r\right)^{-1}.$$

The amplification factor takes the following form:

$$G=\frac{1+r\left(1-r\right)S+r^2S^2}{1+r}.$$

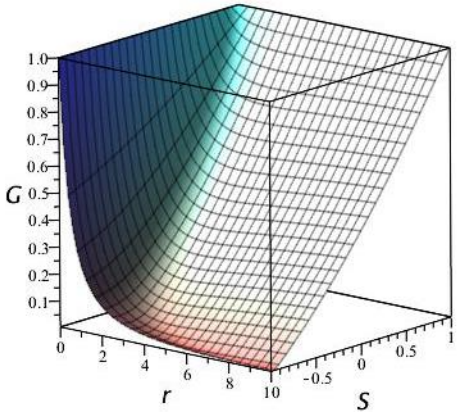

Figure 2. *The graph of the absolute value of the stability function G for Algorithm 2,* $r = 2\alpha h / \Delta x^2$, $S = \cos k\Delta x$.

This function is plotted on the left side of Fig. 3. From the figure it can be seen that for $S = -1$, G exceeds 1 for increasing r thus one could conclude that this algorithm is only conditionally stable. To be more specific, $G(S=-1) = (1 - r + 2r^2)/(1+r)$, which takes the value 1 at r=1. It means that, according to this analysis, the stability is guaranteed only if $r = \frac{2\alpha h}{\Delta x^2} \le 1$ i.e. $h \le \frac{1}{2}\Delta x^2$ for α=1. This condition is the same as the well-known limit for the Explicit Euler method. We will see that the numerical experiences show that this algorithm is unconditionally stable. This apparent contradiction deserves some explanation. When Gordon proved the unconditional stability of this algorithm (following a different way of calculations), he considered it as a two-step method with doubled time step size [12]. This makes a difference because the roles of the odd and even nodes are interchanged and indeed, the second step stabilizes the whole algorithm. We stress that this does not mean that our calculation has no sense as it serves with a false result, because, contrary to the outcome of it, the algorithm is stable. But, as we will see in Part 2 and 3, the algorithm is highly inaccurate for large time step sizes and the analysis presented here helps to reveal the reason of this inaccuracy.

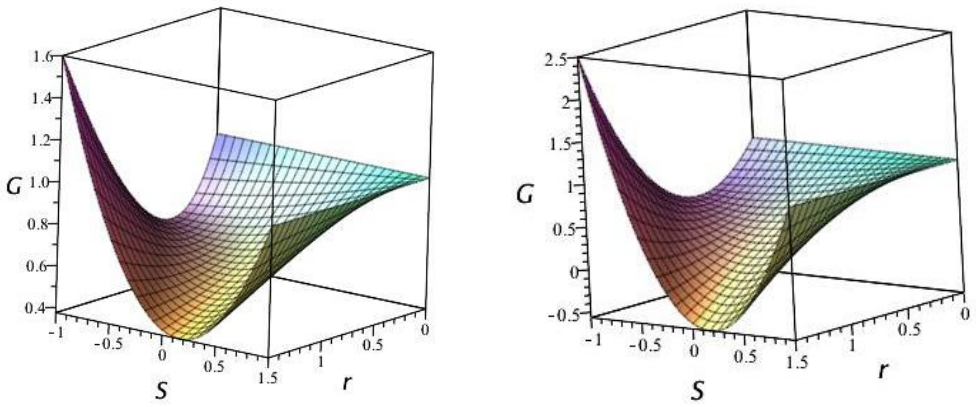

Figure 3. *The graph of the stability functions G, where* $r = 2\alpha h / \Delta x^2$, $S = \cos k\Delta x$.
Left side: Algorithm 3. Right side: Algorithm 4.

Calculation for Algorithm 4 (explicit+explicit hopscotch). Using (15) we can immediately write

$$\varepsilon_i^{n+1} = (1-r)\varepsilon_i^n + \frac{r}{2}\left(\varepsilon_{i-1}^n + r\left(\frac{\varepsilon_{i-2}^n + \varepsilon_i^n}{2} - \varepsilon_{i-1}^n \right) + \varepsilon_{i+1}^n + r\left(\frac{\varepsilon_{i+2}^n + \varepsilon_i^n}{2} - \varepsilon_{i+1}^n \right) \right),$$

which, after substituting formulas (19) gives:

$$E(t+\Delta t) = (1-r)E(t) + \frac{r}{2}\left[2E(t)(1-r)\cos s + 2E(t)r\cos^2 s \right].$$

The amplification factor:

$$G = 1 - r + r(1-r)\cos s + \frac{r^2}{2}\cos^2 s = 1 - r + r(1-r)S + r^2 S^2.$$

It is easy to solve the G=1 equation by hand calculation, and it has two solutions r=0 and $r = -\frac{1}{S}$. We present the graph of the $G = G(r,S)$ function on the right side of Fig. 3. It can be seen that G is constantly 1 for $S = \cos s = \cos k\Delta x = 1$, when k=0. This reflects the behaviour of the constant error-function $\varepsilon_i^n = E(t)e^0$: the scheme preserves this error as the central difference formula gives zero for this constant function. On the other hand, as r reaches 1, G reaches and then exceeds 1 for $S = -1$, which yields that the critical value of r is 1, i.e. the stability is guaranteed if $r = \frac{2\alpha h}{\Delta x^2} \le 1$ i.e. $h \le \frac{1}{2}\Delta x^2$ for α=1, the same as in the case of the previous algorithm. However, the figure shows that now G increases much faster for increasing r when $S = -1$ than in the case of Algorithm 3. From this we can conclude that either this algorithm is more inaccurate or even unstable for larger time step sizes. The numerical experiences in part 2 and 3 will clearly confirm the second option, i.e. that this algorithm is indeed only conditionally stable, but the threshold for the time step size h is larger. We point out again that for a more exact analysis of stability, two time steps should be investigated, as the second time step is not identical to the first one because of the roles of the odd and even nodes are interchanging step by step.

Calculation for Algorithm 5 (UPFD+Explicit Hopscotch). From (17) we obtain

$$\varepsilon_i^{n+1} = (1-r)\varepsilon_i^n + \frac{r}{2}\left(\frac{\varepsilon_{i-1}^n + \frac{r}{2}\left(\varepsilon_{i-2}^n + \varepsilon_i^n\right)}{1+r} + \frac{\varepsilon_{i+1}^n + \frac{r}{2}\left(\varepsilon_i^n + \varepsilon_{i+2}^n\right)}{1+r} \right) =$$

$$= (1-r)\varepsilon_i^n + \frac{r}{1+r}\frac{\varepsilon_{i-1}^n + \varepsilon_{i+1}^n}{2} + \frac{r^2}{1+r}\frac{\varepsilon_{i-2}^n + 2\varepsilon_i^n + \varepsilon_{i+2}^n}{4},$$

which yields:

$$E(t+\Delta t)e^{ikx} = (1-r)E(t)e^{ikx} + \frac{r}{1+r}E(t)e^{ikx}\cos s + \frac{r^2}{1+r}E(t)e^{ikx}\cos^2 s.$$

The amplification factor:

$$|G| = 1 - r + \frac{rS + r^2 S^2}{1+r}.$$

As one can see in the left side of Fig. 4, this factor can be smaller than -1 for large time step sizes, thus one could conclude that this algorithm is only conditionally stable. However, according to the numerical experiences this algorithm is very similar to Algorithm 3 and seems to be unconditionally stable, therefore we can reason that this one-step stability analysis is not sufficient here.

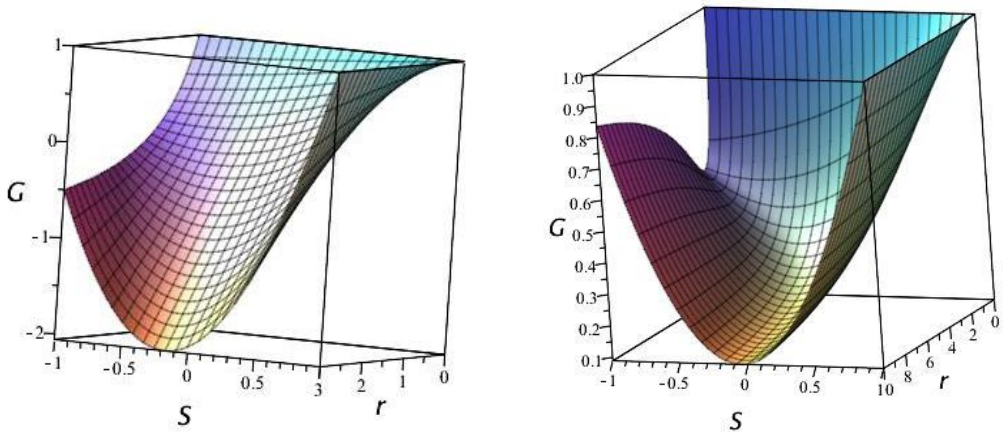

Figure 4. *The graph of the stability functions G, where* $r = 2\alpha h / \Delta x^2$, $S = \cos k\Delta x$. *Right side: Algorithm 5. Left side: Algorithm 6.*

Calculation for Algorithm 6 (UPFD-UPFD Hopscotch). From (18) we obtain

$$\varepsilon_i^{n+1} = \frac{1}{1+r}\left[\varepsilon_i^n + \frac{r}{2}\left(\frac{\varepsilon_{i-1}^n + \frac{r}{2}\left(\varepsilon_{i-2}^n + \varepsilon_i^n\right)}{1+r} + \frac{\varepsilon_{i+1}^n + \frac{r}{2}\left(\varepsilon_i^n + \varepsilon_{i+2}^n\right)}{1+r}\right)\right],$$

from which it readily follows that

$$E(t+\Delta t) = \frac{E(t)}{1+r} + \frac{r}{2}\frac{2E(t)\cos s + \frac{r}{2}E(t)\left(e^{-2is} + 2 + e^{2is}\right)}{(1+r)^2},$$

thus the amplification factor:

$$G = \frac{1}{1+r} + \frac{r}{2}\frac{2\cos s + r(1+\cos 2s)}{(1+r)^2} = \frac{1}{1+r} + r\frac{S + rS^2}{(1+r)^2}.$$

The value of the G function in S=1 is 1. We can calculate

$$\frac{dG}{dS} = \frac{r}{(1+r)^2}\frac{d}{dS}\left(S + rS^2\right) = \frac{r}{(1+r)^2}(1+2rS).$$

Now if *S* starts to decrease from 1, *G* will also decrease. At $S=-1/2r$ the value of *G* attains its minimum and will start to increase again, but it is easy to see that it does not reach 1 again. It is also easy to see that at $S=-1/2r$ the value of *G* is

$$G\left(S=-1/2r\right)=\frac{1}{1+r}+\frac{-1/2+r/4}{\left(1+r\right)^2},$$

which cannot be smaller than 0, and this means that the method is stable for any *r*. We present the *G* function on the right side of Fig. 4.

5. Positivity of the methods

Theorem 2. *In case of Algorithms 1, 2 and 6, the new values* u_i^{n+1} *are the convex combinations of the old values* u_j^n, $j=1,...,N$.

Proof: For algorithm 1 it is easy to see that the coefficients of u_i^n, u_{i-1}^n, and u_{i+1}^n in the expression (6) of u_i^{n+1} are the following:

$$\frac{1}{1+r}, \frac{r/2}{1+r}, \text{ and } \frac{r/2}{1+r},$$

which are all nonnegative, not larger than 1 and their sum is 1. Algorithm 2 and 6 are more difficult from this point of view, because at Stage 2 the already calculated u_{i-1}^{n+1} or u_{i-1}^{n+1} and u_{i+1}^{n+1} values are used, respectively. Thus we have to recall the following simple lemma, the associativity of convex combinations [36, p. 28], on which we build during the proof of the remaining part of the theorem.

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

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

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

In case of Algorithm 2 and 6 the values u_{i-1}^{n+1} or u_{i-1}^{n+1} and u_{i+1}^{n+1}, which have been calculated at Stage 1 by the UPFD method, are convex combinations of the old values. Now the lemma immediately implies that the values obtained at Stage 2 are also convex combinations of the same old values.

Corollary 1: Algorithms 1, 2 and 6, when applied to Eq. (1) are not only preserve positivity, but satisfy the Maximum and Minimum principle [20, p. 87], i.e. the extreme values of the function *u* occur among the initial or the prescribed boundary values, which is a physical property of heat conduction (without external heat sources) due to the second law of thermodynamics.

Corollary 2: The previous corollary, namely the fulfilment of the Maximum and Minimum principle implies the stability for Algorithms 1, 2 and 6, therefore the considerations explained in this section give a second, independent proof of the stability of these algorithms.

Remark: As Algorithms 3, 4 and 5 contains an Explicit Euler step, they obviously cannot be positivity-preserving. We note that it does not mean that they are always less accurate, as we will see in the next two parts.

6. Summary

Using two known, but non-conventional algorithms, namely the UPFD and the odd-even hopscotch methods, we constructed four new algorithms for the numerical solution of the homogeneous heat equation. In this part of the series, we examined all the 6 algorithms analytically, while the numerical experiments will be presented in Part 2 and 3. 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 by large time step sizes. However, only Algorithm 4 can be unstable in reality, which implies that 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 interchange 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 still can be used to explain the inaccuracy of these methods for large time step size, which is observed in the numerical experiments. A similar problem emerged with the order of convergence: we will see that two of the algorithms are in fact second order, but this could be proven only by calculations incorporating two time steps. We note that in the Summary of Part 3 a detailed comparison of the properties and performance of the six methods will be presented.

References

[1] Cengel, Y. A., Ghajar, A. J.: *Heat and Mass Transfer*, Fundamentals & Application, Fifth Edition in SI Units. New York: McGraw-Hill Science/Engineering/Math, 2015.

[2] Özişik, M. N.: *Finite Difference Methods in Heat Transfer*, CRC Press, 2017. **https://doi.org/10.1201/9781315168784**

[3] Agbavon, K. M., Appadu, A. R.: C*onstruction and analysis of some nonstandard finite difference methods for the FitzHugh–Nagumo equation*, Numer. Methods Partial Differ. Equ., vol. 36, no. 5, (2020) pp. 1145–1169. **https://doi.org/10.1002/num.22468**

[4] Mebrate, B.: *Numerical solution of a one dimensional heat equation with dirichlet boundary conditions*, Am. J. Appl. Math., vol. 3, no. 6, (2015) pp. 305–311. **https://doi.org/10.11648/j.ajam.20150306.20**

[5] Heydari, M. H.: *Numerical solution of the one-dimensional heat equation by using Chebyshev Wavelets Method*, J. Appl. Comput. Math., vol. 1, no. 6, 2012, **https://doi.org/10.4172/2168-9679.1000122**

[6] Kudryashov, N. A.: *One method for finding exact solutions of nonlinear differential equations*, Commun. Nonlinear Sci. Numer. Simul., vol. 17, no. 6, (2012) pp. 2248–2253. **https://doi.org/10.1016/j.cnsns.2011.10.016**

[7] Hosseini, K., Ansari, R., Gholamin, P.: *Exact solutions of some nonlinear systems of partial differential equations by using the first integral method*, J. Math. Anal. Appl., vol. 387, no. 2, (2012) pp. 807–814. **https://doi.org/10.1016/j.jmaa.2011.09.044**

[8] 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., vol. 25, no. 2, (2020) pp. 241–256. **https://doi.org/10.3846/mma.2020.10459**

[9] Barna, I. F., Guedda, M., Bognár, G., Hriczó, K., Mátyás, L.: *Analytic traveling-wave solutions of the Kardar-Parisi-Zhang interface growing equation with different kind of noise terms*, in Differential and Difference Equations with Applications. ICDDEA 2019., 2019, **https://doi.org/10.1007/978-3-030-56323-3_19**

[10] Barna, I. F., Bognár, G., Hriczó, K.: *Self-similar analytic solution of the two-dimensional Navier-Stokes equation with a non-newtonian type of viscosity*, Math. Model. Anal., vol. 21, no. 1, (2016) pp. 83–94. **https://doi.org/10.3846/13926292.2016.1136901**

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

[12] 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**

[13] 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**

[14] 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**

[15] Chapra, S. C., Canale, R. P.: *Numerical methods for engineers*, Seventh Edition, 7th ed. New York: McGraw-Hill Science/Engineering/Math, 2015.

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

[17] Abdul Ghaffar, Z. S., Alias, N., Sham Ismail, F., Mohamed Murid, A. H., Hassan, H.: *Sequential algorithm of parabolic equation in solving thermal control process on printed circuit board*, Malaysian J. Fundam. Appl. Sci., vol. 4, (2008) pp. 379–385. **https://doi.org/10.11113/mjfas.v4n2.46**

[18] Hirsch, C.: *Numerical computation of internal and external flows*, volume 1: Fundamentals of numerical discretization. Wiley, 1988.

[19] Hiriart-Urruty, J.-B., Lemaréchal, C.: *Fundamentals of convex analysis*. Berlin: Springer Verlag, 2001. **https://doi.org/10.1007/978-3-642-56468-0**

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