UNIVERSITY OF MISKOLC
FACULTY OF MECHANICAL ENGINEERING AND INFORMATICS

DESIGNING AND INVESTIGATING NUMERICAL METHODS FOR SOLVING THE HEAT CONDUCTION EQUATION

PHD THESES

Prepared by

Mahmoud Saleh
Engineering of Mechanical (BSc),
Engineering of Manufacturing (MSc)

ISTVÁN SÁLYI DOCTORAL SCHOOL OF MECHANICAL ENGINEERING SCIENCES

TOPIC FIELD OF BASIC ENGINEERING SCIENCES
TOPIC GROUP OF TRANSPORT PROCESSES AND MACHINES

Head of Doctoral School

Dr. Gabriella Bognár
DSc, Full Professor

Head of Topic Group

Dr. László Baranyi
Full Professor

Scientific Supervisor

Dr. Endre Kovács

Miskolc
2023

CONTENTS

SUPERVISOR'S RECOMMENDATIONS

Date

Supervisor

LIST OF SYMBOLS AND ABBREVIATIONS

GREEK LETTERS

LATIN LETTERS

SUBSCRIPTS

1. INTRODUCTION

1.1. The Application and the Importance of Heat Conduction

In many areas of engineering, the study of mass and heat transfer is of fundamental importance. Understanding the physical principles underlying the different heat transfer modes is crucial for engineers, as is being able to calculate the amount of energy transported per unit time using the rate equations [1]. To enhance the efficiency of the equipment like condensers, boilers, economizers in a thermal power plant and air pre-heaters, a mechanical engineer may be interested in understanding the mechanisms of heat transfer involved in their operations. Due to the importance of safe operation in their design, nuclear power facilities require accurate information on heat transfer. Systems for refrigeration and air conditioning also include heat-exchanging components, which require careful design. Electrical engineers are concerned about preventing material damage to electric motors, generators, and transformers caused by hot spots created by incorrect heat transfer design. An electronic engineer is interested in effective ways to remove heat from chips and semi-conductor devices so that they can operate at temperatures that are safe. Considering how rapidly computing devices are becoming smaller, a computer hardware engineer is interested in the cooling needs of circuit-boards. In the field of chemical engineering, researchers are interested in the process of heat and mass transfer in different chemical reactions. The rate of heat transfer necessary for a certain treatment method is a point of interest to a metallurgical engineer. For example, the rate of cooling during the casting process has a significant impact on the quality of the final product. The rate of heat transfer in the heat shields used in re-entry vehicles and in rocket nozzles is of interest to aeronautical engineers. Food processing, grain drying, and preservation are all of interest to an agricultural engineer. A civil engineer is aware of the effect of the heat transfer on buildings and the thermal stresses developed in structures. A concern of an environmental engineer is the influence of heat on the dispersion of pollutants in the air, their transport through soils, lakes, and oceans, and their effects on life. A bioengineer is generally concerned with the heat and mass transfer mechanisms that occur within the human body, including hypothermia and hyperthermia [2], [3].

The applications of heat and mass transfer outlined above are only a few examples. The principal factors for existence of life on Earth are the solar system and the related energy transfer from the sun. It is true to say that it is very difficult, if not impossible, to completely avoid heat transfer in any process taking place on Earth.

Many critical problems that arise in many engineering equipment can be solved effectively and economically by analysing the mechanisms of the heat and mass transfer. We may take the development of heat pipes as an example. These pipes can transport heat at a rate that is significantly higher than that of copper or silver rods of the same diameters, even at roughly isothermal conditions. Minimizing heat gain in the summer and heat loss in the winter is the foundation of energy-efficient house design. By designing efficient cooling systems, it is possible to develop modern gas turbine blades where the gas temperature is higher than the melting point of the blade material. We show another example of a successful heat transfer design. The design of computer chips, which experience heat flux similar to that seen in re-entry vehicles, is once again a success story in heat transfer design, particularly when the surface temperature of the chips is constrained to less than 100 ℃ [3], [4] .

Although there are many successful heat transfer designs, further developments on heat and mass transfer studies are necessary in order to increase the life span and efficiency of the many devices discussed previously, which can lead to many more new inventions. Even though there are many efficient heat-transfer designs, more work has to be done on heat and mass transfer research to increase the lifespan and efficiency of the devices we have already covered and maybe inspire the development of new technologies [3].

1.2. Governing Differential Equation of Heat Conduction

The differential control volume is defined in Figure 1.1 for the Cartesian coordinate system. The corresponding volume and mass of the differential control volume are defined, respectively, as

$$dv = dxdydz \quad \text{and} \quad dm = \rho dxdydz, \tag{1.1}$$

where ρ is the mass density $\left(kg/m^3\right)$ of the control volume. The differential approach will assume a continuum such that all properties do not change microscopically. Neglecting any changes in the kineticand potential energy of the control volume, and applying the above assumptions, conservation of energy can be expressed as follows

$$\begin{pmatrix}\text{Net rate of}\\ \text{heat transfer by}\\ \text{conduction}\end{pmatrix}+\begin{pmatrix}\text{Rate of}\\ \text{energy}\\ \text{generation}\end{pmatrix}=\begin{pmatrix}\text{Rate of}\\ \text{change of}\\ \text{internal energy}\end{pmatrix}.$$

The rate of change of internal energy within the control volume is $\rho c\frac{\partial T}{\partial t}dxdydz$ [5]. The expression of conservation of energy can be written mathematically as follows

$$\delta\dot{Q}+\delta\dot{E}_{gen}=\rho c\frac{\partial T}{\partial t}dxdydz, \tag{1.2}$$

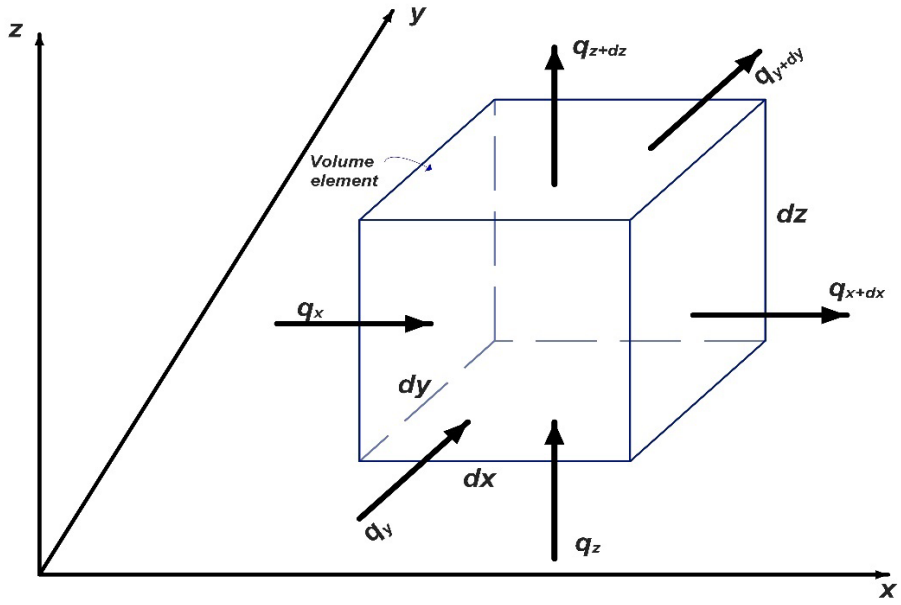

Figure 1.1. Differential control volume for derivation of the heat conduction in cartesian coordinate.

where $\delta\dot{E}_{gen}$ (W) represents the rate of energy generation within the conrol volume, and $\delta\dot{Q}$ (W) represnts the net rate of transfer into the control volume due to the conduction, with positive $\delta\dot{Q}$ representing heat transfer into the system.

The net rate of heat transfer in and out of the control volume, the first term in Eq. (1.2), is given as follows

$$\delta\dot{Q}=\left(q_x-q_{x+dx}\right)+\left(q_y-q_{y+dy}\right)+\left(q_z-q_{z+dz}\right), \tag{1.3}$$

where the heat fluxes terms entering the control volume can be calculated using Fourier's law

$$q_x=-kA_x\frac{\partial T}{\partial x}\qquad\text{where}\qquad A_x=dydz, \tag{1.4}$$

$$q_y=-kA_y\frac{\partial T}{\partial y}\qquad\text{where}\qquad A_y=dxdz, \tag{1.5}$$

$$q_z = -kA_z \frac{\partial T}{\partial z} \qquad \text{where} \qquad A_z = dxdy. \tag{1.6}$$

The heat fluxes exiting the control volume can be calculated using Taylor series. Negleting Higher-order terms, for x direction the term can be written as follows

$$q_{x+dx} = q_x + \frac{\partial q_x}{\partial x} dx = -kA_x \frac{\partial T}{\partial x} + \frac{\partial}{\partial x}\left(-kA_x \frac{\partial T}{\partial x} \right) dx. \tag{1.7}$$

Using Eqs. (1.4) and (1.7), the net rate of heat transfer in x direction can be written

$$q_x - q_{x+dx} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x} \right) dxdydz. \tag{1.8}$$

In similar way, the net heat transfer in x and y directions can be calculated

$$q_y - q_{y+dy} = \frac{\partial}{\partial y}\left(k \frac{\partial T}{\partial y} \right) dxdydz, \tag{1.9}$$

$$q_z - q_{z+dz} = \frac{\partial}{\partial z}\left(k \frac{\partial T}{\partial z} \right) dxdydz. \tag{1.10}$$

We substitute Eqs. (1.8), (1.9), and (1.10) into Eq. (1.3) to get

$$\delta \dot{Q} = \left(q_x - q_{x+dx} \right) + \left(q_y - q_{y+dy} \right) + \left(q_z - q_{z+dz} \right).$$

The rate of energy generation within the conrol volume can be calculated considering the volumetric rate of internal energy generation $g \left(\mathrm{W}/\mathrm{m}^3 \right)$

$$\delta \dot{E}_{gen} = g dxdydz. \tag{1.12}$$

Eqs. (1.11) and (1.12) can be introduced into Eq. (1.2) in order to provide the general heat equation

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y}\left(k \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z}\left(k \frac{\partial T}{\partial z} \right) + g. \tag{1.13}$$

Using vector notation, the previous equation can be expressed [6], [7]

$$\rho c \frac{\partial T}{\partial t} = \nabla \left(k \nabla T \right) + g, \tag{1.14}$$

where ∇ is the differential vector operator, $\nabla = \vec{i}\left(\frac{\partial}{\partial x} \right) + \vec{j}\left(\frac{\partial}{\partial y} \right) + \vec{k}\left(\frac{\partial}{\partial z} \right)$.

If the thermal conductivity is constant, Eq. (1.14) can be reduced to the form

$$\frac{1}{\alpha} \frac{\partial T}{\partial t} = \nabla^2 T + \frac{g}{k}, \tag{1.15}$$

where $\alpha = \frac{k}{\rho c}$ is the thermal diffusivity of the medium.

The principle statement of the heat equation is that in the presence of different temperatures, heat flows occur, which finally lead to a temperature equalization. The analogous situation is also found with concentration differences in substances. Due to such concentration differences, mass flows occur, which lead to an equalization of the concentrations. So, under more general circumstances, Eq. (1.14) can be written in the general form

$$c\rho \frac{\partial u}{\partial t} = \nabla\left(k \nabla u\right) + c\rho q, \tag{1.16}$$

where $u : \mathbb{R}^3 \times \mathbb{R} \mapsto \mathbb{R};\ \left(\vec{r},t\right) \mapsto u\left(\vec{r},t\right),\ u\left(\vec{r},t=0\right) = u^0\left(\vec{r}\right)$ and $q = q\left(\vec{r},t\right)$, while $c = c\left(\vec{r},t\right)$, $\rho = \rho\left(\vec{r},t\right)$, and $k = k\left(\vec{r},t\right)$ are known nonnegative functions. In case of diffusive mass transfer, $u = u\left(x,t\right)$ or $u = u\left(\vec{r},t\right)$ is the concentration of the particles. In case of heat conduction, u refers to the temperature, $\alpha = k/(c\rho)$ is the thermal diffusivity of the medium, k, ρ, and c are the heat conductivity, the specific heat, and the mass density, while q is the intensity of the heat sources (due to electric currents, electromagnetic radiation, etc.), respectively. Here, we must emphasize that the term $g\left[\mathrm{W}/\mathrm{m}^3\right]$ in Eq. (1.14) is not equivalent to the term $q\left[\mathrm{K}/\mathrm{s}\right]$ in Eq. (1.16).

The generalizations of the diffusion equation, such as the advection-diffusion-reaction equation can model mass transport in different fields of science such as biology, chemistry, and physics. For example, the proteins in embryos [8], the atoms in carbon nanotubes [9], and the charge carriers in semiconductors [10]. Moreover, very similar equations or system of equations have been used to model the fluid flow through porous media, such as ground water, crude oil in underground reservoirs [11], and the moisture [12].

1.3. Numerical Methods for Solving Heat Conduction Equation

The analytical solutions for the heat conduction equation are available usually when the problems are highly simplified in simple geometry. Not only the geometry but also the boundary and initial conditions can make the analytical solution impossible [4]. different numerical methods have been used to solve the heat conduction problems. Some of these methods were applied before the development of the powerful digital computers, and as a result they are no longer necessary. Since the development of powerful computers, the finite difference method (FDM) and the finite element method (FEM) have been the two numerical approaches which have received the most success and popularity [7].

Variety of finite difference methods have been developed and applied to solve the PDEs, such as the heat conduction equation and its generalization, for example, the explicit method, the fully implicit method, and the ADI method [13]–[15]. One of the most common approaches to solve the

PDEs numerically is to discretize the spatial variable which converts the PDEs into a system of ODEs. After that we can solve the system of ODEs at each time level [16]. Nevertheless, most of these methods are tested and evaluated under conditions where the coefficients in the equations, such as the diffusivity α, are independent of the space variable. However, there are systems in real applications where the physical properties of material can be drastically different at adjacent points, for instance in a microprocessor. As a result, the coefficients and, consequently, the eigenvalues of the matrix system can have a range of several order of magnitude so the problem can be extremely stiff.

The traditional explicit methods, either Adams-Bashforth or Runge-Kutta types, are conditionally stable, very small time-step sizes must be applied regardless of the measurement errors of the input data and the requirements on the accuracy of the output. It implies that the solution blows up if the time-step size exceeds the threshold number, or what is known as the CFL limit. Even the professional commercial adaptive time-step size solvers such as ode23 and ode45 of MATLAB can experience instability when the tolerance is not so small [17].

On the other hand, implicit methods offer much better stability properties, and this why they are widely applied to solve these equations.[18]–[20]. For example, in his work [21] Mascagni applied the backward Euler method to the Hodgkin-Huxley equations. Manaa and Sabawi studied and compared the explicit Euler and implicit Crank-Nicolson methods when they are applied to Huxley equation. They found that the explicit Euler method was faster, but less stable and accurate than Crank-Nicolson [22]. Coupled hydrodynamics and nonlinear heat conduction problems were solved numerically by Kadioglu and Knoll by treating the heat equation implicitly and the hydrodynamics explicitly [23]. They mentioned that this technique, the so called IMEX, is typical for solving such kind of problems. Another example is in the field of reservoir-simulation, where the pressure equation is treated implicitly, and saturation equation is treated explicitly [24]. However, Lee and Tene used a fully implicit method to solve the problem of reservoir-simulation [25].

The most significant problem with implicit method is that each time step requires the solution of a system of algebraic equations, which cannot simply be parallelized. In case of one-dimensional system where the matrix is tridiagonal and the number of nodes is small, the numerical computation can be fast, and it is hard to beat the implicit method. In contrast, the numerical computations can be time-consuming when we use the implicit method to solve mor complicated systems, such as reservoir-simulation with one trillion cells. However, there is an increasing trend towards parallelism in the recent years [26], [27], which can be considered as an advantage for the explicit methods.

The second problem with most implicit or explicit methods is that they can lead to qualitatively unacceptable solutions, such as unphysical oscillations or negative values of the otherwise non-negative variables. These variables can be concentrations, densities, or temperatures measured in Kelvin, and the numerical methods should preserve their positivity. To overcome this problem, Chen-Charpentier and others developed and investigated the fully explicit and unconditionally

positivity preserving finite difference (UPFD) scheme in order to solve advection-diffusion reaction equations [28]–[30]. After that Kolev treated a model of cancer migration and invasion. That model consists of an ordinary differential equation and two partial differential equations with diffusion terms. They discretized the ordinary equation and one of the partial differential equations using implicit scheme, while the other partial differential equation was solved by an explicit scheme which is similar to (UPDF) scheme [31]. However, their method, as well as the original UPFD scheme, has only first order temporal accuracy. Another positivity preserving scheme was developed by Chertock and Kurganov to solve a system of advection-reaction-diffusion equations which describes chemotaxis model. However, their method is positivity preserving only if the time step size does not exceed the CFL limit [32].

There exist unconditionally stable explicit methods in the case of linear heat equation. For instance, the odd-even hopscotch algorithm [33], [34] and the Alternating Direction Explicit (ADE) scheme [35] both have second order temporal accuracy.

In my research, I worked with my supervisor on improving and investigating families of conventional and novel explicit methods for solving linear and nonlinear diffusion equation based on fundamentally new way of thinking. In some cases, the improved methods are proven to be unconditionally stable, positivity preserving. Those schemes are applied to extremely stiff and inhomogeneous systems. Also, some adaptive time step controllers are constructed and investigated.

1.4. Outline of the Thesis

In Chapter 2, the fully explicit discretization for the spatial variables is illustrated. The novel explcit methodd, constant neighbour CNe the linear neighbour LNe3 methods, are also discussed in this chapter. In Chapter 3, novel odd-even hopscotch-type numerical shcemes are introduced. Systematic numerical experiments are conducted to investigate the performance of those methods. In Chapter 4, families of adaptive time-step controllers are proposed to solve the heat conduction equation. The performance of the I-type controllers and the PI-type controllers is investigated. In Chapter 5, adaptive controllers of type I are suggested to solve the time-dependent diffusion equation in one dimension, where the diffusion coefficient itself depends simultaneously on space and time.

2. EXPLICIT DISCRETIZATION AND EXPLICIT METHODS

In this chapter I will introduce a fully explicit discretization for the space variables in the diffusion equation which converts the PDE into a system of ODEs. Later, the methods we construct will be applied to the resulted system of ODEs. Also, I will illustrate briefly in this chapter some conventional and novel methods which are the cornerstone of all the schemes I will construct in subsequent chapters.

2.1. A Fully Explicit Discretization

The second order linear parabolic partial diffusion equation, or the so-called heat can be written as follows

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + q, \tag{2.1}$$

Where u is the temperature (the concentration in case of diffusion-equation), $\alpha = k/(c\rho) > 0$ is the thermal diffusivity, q, k, c, and ρ are the intensity of heat sources (chemical reactions, radioactive decay, radiation, etc.), heat conductivity, specific heat and (mass) density, respectively.

To solve the heat equation numerically, the most common starting step is the same as in the standard method of lines. The most typical finite difference scheme to discretize the space variable is the second order central difference formula [36]

$$\frac{\partial^2}{\partial x^2} f(x_{\mathrm{i}}, t_{\mathrm{j}}) \approx \frac{\dfrac{f(x_{\mathrm{i+1}}, t_{\mathrm{j}}) - f(x_{\mathrm{i}}, t_{\mathrm{j}})}{\Delta x} + \dfrac{f(x_{\mathrm{i\text{-}1}}, t_{\mathrm{j}}) - f(x_{\mathrm{i}}, t_{\mathrm{j}})}{\Delta x}}{\Delta x}. \tag{2.2}$$

The author in [37] presented the discretization more generally than traditional numerical analysis textbooks often did, for example, he did not consider α, k, c and ρ as spatially uniform, because the discretization must represent the material properties of the real life systems. In other words, instead of treating Eq. (2.1) he treated the more general formula

$$c\rho \frac{\partial u}{\partial t} = \nabla \left(k \nabla u \right) + c\rho q. \tag{2.3}$$

In my numerical experiments I will treat inhomogeneous systems as we will see. That is why I find it is necessary to introduce the generalized formula of discretized space. Using Eq. (2.2) for a one-dimensional, equidistant grid, we get

$$\left.\frac{\partial u}{\partial t}\right|_x = \frac{1}{\Delta x \cdot c\big|_x \, \rho\big|_x} \left(k\Big|_{x+\frac{\Delta x}{2}} \cdot \frac{u(x+\Delta x) - u(x)}{\Delta x} + k\Big|_{x-\frac{\Delta x}{2}} \cdot \frac{u(x-\Delta x) - u(x)}{\Delta x} \right) + q\big|_x. \tag{2.4}$$

To simplify, we will use the index i

$$\frac{du_{\mathrm{i}}}{dt}=\frac{A}{c_{\mathrm{i}}\rho_{\mathrm{i}}A\Delta x}\left(k_{\mathrm{i,i+1}}\cdot\frac{u_{\mathrm{i+1}}-u_{\mathrm{i}}}{\Delta x}+k_{\mathrm{i-1,i}}\cdot\frac{u_{\mathrm{i\text{-}1}}-u_{\mathrm{i}}}{\Delta x}\right)+Q_{\mathrm{i}},\qquad(2.5)$$

where u_i is the temperature of the cell i, $C=c.m=\rho cV$ is the heat capacity of that cell in $\left(\mathrm{J/K}\right)$ units (m is the mass, $V=A\Delta x$ is the volume of the cell). We introduce two other quantities, the heat source term Q,

$$Q_{\mathrm{i}}=\frac{1}{V_{\mathrm{i}}}\int_{V_i}qdV\approx q\ \text{ in }\left[\frac{K}{s}\right]\text{units},$$

and the thermal resistance $R_{\mathrm{ij}}=\frac{\Delta x}{k_{\mathrm{ij}}A}$ in $\left(K/W\right)$ units. In case of nonequidistant grid, the distances between the center of cells are $d_{\mathrm{ij}}=\left(\Delta x_{\mathrm{i}}+\Delta x_{\mathrm{j}}\right)/2$ and the resistances can be calculated by the simple approximation as $R_{\mathrm{ij}}\approx\frac{d_{\mathrm{ij}}}{k_{\mathrm{ij}}A_{\mathrm{ij}}}$. Using the introduced quantities and Eq. (2.5) we can write

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

In case of homogeneous one-dimensional system with equidistance grid, the previous equation can be written as follows

$$\frac{du_{\mathrm{i}}}{dt}=\alpha\frac{u_{\mathrm{i\text{-}1}}-2u_{\mathrm{i}}+u_{\mathrm{i+1}}}{\left(\Delta x\right)^2}+Q_{\mathrm{i}}.\qquad(2.6)$$

We prefer to use the ODE system for a general (perhaps unstructured) grid, which gives the time derivative of each temperature independently of any coordinate-system:

$$\frac{du_{\mathrm{i}}}{dt}=\sum_{\mathrm{j}\neq\mathrm{i}}\frac{u_{\mathrm{j}}-u_{\mathrm{i}}}{R_{\mathrm{i,j}}C_{\mathrm{i}}}+Q_{\mathrm{i}}.\qquad(2.7)$$

Which can be written in matrix form

$$\frac{d\vec{u}}{dt}=M\vec{u}+\vec{Q},$$

where the diagonal element of matrix M can be written as follows

$$m_{ii}=\sum_{j\in\mathrm{neighbours}(i)}\frac{-1}{R_{i,j}C_i}=\frac{-1}{\tau_i}.\qquad(2.9)$$

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

$$\tau_i=\frac{\Delta x_i^2}{2\alpha}\ ,\ (1<i<N).\qquad(2.10)$$

The off-diagonal $m_{ij} = 1/\left(R_{ij}C_i\right)$ element of the M matrix can be nonzero only if the cells i and j are neighbours. From this point, all summations are going over the neighbours of the actual cell, which will be denoted by $j \in n(i)$. Unless stated otherwise, we consider closed (zero Neumann) boundary conditions, i.e., the edge of the examined domain is thermally isolated regarding conductive type heat transfer. To help the reader to imagine, we present the arrangement of the variables in Figure 2.1 for a 2D system of 4 cells. We emphasize that the shape and arrangement of the cells are not necessarily regular.

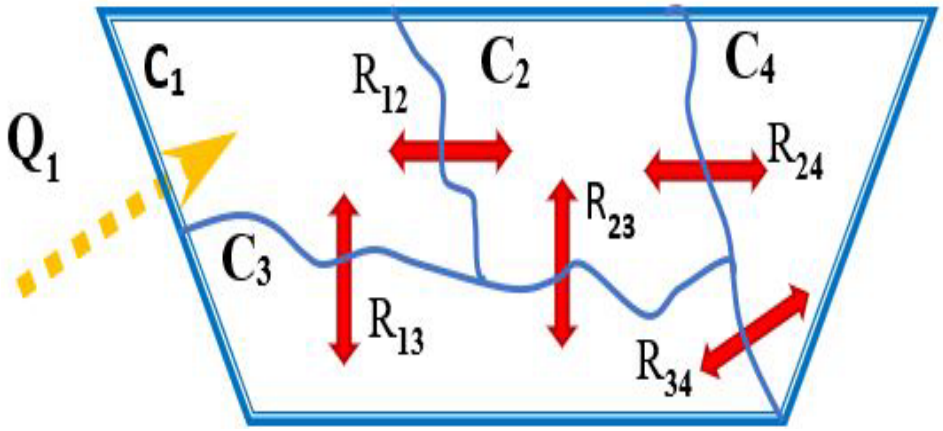

Figure 2.1. A system of 4 cells

For this system, the system of ODEs in matrix form can be written as

$$\frac{d}{dt}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} \frac{-1}{C_1R_{12}}+\frac{-1}{C_1R_{13}} & \frac{1}{C_1R_{12}} & \frac{1}{C_1R_{13}} & 0 \\ \frac{1}{C_2R_{12}} & \frac{-1}{C_2R_{12}}+\frac{-1}{C_2R_{23}}+\frac{-1}{C_2R_{24}} & \frac{1}{C_2R_{23}} & \frac{1}{C_2R_{24}} \\ \frac{1}{C_3R_{13}} & \frac{1}{C_3R_{23}} & \frac{-1}{C_3R_{13}}+\frac{-1}{C_3R_{23}}+\frac{-1}{C_3R_{34}} & \frac{1}{C_3R_{34}} \\ 0 & \frac{1}{C_4R_{24}} & \frac{1}{C_4R_{34}} & \frac{-1}{C_4R_{24}}+\frac{-1}{C_4R_{34}} \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} + \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{pmatrix}.$$

2.2. Some Explicit Methods

In this subsection we will review two novel methods and one unconventional method which are the constant-neighbour method CNe, the linear-neighbour method LNe and the unconditionally positive finite difference method UPFD. The constant-neighbour method CNe is a new explicit scheme that was introduced by Kovács and Gilicz [38] in 2018. Later, Kovács analysed this method mathematically in order to investigate its properties such as the stability and the convergence and verified that by numerical experiments. Also, he developed the linear-neighbour

method (LNe) which uses the CNe method as predictor. However, those methods depend on a new way of thinking [37], [39]. In 2013, Chen-Charpentier and Kojouharov introduced a new class of Finite Difference methods for advection–diffusion reaction equations that guarantees the positivity of the solutions, independent of the time step and mesh size [40]. Also, some conventional explicit methods will be discussed.

2.2.1. The Constant-Neighbour Method CNe

Considering Eq. (2.8), the constant-neighbour can be understood by the following steps:

1a) we make an assumption: when we calculate the new value of a variable $u_{\mathrm{i}}^{\mathrm{n+1}}$, we neglect that other variables (the neighbours) are also changing during the time step $h=\Delta t$. It means that we consider u_j a constant ($u_j^n = u_j^{n+1}$) if $\mathrm{j}\neq\mathrm{i}$. That assumption converts the coupled system of ODEs in Eq. (2.8) into uncoupled system of the form:

$$\frac{du_{\mathrm{i}}}{dt} = a_{\mathrm{i}} - \frac{u_{\mathrm{i}}}{\tau_{\mathrm{i}}}, \tag{2.11}$$

where we introduced

$$a_{\mathrm{i}} = \sum\nolimits_{\mathrm{j}\in n(\mathrm{i})} m_{\mathrm{ij}} u_{\mathrm{j}}^{\mathrm{n}} + Q_{\mathrm{i}}, \tag{2.12}$$

and the characteristic time or time-constant of the cell

$$\tau_{\mathrm{i}} = \frac{-1}{m_{\mathrm{ii}}} = \frac{C_{\mathrm{i}}}{\sum\nolimits_{\mathrm{j}\in n(\mathrm{i})} \frac{1}{R_{\mathrm{ij}}}}. \tag{2.13}$$

1b) it is straightforward that Eq. (2.11) can be solved analytically. The solution at the end of the time step can be used as predictor values:

$$u_{\mathrm{i}}^{\mathrm{n+1,pred}} = u_{\mathrm{i}}^{\mathrm{n}} \cdot e^{-\frac{h}{\tau_{\mathrm{i}}}} + a_{\mathrm{i}}\tau_{\mathrm{i}} \cdot \left(1 - e^{-\frac{h}{\tau_{\mathrm{i}}}}\right). \tag{2.14}$$

2.2.2. The Linear-Neighbour Method LNe

This method can also be achieved by two steps

2a) this step is the corrector, here we can make the assumption more realistic by assuming that the neighbouring values u_j of the actual cell u_i are changing linearly during the time step $h=\Delta t$. For cell of index i we introduce the effective slope

$$s_{\mathrm{i}} = \frac{a_{\mathrm{i}}^{\mathrm{pred}} - a_{\mathrm{i}}}{h}, \tag{2.15}$$

where $a_{\mathrm{i}}^{\mathrm{pred}} = \sum_{\mathrm{j}\in n(\mathrm{i})} m_{\mathrm{ij}} u_{\mathrm{j}}^{\mathrm{n+1,pred}} + Q_{\mathrm{i}}$ contains the predictor values which were introduced in Eq. (2.14). based on that approximation, we geta new uncoupled ODE system:

$$\frac{du_{\mathrm{i}}}{dt} = s_{\mathrm{i}} t + a_{\mathrm{i}} - \frac{u_{\mathrm{i}}}{\tau_{\mathrm{i}}}. \tag{2.16}$$

2b) Again Eq. (2.16) can be solved analytically and the solution at the end of the time step:

$$u_{\mathrm{i}}^{\mathrm{n+1}} = u_{\mathrm{i}}^{\mathrm{n}} e^{-\frac{h}{\tau_{\mathrm{i}}}} + \left(a_{\mathrm{i}}\tau_{\mathrm{i}} - s_{\mathrm{i}}\tau_{\mathrm{i}}^{2}\right)\left(1 - e^{-\frac{h}{\tau_{\mathrm{i}}}}\right) + s_{\mathrm{i}}\tau_{\mathrm{i}} h. \tag{2.17}$$

The scheme in Eq. (2.17) is called LNe2 and if we iterate the step (2b) we obtain the LNe3 method. To do that, we use the value of u_i^{n+1} calculated in Eq. (2.17) as a predictor, calculate the new value of $a_{\mathrm{i}}^{\mathrm{pred}}$ based on that corrector and then substitute it into Eq. (2.15) to get the new value for the effective slope. We substitute the new value of the effective slope into Eq. (2.16) to obtain uncoupled and linear system of ODEs which in turn can be solved analytically. If we go further with the iterations, we obtain LNe4, LNe5…etc.

2.2.3. Unconditionally Positivity Preserving Scheme

To illustrate this method, Chen-Charpentier and Kojouharov [40] simply considered the one-dimensional advection-diffusion equation with linear decay:

$$\frac{\partial u}{\partial t} + \mu\frac{\partial u}{\partial x} - D\frac{\partial^2 u}{\partial x^2} = -Ku, \quad (x,t) \in [0,\, x_{\max}] \times [0,\, t_{\max}], \tag{2.18}$$

for the unknown concentration function $u = u(x,t)$, with appropriate boundary and initial conditions and the parameters μ, D and K are positive constants.

We divide the space interval $[0,\, x_{\max}]$ into subdivisions $x_0 < x_1 <, ..., < x_N$. While $x_i = (i-1)\Delta x,\ i = 1, ..., N,\ \Delta x = \left(\frac{x_{\max}}{N-1}\right)$ and of course $x_N = x_{\max}$. We also divide the time interval $[0,\, t_{\max}]$ using equal time steps of size Δt. Let u_i^n be the approximation of the unknown concentration function at time step n and node i. The unconditionally positive finite difference (UPFD) scheme can be written:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \mu\frac{u_i^{n+1} - u_i^n}{\Delta x} - D\frac{u_{i+1}^n - 2u_i^{n+1} + u_{i-1}^n}{\Delta x^2} = -Ku_i^{n+1}.$$

The last formula can be written in explicit form

$$u_i^{n+1} = \frac{\hat{D}u_{i+1}^n + \frac{u_i^n}{\Delta t} + \left(\hat{\mu} + \hat{D}\right)u_{i-1}^n}{\frac{1}{\Delta t} + \hat{u} + 2\hat{D} + K}. \tag{2.20}$$

where $\hat{\mu} = \frac{\mu}{\Delta x}, \hat{D} = \frac{D}{\Delta x^2}$. This scheme can be applied to Eq. (2.1), in the absence of the advection term, where $\hat{\mu}$ will be zero.

2.2.4. The Odd-Even Hopscotch Structure OEH

The odd-even hopscotch algorithm firstly appeared in Gordon's work in 1965 [41]. Later in 1970, Gourlay analysed and reformulated that algorithm [42], [43]. It is designed to be a fast, general purpose algorithm which generates solutions with small expenditure of machine and human time [44]. To improve its accuracy, this fully explicit two-stage scheme has gone through modification and generalization processes, but always in the direction of implicitness, as far as we know. A variety of methods has been constructed starting with OEH and ending with ADI [42]. The OEH method has been applied to the Gray-Scott reaction-diffusion [45] and the Frank-Kamenetskii [46], the incompressible Navier-Stokes Equations [47], the Burgers' equation [48] and even to the Dirac-equation [49]. A vectorized version of OEH method has been implemented in order to solve two-dimensional Burgers' equation [50]. It has been found that the vectorization increased the efficiency of the solver. For solving two dimensional Burgers' equation, hybrid finite schemes were developed recently such as hopscotch-Crank-Nicolson-Du Fort-Frankel-Lax Friedrichs, hybrid hopscotch-Crank Nicolson-Lax Friedrichs and hopscotch Crank-Nicolson-Du Fort Frankel [51], [52].

Let us consider a spatially discretized system as that one in Eq. (2.8). In order to understand the structure of OEH, we illustrate the so-called bipartite grid. In that grid the set of nodes are divided into two subsets which are A (odd nodes, dark dots in Figure 2.2) and B (even nodes, light dots in Figure 2.2). The calculation of the values of the unknown variable u consisits of two stages. In the first stage, the values u of the subset A are calculated at time level $(n+1)$ using the only the values at time level (n). This process is depicted by thin green arrows in Figure 2.2. In the second stage, the values u of the subset B are calculated at time level $(n+1)$ using the values at time levels (n) and $(n+1)$ as well. This process is depicted by thick red arrow in Figure 2.2. In the time level $(n+1)$, we change the role of the subsets A and B. In other words, the values of of subset B are calculated in the first stage and then the values of subset A are claclulated in the second stage. The first stage in the orginal OEH uses the explicit Euler scheme, while the second stage uses the implicit Euler scheme.

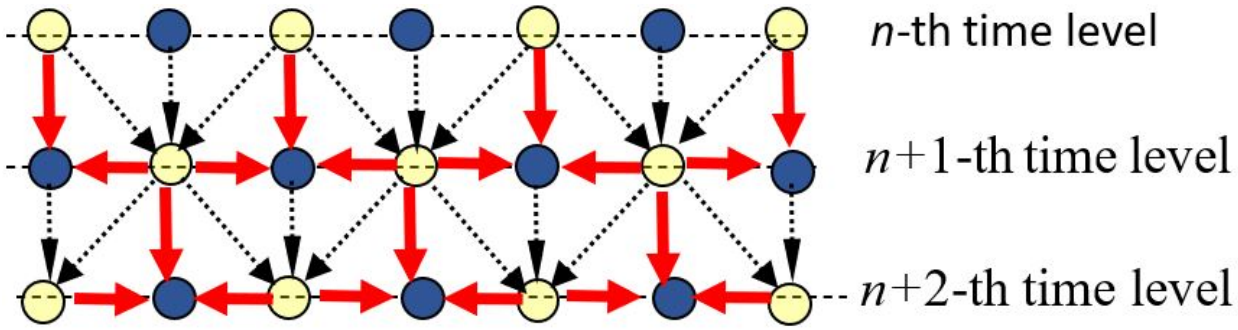

Figure 2.2. The stencil of the odd-even hopscotch algorithm. Thin black arrows (thick red arrows) indicate operations at Stage 1 (Stage 2).

3. NEW STABLE, EXPLICIT, SECOND ORDER HOPSCOTCH METHODS FOR DIFFUSION-TYPE PROBLEMS

In this chapter, I will illustrate novel numerical shcemes for solving the diffusion or heat equation. Let me give a brief chronology of our work. In the first phase [53], [54], we started by creating new numerical schmes. Some of these schemes were designed based on odd-even hopscotch structure. They were tested in the case of one dimensional systems. In the second phase [55], the schemes were investigated by solving more complicated systems, for example, two sapce dimensions and inhomogeneous media. The results inspired us to start the third phase [56]. So, we decided to systematically construct and test odd-even hopscotch-type numerical. Among the studied explicit two-stage methods some of them are unconditionally stable and their convergence rate in time step size is of the second order, which is analytically proved as well. The best methods are applied to the nonlinear Fisher's equation to illustrate that those methods work also for nonlinear equations. In order to examine the competitiveness of the new schemes, we test them, in case of heat equation, against widely used numerical solvers considering strongly inhomogeneous media and thus the coefficients strongly depend on space. The results show that the new schemes are significantly more effective than the widely used explicit or implicit methods, especially in case of extremely large stiff systems.

3.1. The Construction of the Methods

3.1.1. The Construction of the Used Formula for One Dimensional System

In case of one space dimensional equation (2.1) in the absence of the heat source, we discretize the space by creating nodes based on the usual rule $x_i = i\Delta x, i = 0,...,N$ as in Subsection 2.1. Using Eq. (2.6) in the absence of the heat source, we can discretize the time to get:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2}. \tag{3.1}$$

From the last formula we obtain the explicit Euler scheme:

$$u_i^{n+1} = u_i^n + r\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right), \tag{3.2}$$

where n refers to the time level and r is the mesh ratio:

$$r = \frac{\alpha h}{\Delta x^2} = -\frac{m_{ii} h}{2} > 0, \quad h = \Delta t, \quad 1 < i < N. \tag{3.3}$$

In the right-hand side of Eq. (3.2), the value of the unknown variables u are considered at time level (n). If we consider these variables at time level ($n+1$), we obtain the implicit Euler scheme:

$$u_i^{n+1} = u_i^n + r\left(u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}\right). \tag{3.4}$$

In the unconditionally positive finite difference scheme (Subsection 2.2.3), the right-hand side of Eq. (3.2) is treated differently. The values of the unknow variables u are taken at time level ($n+1$) only for the node of interest i, while considered at time level (n) for the other nodes. Now for our studied system, and after considering Eq. (2.19) we obtain:

$$u_i^{n+1} = u_i^n + r\left(u_{i-1}^n - 2u_i^{n+1} + u_{i+1}^n\right). \tag{3.5}$$

The right-hand side can be also treated as in the trapezoidal or Crank-Nicolson (CrN) method:

$$u_i^{n+1} = u_i^n + r\left(\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{2} + \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{2}\right). \tag{3.6}$$

That is known implicit formula. In the third term of the right-hand side of Eq. (3.6), we can take the neighbours at time level (n) to obtain the explicit version:

$$u_i^{n+1} = u_i^n + r\left(\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{2} + \frac{u_{i-1}^n - 2u_i^{n+1} + u_{i+1}^n}{2}\right). \tag{3.7}$$

In the second term of the right-hand side of Eq. (3.6), we can take the neighbours at time level ($n+1$) to obtain a new formula:

$$u_i^{n+1} = u_i^n + r\left(\frac{u_{i-1}^{n+1} - 2u_i^n + u_{i+1}^{n+1}}{2} + \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{2}\right). \tag{3.8}$$

The constant neighbour (CNe) method and the linear neighbour (LNe) methods were explained in Subsection 2.2 for a general system. In our simple one-dimensional homogeneous medium, Eq. (2.12) becomes:

$$a_i = \sum_{j \in \text{neighbours}(i)} m_{ij} u_j^n = \alpha \frac{u_{i-1}^n + u_{i+1}^n}{\Delta x^2} = \frac{r}{h}\left(u_{i-1}^n + u_{i+1}^n\right). \tag{3.9}$$

We emphasize here that the value of a_i do not change during the time step. The CNe scheme in Eq. (2.14) becomes:

$$u_i^{n+1} = u_i^n \cdot e^{m_{ii}h} - \frac{a_i}{m_{ii}}\left(1 - e^{m_{ii}h}\right). \tag{3.10}$$

In the LNe method, we must have a predicted values for the unknown variables u_i^{pred} that are valid at the end of time step. Those predicted values were obtained by the CNe method as explained in Subsection 2.2.2. In this chapter we will see that the predicted values can be obtained by other schemes such as explicit Euler. After we obtain the predicted values of the unknown varaible u_i^{pred} (by some method), we can calculate the effective slope as we did in Eq. (2.15):

$$s_i = \frac{a_i^{\text{pred}} - a_i}{h}. \tag{3.11}$$

The value of a_i is the same as in Eq. (3.9) and

$$\begin{aligned} a_i^{\text{pred}} &= \sum_{j \in \text{neighbours}(i)} m_{ij} u_j^{\text{n+1,pred}} = \alpha \frac{u_{i-1}^{\text{n+1,pred}} + u_{i+1}^{\text{n+1,pred}}}{\Delta x^2} \\ &= \frac{r}{h}\left(u_{i-1}^{\text{n+1,pred}} + u_{i+1}^{\text{n+1,pred}}\right). \end{aligned} \tag{3.12}$$

Now we have all the necessary information and after considering the definition of r, the LNe scheme in Eq. (2.17) can be written as follows:

$$u_i^{\text{n+1}} = u_i^{\text{n}} e^{-2r} + a_i \frac{h}{2r}\left(1 - e^{-2r}\right) + s_i \frac{h^2}{2r}\left(1 - \frac{1 - e^{-2r}}{2r}\right). \tag{3.13}$$

The formula in Eq. (3.11) can be finalized as follows:

$$s_i = \frac{r}{h^2}\left(u_{i-1}^{\text{n+1}} + u_{i+1}^{\text{n+1}} - u_{i-1}^{\text{n}} - u_{i+1}^{\text{n}}\right), \tag{3.14}$$

and

$$\begin{aligned} u_i^{\text{n+1}} &= u_i^{\text{n}} e^{-2r} + \frac{1}{2}\left(u_{i-1}^{\text{n}} + u_{i+1}^{\text{n}}\right)\left(1 - e^{-2r}\right) \\ &+ \frac{1}{2}\left(u_{i-1}^{\text{n+1,pred}} + u_{i+1}^{\text{n+1,pred}} - u_{i-1}^{\text{n}} - u_{i+1}^{\text{n}}\right)\left(1 - \frac{1 - e^{-2r}}{2r}\right). \end{aligned} \tag{3.15}$$

3.1.2. A Family Of New Odd-Even Hopscotch Algorithms

In Subsection 2.2.4, I illustrated that the original odd-even Hopscotch method consists of two stages. The first stage in the orginal OEH uses the explicit Euler scheme, while the second satge uses the implicit Euler scheme. Our objective in this chapter is to systematically examine all the possible combinations of the previously specified formulas. Thus, in the first stage, we can use those explicit formulas or those that can be made explicit. We use Eqs. (3.2), (3.5), (3.7), (3.10) respectively. After expressing explicitly the new values of u_i^{n+1}, the formulas of the first stage of hopscotch structure can written.

Satge 1 formulas

A) Explicit Euler

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

B) UPFD

$$u_i^{n+1} = \frac{u_i^n + r\left(u_{i-1}^n + u_{i+1}^n\right)}{1+2r}. \tag{3.17}$$

C) Explicit-neighbour "Crank-Nicolson"

$$u_i^{n+1} = \frac{(1-r)u_i^n + r\left(u_{i-1}^n + u_{i+1}^n\right)}{1+r}. \tag{3.18}$$

D) CNe, constant neighbour

$$u_i^{n+1} = u_i^n e^{-2r} + \frac{u_{i-1}^n + u_{i+1}^n}{2}\left(1-e^{-2r}\right). \tag{3.19}$$

In the second stage, we start with Eqs. (3.2), (3.5), (3.6), (3.8), (3.10) and (3.15) respectively. If the values of the neighbours u_{i+1}^{n+1} and u_{i-1}^{n+1} are not used in the second stage, then the hopscotch structure makes no sense. Therefore, in the second stage, the explicit Euler and CNe formulas will be modified. Since the the values of the neighbours are already calculated in the first stage, the UPFD scheme in Eq. (3.5) coincides with the implicit Euler scheme in Eq. (3.4). Now we can write the formulas of the second stage.

Satge 2 formulas

1. Explicit Euler

$$u_i^{n+1} = (1-2r)u_i^n + r\left(u_{i-1}^{n+1} + u_{i+1}^{n+1}\right). \tag{3.20}$$

2. UPFD (Implicit Euler)

$$u_i^{n+1} = \frac{u_i^n + r\left(u_{i-1}^{n+1} + u_{i+1}^{n+1}\right)}{1+2r}. \tag{3.21}$$

3. Crank-Nicolson

$$u_i^{n+1} = \frac{r}{1+r}\left(\frac{u_{i-1}^{n+1} + u_{i+1}^{n+1}}{2} + \frac{u_{i-1}^n + u_{i+1}^n}{2}\right) + \frac{1-r}{1+r}u_i^n. \tag{3.22}$$

4. Implicit-Neighbour Crank-Nicolson

$$u_i^{n+1} = \frac{(1-r)u_i^n + r\left(u_{i-1}^{n+1} + u_{i+1}^{n+1}\right)}{1+r}. \tag{3.23}$$

$$u_i^{n+1} = u_i^n e^{-2r} + \frac{u_{i-1}^{n+1} + u_{i+1}^{n+1}}{2}\left(1-e^{-2r}\right). \tag{3.24}$$

6. Linear Neighbour LNe

$$u_i^{n+1} = u_i^n e^{-2r} + \frac{1}{2}\left(u_{i-1}^n + u_{i+1}^n\right)\left(\frac{1-e^{-2r}}{2r} - e^{-2r}\right) + \frac{1}{2}\left(u_{i-1}^{n+1} + u_{i+1}^{n+1}\right)\left(1 - \frac{1-e^{-2r}}{2r}\right). \quad (3.25)$$

Each formula from the first stage can be combined with six possible formulas from the second stage in hopscotch structure resulting in $4 \times 6 = 24$ possible methods. For instance, the methods denoted by A2 refers to the original well-know OEH, while the method denoted by A6 means that the explicit Euler formula is used in the first stage and the linear-neighbour formula is used in the second stage.

3.1.3. The Construction Of the Methods for General 2D System

In Subsection 2.1, I derived the formula of spatially discretized heat conduction equation in case of general system. For a general system such that in Eq. (2.3), in the absence of the heat source, Eq. (2.8) can be written as follows

$$\frac{d\vec{u}}{dt} = M\vec{u}. \quad (3.26)$$

Recall that the previous equation is independent of any coordinate system. For the sake of simplicity, we introduce the following notations.

$$r_i = \frac{h}{\tau_i}, \quad A_i = h\sum_{j\neq i} m_{ij} u_j^n = h \sum_{j\in neighbours(i)} \frac{u_j^n}{C_i R_{ij}}, \quad (3.27)$$

and

$$A_i^{new} = h\sum_{j\neq i} m_{ij} u_j^{n+1} = h \sum_{j\in neighbours(i)} \frac{u_j^{n+1}}{C_i R_{ij}}. \quad (3.28)$$

Now we are ready to introduce the formulas taht can be applied in the first and second stages for general system.

Stage 1 formulas

A) Explicit Euler

$$u_i^{n+1} = \left(1 - r_i\right)u_i^n + A_i. \quad (3.29)$$

B) UPFD

$$u_i^{n+1} = \frac{u_i^n + A_i}{1 + r_i}. \quad (3.30)$$

C) Explicit-neighbour “Crank-Nicolson”

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

D) CNe, constant neighbour

$$u_i^{n+1} = u_i^n \cdot e^{-r_i} + \frac{A_i}{r_i}\left(1 - e^{-r_i}\right). \tag{3.32}$$

Stage 2 formulas

1. Explicit Euler

$$u_i^{n+1} = \left(1 - r_i\right)u_i^n + A_i^{new}. \tag{3.33}$$

2. UPFD (Implicit Euler)

$$u_i^{n+1} = \frac{u_i^n + A_i^{new}}{1 + r_i}. \tag{3.34}$$

3. Crank-Nicholson

$$u_i^{n+1} = \frac{A_i + A_i^{new} + \left(2 - r_i\right)u_i^n}{2 + r_i}. \tag{3.35}$$

4. Implicit-neighbour "Crank-Nicholson"

$$u_i^{n+1} = \frac{2A_i^{new} + \left(2 - r_i\right)u_i^n}{2 + r_i}. \tag{3.36}$$

5. CNe, constant neighbour

$$u_i^{n+1} = u_i^n \cdot e^{-r_i} + \frac{A_i^{new}}{r_i}\left(1 - e^{-r_i}\right). \tag{3.37}$$

6. LNe, Linear neighbour

$$u_i^{n+1} = u_i^n e^{-r_i} + \left(A_i - \frac{A_i^{new} - A_i}{r_i}\right)\frac{1 - e^{-r_i}}{r_i} + \frac{A_i^{new} - A_i}{r_i}. \tag{3.38}$$

By implementing these formulas in the hopscotch structure, one can obtain the generalized versions of the 24 combinations which were mentioned in the previous subsection

3.2. Numerical Experiments and Results

The numerical solution is compared with the reference solution, or the benchmark solution, only at end of the time interval t_{fin}. This time interval will be defined later for each experiment. The accuracy is measured by using the global L_∞ error, which is the maximum of the absolute difference between numerical solution u_j^{num} (calculated by one of the previously introduced numerical methods at the end of the time interval) and the reference solution u_j^{ref}, which can be either the analytical or very accurate numerical solution.

$$\text{Error}(L_\infty) = \max_{0 \le j \le N} \left| u_j^{\text{ref}}(t_{\text{fin}}) - u_j^{\text{num}}(t_{\text{fin}}) \right|. \tag{3.39}$$

Also I will introduce two other error norms which will be used for general grid. The first error norm is the L_1, that can be defined as the average error

$$\text{Error}(L_1) = \frac{1}{N} \sum_{0 \le j \le N} \left| u_j^{\text{ref}}(t_{\text{fin}}) - u_j^{\text{num}}(t_{\text{fin}}) \right|. \tag{3.40}$$

The second error norm is same difference, but weighted with the capacities of the cells

$$\text{Error}(Energy) = \frac{1}{N} \sum_{0 \le j \le N} C_j \left| u_j^{\text{ref}}(t_{\text{fin}}) - u_j^{\text{num}}(t_{\text{fin}}) \right|. \tag{3.41}$$

The last equation gives the error in terms of energy in case of heat equation. It takes into consideration that a temperature deviation in a big cell has more crucial effect in practice than in a tiny cell.

If deal thermally isolated system, the matrix M has one zero eigenvalue (due to the fact that the total heat would be constant without heat sources), while all the other eigenvalues are necessarily negative. If we denote the eigenvalues of the matrix M with the smallest (nonzero) and largest absolute values by λ_{MIN} and λ_{MAX}, respectively, then one can define and calculate the stiffness ratio as follows

$$SR = \lambda_{\text{MAX}} / \lambda_{\text{MIN}}. \tag{3.42}$$

We are also going to use another important quantity to characterize the level difficulty of the problem, which is the maximum possible time step size for the explicit Euler scheme FTCS that guarantees the stability. That value can be calculated as follows

$$h_{\text{MAX}}^{\text{FTCS}} = \left| 2 / \lambda_{\text{MAX}} \right|. \tag{3.43}$$

The simulations are conducted using the MATLAB R2020b software on a desktop computer with an Intel Core i5-9400 as CPU, 16.0 GB RAM. The running time is measured by the built-in tic-toc function of MATLAB.

3.2.1. Tests for the First Evaluation

We examine two space-dimensional rectangle-type lattices of $N = N_{\text{X}} \times N_{\text{Z}}$ cells with zero Neumann boundary conditions, i.e. thermal isolation. Several numerical experiments have been conducted in order to test the behaviour of the 24 methods. The methods have been evaluated based on three crucial properties which are the stability, positivity and the order of convergence. We solve the system in Eq. (3.26) where the values of the heat capacities and the resistances are defined using the following formulas

$$C_i = 10^{(\alpha_C - \beta_C \times rand)},\ R_{x,i} = 10^{(\alpha_{Rx} - \beta_{Rx} \times rand)},\ R_{z,i} = 10^{(\alpha_{Rz} - \beta_{Rz} \times rand)}, \qquad (3.44)$$

where rand is a built-in function in MATLAB which generates uniformly distirbuted-random numbers in the interval $[0,1]$. It means that Eq. (3.44) generates values with a log-uniform distribution. The values of the exponents $\alpha_C, \beta_C, \alpha_{Rx}, \beta_{Rx}, \alpha_{Rz}, \beta_{Rz}$ have been chosen to generate systems with very different $h_{\mathrm{MAX}}^{\mathrm{FTCS}}$ and stiffness ratio. For instance, $\alpha_C = 1, 2,$ or 3, $\beta_C = 2, 4,$ or 6. The spatial dimensions of the systems have been 5×3 and 30×30. The intial conditions have been generated using the ranomly $u_i(0) = rand$. The end of the time interval has been set to $0.1, 1$ and 10. To calculate the reference solution, the implicit ode15s solver has been used while imposing a strict error tolerance ('RelTol' and 'AbsTol' were both 10^{-12}). Here we consider that an algorithm is stable if we never observe that the solution is explodes, i.e. if the error does not grow indefinitely as t_{fin} grows regardless of the time step size. However, the stability will be rigorously examined later. Due to the second law of thermodynamics, the solution should follow the Maximum and Minimum principles [57, p. 87]. We consider that the method is positivity preserving if it never violates this principle, i.e. in this case no value of u exceeds the $[0,1]$ interval, which has been examined at the end of each time step. This property obviously implies the stability. If a method is stable but not positivity preserving, then unphysical oscillations can arise with larger amplitude than the function value, but these are finally stabilized at a finite level. The original OEH method typically behaves like this. The results for the 24 OEH combinations are summarised in Table 3.1. I emphasize that at this point those properties are evaluated only through numerical experiments without any analytical investigation. Based on these intensive tests I chose algorithms A2, B1, C4, C5, D4 and D5 for further numerical and analytical investgation.

	1) Exp Eu	2) UPFD	3) CrN	4) IN CrN	5) CNe	6) LNe
A) Exp Eu	**U**	**S 2**	**U**	**U**	**S**	**S**
B) UPFD	**S 2**	**P**	**S**	**S**	**P**	**S**
C) EN CrN	**U**	**S**	**S**	**S 2**	**S 2**	**S**
D) CNe	**S**	**P**	**S**	**S 2**	**P 2**	**P**

Table 3.1. Properties of the 24 algorithms. The letter U, S and P mean unstable, stable, and positivity preserving, respectively. The number 2 means that the algorithm is second order (all other schemes are first order).

3.2.2. Solution of the Nonlinear Fisher's Equation

I will treat the so-called Fisher-KPP equation [58] which is nonlinear reaction-diffusion equations of the following form

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + \beta u\left(1-u\right). \tag{3.45}$$

Originally this equation was introduced to describe how advantageous gene-variants spread in space and time. In this case β>0 is the coefficient of reaction (proliferation rate in the original equation). Since then it has been applied to model other phenomena such as propagation of fronts in autocatalytic chemical reactions [59] and combustion processes [60]. In this experiment, we solve Eq. (3.45) with with $\alpha = 1$, subject to the following initial condition:

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

The analytical solution of this problem is the following travelling wave function [61], [62]:

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

The appropriate Dirichlet boundary conditions are prescribed at the two ends of the interval:

$$u(x = x_0, t) = \left(1 + e^{\sqrt{\frac{\beta}{6}}x_0 - \frac{5}{6}\beta t}\right)^{-2}, \quad u(x = x_{\text{fin}}, t) = \left(1 + e^{\sqrt{\frac{\beta}{6}}x_{\text{fin}} - \frac{5}{6}\beta t}\right)^{-2}.$$

The values of the exact solution obviously lie in the unit interval, $u\left(x,t\right) \in \left[0,1\right]$, $x,t \in \mathbb{R}$. I note that it is hard to keep this property for the numerical solution in case of larger time step sizes.

First, I have tried the simplest and most straightforward way to incorporate the nonlinear reaction term to the method: I added the extra term $\beta u_i^{\text{n}}\left(1-u_i^{\text{n}}\right)h$ at the end of each stage to the value of the variable $u_i^{\text{n+1}}$. However, the obtained results were not promising at all, the errors were rather large. Then I arranged this addition in a separate third stage, where a loop is going through all the nodes with the following operation:

$$u_i^{\text{n+1}} = u_i^{\text{n+1,pred}} + \beta u_i^{\text{n+1,pred}}\left(1-u_i^{\text{n+1,pred}}\right)h,$$

where $u_i^{\text{n+1,pred}}$ is the result of the first and the second stage using the same formulas as in Subsection 3.1.1, i.e. with taking into account the effect of the diffusion term only. With this modification, I obtained much better results. In many numerical experiments, i.e., for several values of x_{fin}, t_{fin}, and β. Unfortunately, in some other cases the nonlinear term made the algorithms, especially the A2 and B1, unstable. The instability appeared usually for large values of β and h and manifested itself when the final time t_{fin} was large. I attempted to solve this

instability problem by treating the nonlinear term in a semi-implicit way according to the following arrangement:

$$u_i^{n+1} = u_i^{n+1,pred} + \beta u_i^{n+1,pred}\left(1 - u_i^{n+1}\right)h .$$

Note that now the new value u_i^{n+1} is present in the bracket on the right-hand side. This equation can be arranged into a fully explicit form:

$$u_i^{n+1} = \frac{1+\beta h}{1+\beta h u_i^{n+1,pred}} u_i^{n+1,pred}. \qquad (3.46)$$

Remark 1 According to formula (3.46), the new value exceeds one, i.e. $u_i^{n+1} > 1$ if and only if

$$\left(1+\beta h\right)u_i^{n+1,pred} > 1 + \beta h u_i^{n+1,pred} \quad \Leftrightarrow \quad u_i^{n+1,pred} > 1.$$

This means that if the original values u_i^n were majorated by 1 and if the A2, …, D5 algorithms did not terminate this property during the first two stages, then the final result $u_i^{n+1,pred}$ will be still majorated by 1.

I performed several tests with the algorithms using this nonconventional semi-implicit treatment of the nonlinear reaction term and obtained many curves. I found that the methods were always stable, even if the $u_i^{n+1,pred}$ values were above 1. Especially A2 (the original OEH) and, usually in a much less extent, B1 (the reversed OEH) can produce $u_i^{n+1,pred}$ values above 1, but these potentially unstable elevations never grow unboundedly and usually diffuse away. However, this diffusion can be very slow in the case of the original OEH method. From this point of view, the original OEH is the weakest, which will be illustrated through the following concrete numerical experiment. We fix the space interval to $x \in [0,2]$, which is discretized by dividing it into 200 equal parts: $x_j = x_0 + j\Delta x$, $j = 0,...,200$, $\Delta x = 0.01$ where $x_0 = 0$. We set $\beta = 8$ and $t_{fin} = 2$. The errors as a function of the time step size *h* are presented in Figure 3.1. I have tried to perform these calculations in case of a non-equidistant mesh. The original OEH teeters on the brink of instability for large values of *h* but never actually becomes unstable, even when β is as large as 10^8.

I emphasize that all examined methods have been convergent in all numerical experiments, so for small h they produced small errors. Still, I do not state that I have found the optimal way to incorporate the nonlinear term into our formulas, nor that these algorithms are the best to solve Fisher's equation. This small subsection is only to demonstrate that these hopscotch-type methods can be successfully applied to solve nonlinear equations as well.

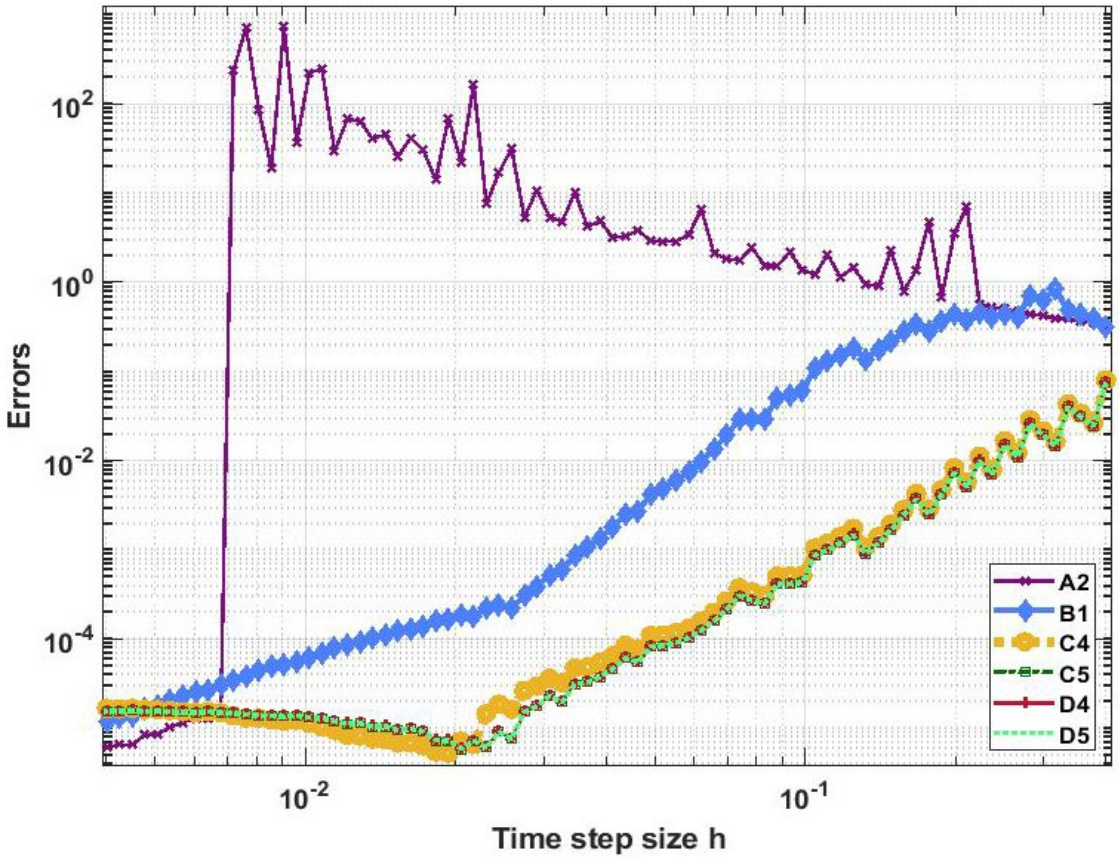

Figure 3.1. L_∞ errors as a function of the time step size for Fisher's equation with $\beta = 8$.

3.2.3. Comparison with Other Numerical Solvers, First Case

We examine two space-dimensional rectangle-type lattices of $N_x = 100, N_z = 100$, which means that we have $N = 10000$ cells. The system is subjected to zero Neumann boundary conditions. The value of the final time t_{fin} has been chosen to be 0.1. To set the the values of the capacities and resistances in Eq. (3.44), the exponents were chosen to be:

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

It means that the capacities have a log-uniform distribution in the interval $[0.01, 100]$. The stifness ratio of this system, calculated by Eq. (3.42), is $SR = 3.1 \times 10^8$. The maximum time-step, calculated by Eq. (3.43), is $h_{MAX}^{FTCS} = 7.3 \times 10^{-4}$. We compare the performance of the designed methods and that professionally coded and extensively tested MATLAB solvers, which are the followings:

- ode15s, variable-step, variable-order (VSVO) solver based on the (implicit) numerical differentiation formulas (NDFs) of orders 1 to 5, where the letter s denotes that the code has been developed for stiff problems
- ode23s, a modified (implicit) Rosenbrock formula of order 2

- ode23t, an implementation of the (implicit) trapezoidal rule using a "free" interpolant
- ode23tb, a combination of the (implicit) trapezoidal rule and a backward differentiation formula
- ode23, the explicit Runge-Kutta-Bogacki-Shampine method of order 2(3)
- ode45, the explicit Runge-Kutta-Dormand-Prince formula of order 4(5)
- ode113, a VSVO Adams-Bashforth-Moulton solver of orders 1 to 13

In Figure 3.2 and Figure 3.3, I present the error produced by the schemes as a function of the time-step size and the total running time respectively. For all MATLAB solvers, the tolerance have started from 'RelTol'= 'AbsTol' $\doteq$ tol=1000, then it has been decreased by a factor of ten until it reached 10^{-5}. In Table 3.2, I present some results obtained by our schemes and Matlab slovers. One can see that the new schemes (as well as the original OEH) are much faster than the conventional explicit or implicit solvers, considering the sa,e level of accuracy.

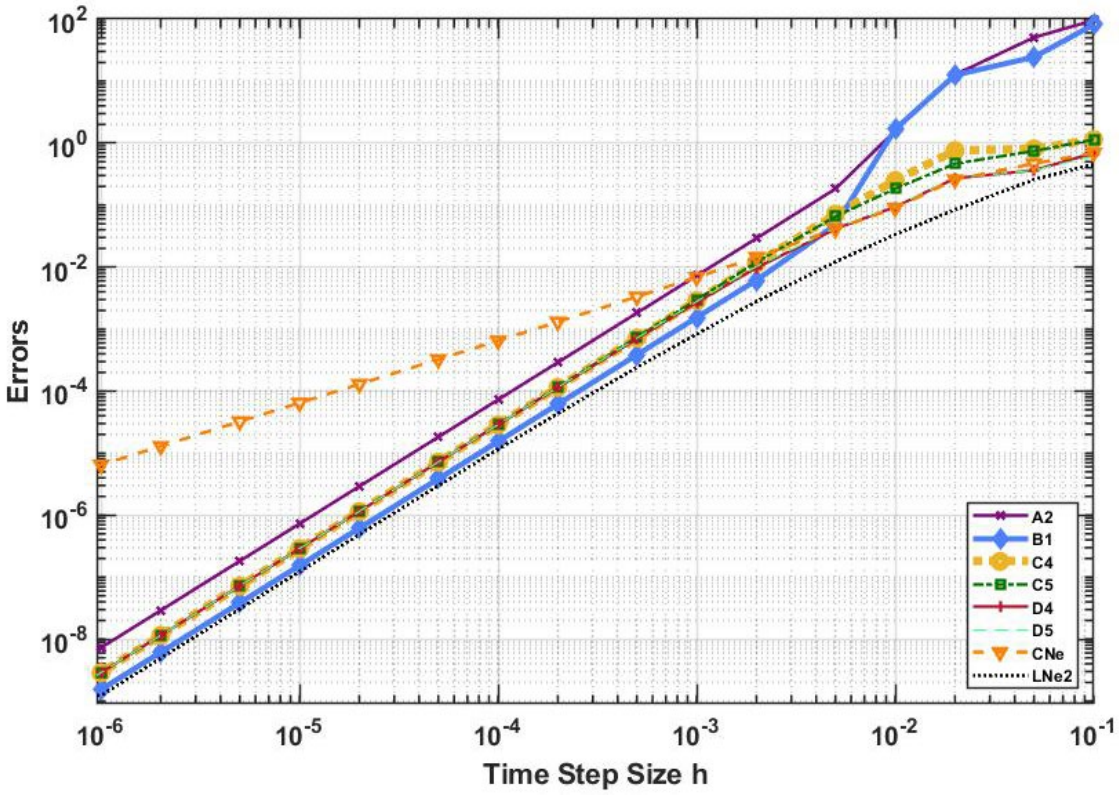

Figure 3.2. L_∞ errors as a function of the time step size for the first (less stiff) system of 10000 cells in case of original CNe, the two-stage LNe and the OEH algorithms.

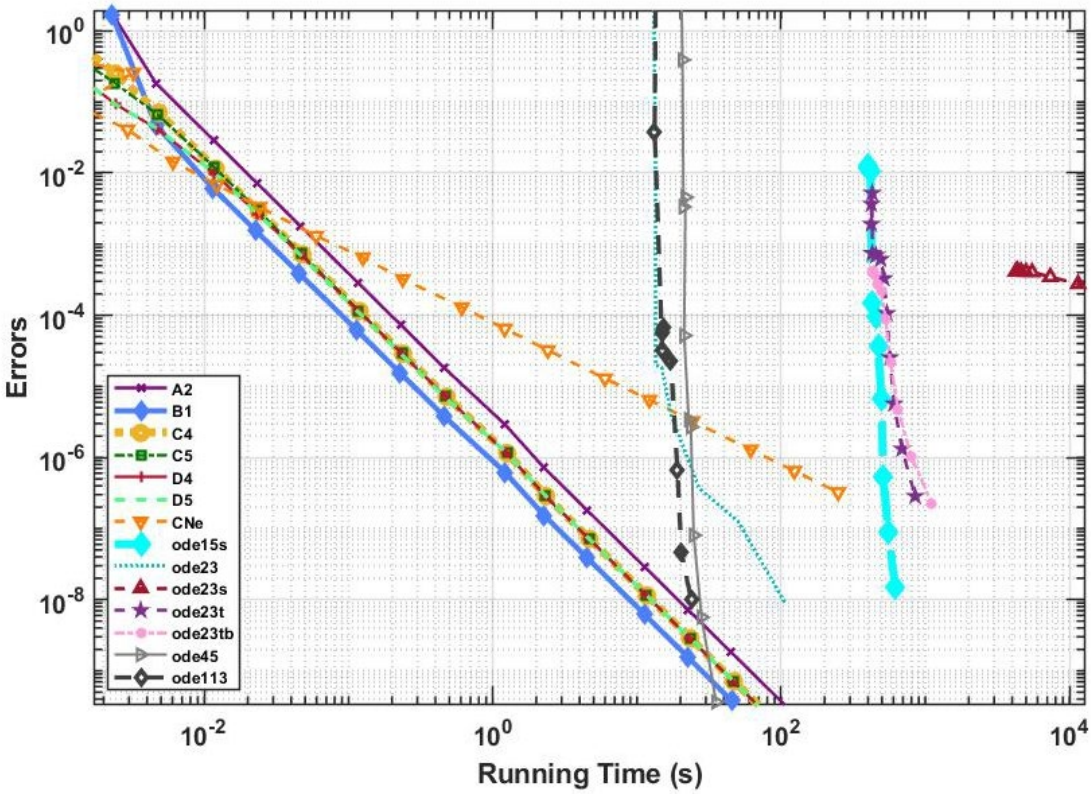

Figure 3.3. L_∞ errors as a function of the total running times for the first system in case of OEH type methods and seven different MATLAB solvers.

Numerical method	Running time	L_∞	L_1	Energy
ode15s, tol=10^{3}	397	1.3×10^{-}	1.1×1	5.62
ode23s, tol=10^{3}	4346	4.2×10^{-}	3.0×1	0.15
ode23t, tol=10^{-8}	849	2.9×10^{-}	2.0×1	1.0×10^{-4}
ode23tb, tol=100	428	4.1×10^{-}	2.9×1	0.14
ode45, tol=0.1	21.0	3.3×10^{-}	6.5×1	2.7×10^{-3}
ode23, tol=10^{-6}	27.0	3.7×10^{-}	9.6×1	4.8×10^{-5}
ode113, tol=10^{-6}	19.1	6.7×10^{-}	4.2×1	1.9×10^{-6}
A2, h=10^{-3}	0.023	7.2×10^{-}	1.1×1	1.2
B1, h=10^{-3}	0.023	1.5×10^{-}	1.6×1	2.3×10^{-2}
B1 , h=10^{-5}	2.26	1.5×10^{-}	1.6×1	2.5×10^{-6}
D5, h=10^{-3}	0.023	2.8×10^{-}	6.8×1	0.65

TABLE 3.2. Performance of different OEH and MATLAB solvers for the first system of 10000 cells.

3.2.4. Comparison with Other Numerical Solvers, Second Case

We use the same set up of the previous experiment but we change only the following

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

which means that we increase the range of the distribution of the resistances and capacities . The geometric mean of resistances in the x direction is 10 while in the z direction it is only 0.1, which means that the system is anisotropic in space. Based on that the stiffness ratio is much higher than the previous experiment $SR = 2.4\times10^{12}$ and $h_{\mathrm{MAX}}^{\mathrm{EE}} = 1.6\times10^{-6}$. The energy-error and the L_∞ errors are presented as a function of total running time in Figure 3.4 and Figure 3.5. I note that the new schemes are two or three orders of magnitude faster than the conventional explicit and implicit methods. I emphasize that none of the MATLAB solvers are able to provide any realistic (non-divergent) results in 300s, while using the OEH algorithms one can get quite accurate results in a few tenths of a second. Table 3.3 shows some results which were obtained by MATALB solvers and the OEH schemes.

We can conclude that B1 scheme (reversed UPFD-EE Hopscotch) is the most efficient scheme to solve these type of problems. The D4, D5, C4 and C5 schemes approximately produced the same errors at the same running time. However, the D5 scheme has a unique property (positivity preserving) , which will be discussed in another section, along with other properties for B1 scheme.

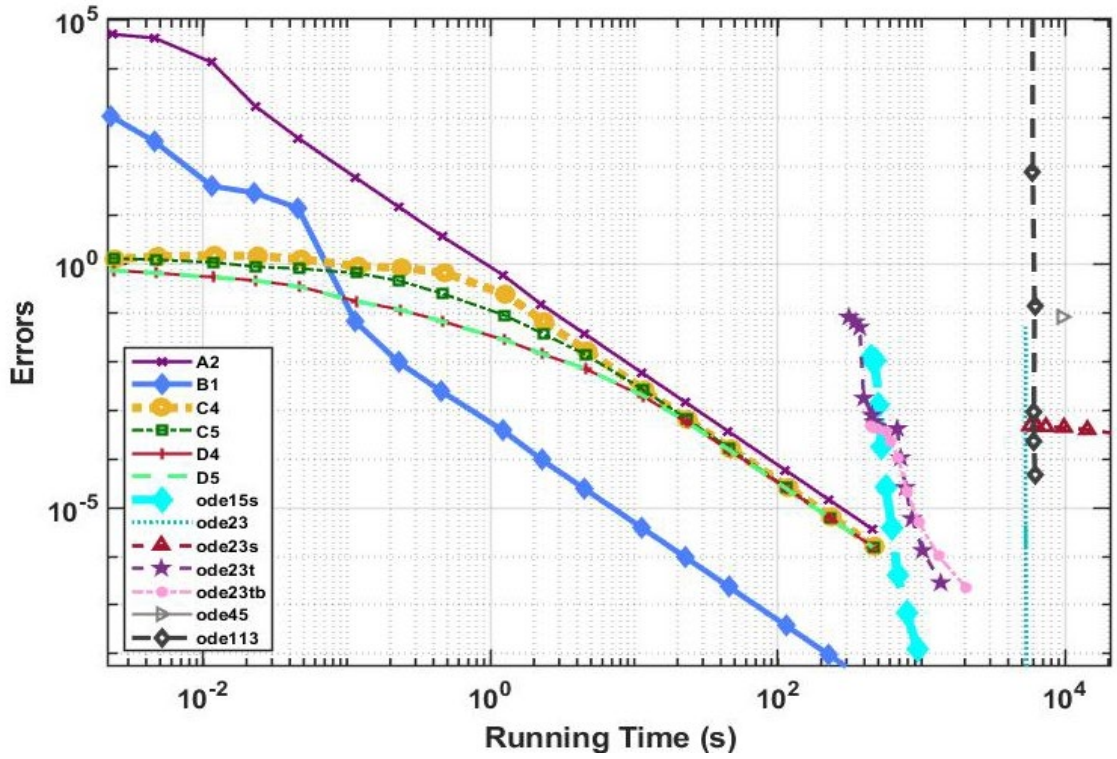

Figure 3.4. L_∞ errors as a function of the total running times for the second (very stiff) system in case of the original and the new hopscotch algorithms, and seven different MATLAB solvers.

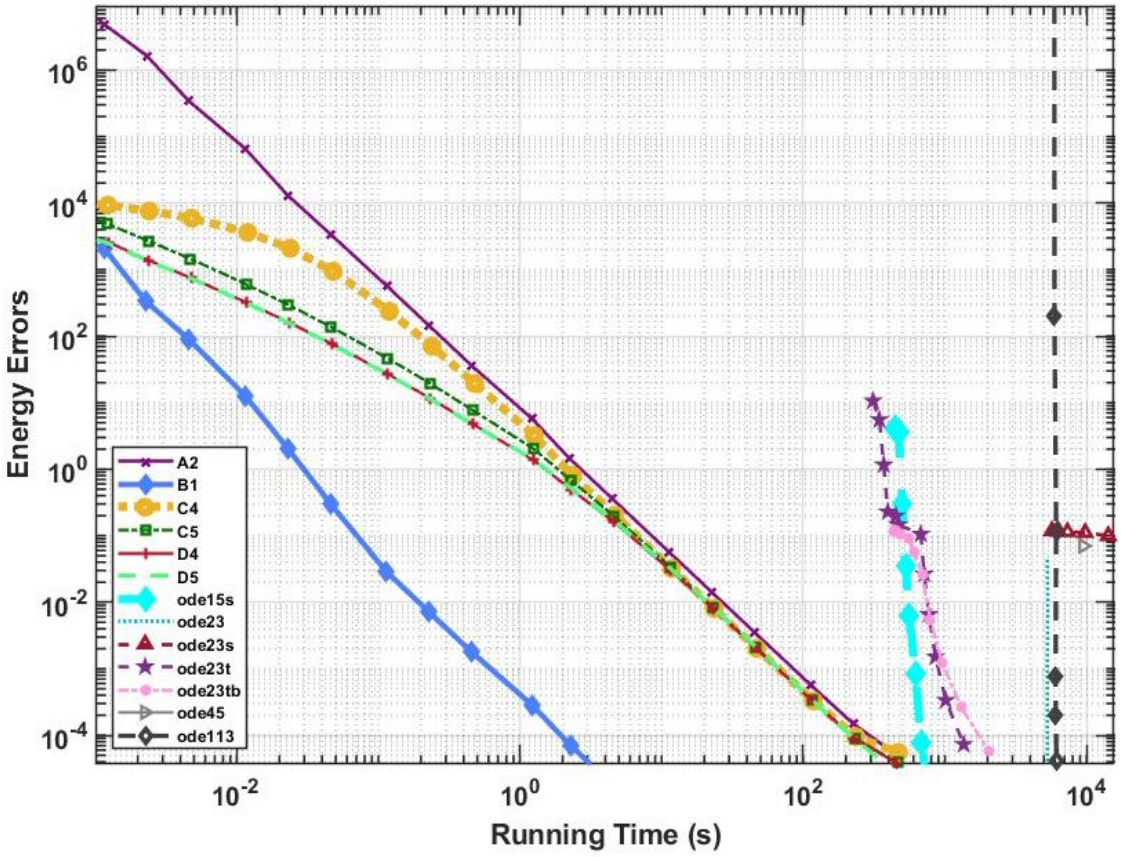

Figure 3.5. Energy errors as a function of the total running times for the second system in case of the hopscotch algorithms, and seven different MATLAB solvers.

Numerical method	Running time	L_∞	L_1	Energy
ode15s, tol=10^{-6}	680	4.1×10^{-}	1.5×1	7.5×10^{-5}
ode23s, tol=1000	5694	4.7×10^{-}	2.4×1	0.12
ode23t, tol=1000	310	8.1×10^{-}	2.1×1	10.6
ode23tb, tol=10^{-8}	2037	2.3×10^{-}	1.2×1	5.8×10^{-5}
ode45, tol=1	9480	8.1×10^{-}	1.5×1	7.0×10^{-2}
ode23, tol=10^{-6}	5317	1.2×10^{-}	2.3×1	1.1×10^{-6}
ode113, tol=10^{-2}	6046	8.9×10^{-}	1.7×1	7.7×10^{-4}
A2, h=10^{-4}	0.23	1.47	1.0×1	142
B1, h=10^{-4}	0.23	9.6×10^{-}	2.1×1	7.1×10^{-3}
B1 , h=10^{-6}	22.6	9.6×10^{-}	2.1×1	1.5×10^{-6}
D5, h=10^{-4}	0.23	1.2×10^{-}	8.4×1	11.7
D5, h=10^{-6}	22.9	5.8×10^{-}	6.0×1	8.5×10^{-3}

TABLE 3.3. Performance of different solvers for the second system of 10000 cells.

3.3. The Analytical Investigation of the Proposed Methods

3.3.1. Stability

First, I deal with the D5 scheme. In our previous work [37] we showed that in case of CNe method, the new value of the temperature u_i^{n+1} of any cell is the weighted average of the cell of interest and its neighbour at the beginning of the time step. In Theorem 1, I will elaborate the previous statement to the D5 algorithm. Let me first evoke a simple lemma which I will use intensively in the proof of the theorems [63, p. 28].

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

$$x = \sum\sum \left(a_i b_{ij} \right) y_{ij},$$

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

Theorem 1. If method D5 is applied to Eq. (1.16) when $q = 0$, the new values u_{i}^{n+1} are he convex combination of the old values u_{j}^n, $j = 1,...,N$.

Proof. We will use the facts that $\tau_{\mathrm{i}} = -1/m_{\mathrm{ii}}$ and $m_{\mathrm{ij,j\neq i}}$ are non-negative quantities, thus $0 < \mathrm{e}^{-h/\tau_{\mathrm{i}}} = \mathrm{e}^{m_{\mathrm{ii}}h} \leq 1$ holds because of physical reasons, for example the Second law of thermodynamics. Let us start with the first stage:

$$u_i^{n+1} = u_i^n \cdot \exp\left(-\frac{h}{\tau_i} \right) + a_i \tau_i \cdot \left(1 - \exp\left(-\frac{h}{\tau_i} \right) \right), \tag{3.47}$$

the term $a_i \tau_i$ can be written as follows

$$a_i \tau_i = \frac{-1}{m_{ii}} \sum\nolimits_{\mathrm{j}\neq i} m_{ij} u_j^n. \tag{3.48}$$

Now we will calculate the coefficients of the u_j^n, $j \neq i$

$$\frac{-m_{ij}}{m_{ii}} = \frac{1}{C_i R_{ij}} \frac{1}{\sum_{k\neq i} \frac{1}{C_i R_{ik}}} = \frac{1/R_{ij}}{\sum_{k\neq i} 1/R_{ik}}. \tag{3.49}$$

It can be easily seen that these coefficients are nonnegative, and their sum is one, which means that $a_i \tau_i$ is a convex combination of the old values u_j^n. Now it is obvious that the coefficients $\exp\left(-h/\tau_i\right)$ and $\left(1 - \exp\left(-h/\tau_i\right)\right)$ in Eq. (3.47) are also nonnegative and their sum is 1, therefore, according to the lemma, u_i^{n+1} is also convex combination of the values u_j^n, $j = 1,...,N$.

In the second stage, a similar argument holds, except that on the right hand side of Eq. (3.48) the onlx calculated values u_i^{n+1} are presented. However, the application of the lemma once again completes the proof since these values are already convex combinations of the old values u_j^n.

Corollary. For the Fisher's equation (3.45) if the initial values are in the interval $u_i^0 \in [0,1]$, then valuess of u are still in this interval for any values of β and time-step size h, provided that the "semi-implicit" treatment in Eq. (3.46) of nonlinearreaction term is applied.

Proof. Theorem 1 and Remark 1 immediately imply the statement.

Theorem 2. Method B1 is unconditionally stable if applied to Eq. (2.1) in the absence of q.

Proof. In [64] the authors examined the stability of the OEH method by Von Neumann stability analysis for the linear convection-diffusion equation. In this theorem, we will follow their way of proof. We consider two subsequent time steps

S1: $u_i^{\mathrm{n}+1} = \dfrac{u_i^{\mathrm{n}} + r\left(u_{i-1}^{\mathrm{n}} + u_{i+1}^{\mathrm{n}}\right)}{1+2r}$ for even i, S2: $u_i^{\mathrm{n}+1} = (1-2r)u_i^{\mathrm{n}} + r\left(u_{i-1}^{\mathrm{n}+1} + u_{i+1}^{\mathrm{n}+1}\right)$ for odd i

S3: $u_i^{\mathrm{n}+2} = \dfrac{u_i^{\mathrm{n}+1} + r\left(u_{i-1}^{\mathrm{n}+1} + u_{i+1}^{\mathrm{n}+1}\right)}{1+2r}$ for even i, S4: $u_i^{\mathrm{n}+2} = (1-2r)u_i^{\mathrm{n}+1} + r\left(u_{i-1}^{\mathrm{n}+2} + u_{i+1}^{\mathrm{n}+2}\right)$ for odd i

and I test the algorithm made as the unification of these four stages, i.e. a doubled time step size $\vec{u}^{\mathrm{n}} \to \vec{u}^{\mathrm{n}+2}$ algorithm. I test $u_i^{\mathrm{n}+1}$ from the formula of S3 by substituting S2 (it is allowed since S2 and S3 refer to the same set of nodes) and obtain:

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

with the notation $\sigma = 2r$ this can be written as

$$(1+\sigma)u_i^{\mathrm{n}+2} = (1-\sigma)u_i^{\mathrm{n}} + \sigma\left(u_{i-1}^{\mathrm{n}+1} + u_{i+1}^{\mathrm{n}+1}\right), \tag{3.50}$$

which is equivalent to Eq. (3.3) in [64] apart from the convection term. Noteworthy that Eq. (3.50) contains only values at the uncoupled set of nodes where the sum of the space and time index $i+n$ is odd. We can proceed with Eq. (3.50) for the investigation of linear stability and "if the computation at the uncoupled set of odd-numbered points is stable, we have also stability at all even points" [64]. Boonkkamp and Verwer obtained that the requirement of stability gives a restriction only to the magnitude of the convection term, thus we can conclude that the scheme is unconditionally stable for the heat or diffusion equation. Indeed, the resulting algebraic equation

$$(1+\sigma)\xi^2 - 2\sigma g\xi - (1-\sigma) = 0,$$

where $g = \cos\theta, \theta \in \mathbb{R}$, has the following roots

$$\xi_{\pm} = \frac{\sigma g \pm \sqrt{\sigma^2\left(g^2-1\right)+1}}{1+\sigma},$$

and the absolute value of these roots is majorated by 1 for all $g \in [-1,1]$, thus the errors cannot grow to infinity. □

We note that I have constructed the stages S1-S4 of the above schemes in a matrix form. If we denote these by Ue, Eo, Uo and Ee, respectively ('U' and 'E' are UPFD and explicit while 'e' and 'o' stand for even and odd cells), the matrix of the two-step algorithm can be written as $H = Ee\left(Uo\left(EoUe\right)\right)$ and $\vec{u}^{\mathrm{n}+2} = H\vec{u}^{\mathrm{n}}$. I have calculated the absolute values of the eigenvalues of this matrix H for several values of N, and found that the largest one is exactly 1, which verifies the unconditional stability property. On the other hand, the (infinity) norm [65] of H is proportional to r, which explain the large errors for large time step sizes. However, the powers of *H* have smaller norms, moreover,

$$\lim_{n\to\infty}\left\|H^n\right\| = 1 \quad \forall r > 0,$$

which makes easier to understand the reason of the unconditional stability.

3.3.2. Convergence

In This subsection, I analyse the consistency and the convergence of B1 and D5 methods following the same way in [66], [67].

Theorem 3. The order of convergence of B1 (reversed UPFD-EE) and D5 (CNe-CNe) hopscotch algorithms as time integrators for fixed spatial discretization Δx is two.

Proof. Since formulas (3.16)-(3.25) are first order by their own, if one considers a single time step, it is obviously impossible to prove that these algorithms are second order. Because of this we consider a doubled time step with time step size 2h. First we have to calculate the formula $u_i^{\mathrm{n}+2}$ in the case of the two methods for both odd and even nodes.

In the case of algorithm B1, by using Eq. (3.20) we have for the even nodes at the end of the first time step

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

Let us introduce the notation $u_{i\pm j}^{n} = u_{i-j}^{n} + u_{i+j}^{n}$, $j \in \{1,2,3\}$. Now we start the second time step by substituting the formula we have just obtained with shifted space index into the terms u_{i-1}^{n} and u_{i+1}^{n} in the UPFD formula (3.17) for the even cells to obtain

$$u_i^{n+2} = \frac{1}{(1+2r)^2}\left[u_i^n + 2r\left(u_{i\pm1}^n + r u_{i\pm2}^n\right)\right]. \tag{3.51}$$

Finally using Eq. (3.20) again we can calculate the new values for the odd cells as well:

$$u_i^{\mathrm{n}+2} = \frac{1-2r}{1+2r}\left[u_i^{\mathrm{n}} + r u_{i\pm1}^n\right] + \frac{r}{(1+2r)^2}\left[u_{i\pm1}^n + 2r\left(2u_i^{\mathrm{n}} + u_{i\pm2}^n + r\left(u_{i\pm3}^n + u_{i\pm1}^n\right)\right)\right]. \tag{3.52}$$

Now we repeat this calculation for D5. For the even nodes at the end of the first time step we have

$$u_i^{n+1} = u_i^n\,\mathrm{e}^{-2r} + \frac{1}{2}\left[u_{i\pm1}^n\,\mathrm{e}^{-2r} + \frac{2u_i^n + u_{i\pm2}^n}{2}\left(1-\mathrm{e}^{-2r}\right)\right]\left(1-\mathrm{e}^{-2r}\right).$$

At the second time step we have, for the even cells

$$\begin{aligned} u_i^{n+2} = {} & u_i^n\,\mathrm{e}^{-4r} + \frac{1}{2}\left(u_{i-1}^n + u_{i+1}^n\right)\mathrm{e}^{-2r}\left(1-\mathrm{e}^{-2r}\right)\left(1+\mathrm{e}^{-2r}\right) \\ & + \frac{2u_i^n + u_{i-2}^n + u_{i+2}^n}{4}\left(1-\mathrm{e}^{-2r}\right)^2\left(1+\mathrm{e}^{-2r}\right), \end{aligned} \tag{3.53}$$

and for the odd cells:

$$u_i^{n+2} = u_i^n\,\mathrm{e}^{-4r} + \left[\begin{array}{l} u_{i\pm1}^n\,\mathrm{e}^{-2r} + u_{i\pm1}^n\,\mathrm{e}^{-4r} + \dfrac{2u_i^n + u_{i\pm2}^n}{2}\mathrm{e}^{-2r}\left(1-\mathrm{e}^{-2r}\right)\left(1+\mathrm{e}^{-2r}\right) \\ + \dfrac{3u_{i\pm1}^n + u_{i\pm3}^n}{4}\left(1-\mathrm{e}^{-2r}\right)^2\left(1+\mathrm{e}^{-2r}\right) \end{array}\right]\frac{1-\mathrm{e}^{-2r}}{2}. \tag{3.54}$$

The following power series expansions will be used:

$$\frac{1}{1+2r} = 1-2r+4r^2-8r^3+O\left(r^4\right), \quad \frac{1}{(1+2r)^2} = 1-4r+12r^2-32r^3+O\left(r^4\right).$$

$$e^{-2r} = 1-2r+2r^2-\frac{4}{3}r^3+O\left(r^4\right), \quad 1-e^{-2r} = 2r-4r^2+\frac{4}{3}r^3+O\left(r^4\right), \text{ etc.} \tag{3.55}$$

Now let us consider one space dimensional equation (2.1) in the absence of the heat source. If we assume that the analytical solution of equation is sufficiently smooth, and denote Δx by s, we can write

$$u_i^{\mathrm{n}+2} = u_i^n + 2hu_t + 2h^2u_{tt} + \frac{4}{3}h^3u_{ttt} + O\left(h^4\right),\ u_{i-1}^n = u_i^n - su_x + \frac{s^2}{2}u_{xx} - \frac{s^3}{6}u_{xxx} + \frac{s^4}{12}u_{xxxx} + O\left(s^5\right),$$

etc.

From this point we omit the higher order terms, after which we obtain

$$u_{i\pm1}^n = 2u_i^n + s^2 u_{xx} + \frac{s^4}{12}u_{xxxx}\,, \quad u_{i\pm2}^n = 2u_i^n + 4s^2 u_{xx} + \frac{4}{3}s^4 u_{xxxx}\,, \quad u_{i\pm3}^n = 2u_i^n + 9s^2 u_{xx} + \frac{27}{4}s^4 u_{xxxx}\,.$$

We substitute the exact solution to the obtained formulas, first to Eq. (3.51). With this we have

$$u_i^{n+2} = \frac{1}{(1+2r)^2}\left[u_i^n + 2r\left(u_{i\pm1}^n + r u_{i\pm2}^n\right)\right] + \varepsilon,$$

where ε is the local truncation error of the doubled time step. Using Eq. (3.55) we obtain

$$\varepsilon = u_i^{n+2} - \left(1 - 4r + 12r^2 - 32r^3\right)u_i^n - \left(2r - 8r^2 + 24r^3\right)u_{i\pm1}^n - \left(2r^2 - 8r^3\right)u_{i\pm2}^n. \quad (3.56)$$

Inserting the truncated expansions into this equation we obtain

$$\varepsilon = \boxed{2hu_t}\,\boxed{+2h^2u_{tt}} + \frac{4}{3}h^3u_{ttt}\,\boxed{-2\frac{\alpha h}{s^2}s^2u_{xx}} + 8\left(\frac{\alpha h}{s^2}\right)^3 s^2u_{xx} - \frac{1}{6}\alpha h s^2 u_{xxxx}\,\boxed{-2\left(\frac{\alpha h}{s^2}\right)^2 s^4u_{xxxx}} + \frac{26}{3}\left(\frac{\alpha h}{s^2}\right)^3 s^4u_{xxxx}$$

Since u is the exact solution of original PDE (2.1) in the absence of heat source, thus

$$\begin{aligned} u_t &= \alpha u_{xx} \\ u_{tt} &= \left(\alpha u_{xx}\right)_t = \alpha\left(u_t\right)_{xx} = \alpha^2 u_{xxxx}, \end{aligned} \quad (3.57)$$

and because of this, the terms in the boxes cancel each other, while other terms can also be simplified using Eq. (3.57), thus the final result is the following:

$$\varepsilon = \frac{4}{3}h^3u_{ttt} - \frac{1}{6}\alpha h\Delta x^2 u_{xxxx} + 8\alpha^2\frac{h^3}{\Delta x^4}u_t + \frac{26}{3}\alpha\frac{h^3}{\Delta x^2}u_{tt}.$$

Thus the leading terms of the global truncation error is

$$\varepsilon^T = \frac{4}{3}h^2u_{ttt} - \frac{1}{6}\alpha\Delta x^2 u_{xxxx} + 8\alpha^2\frac{h^2}{\Delta x^4}u_t + \frac{26}{3}\alpha\frac{h^2}{\Delta x^2}u_{tt}.$$

Now we perform the same procedure in the remaining three cases. Employing Eq. (3.55) to Eq. (3.52) yields

$$\varepsilon = u_i^{n+2} - \left(1 - 4r + 12r^2 - 32r^3\right)u_i^n - \left(2r - 8r^2 + 22r^3\right)u_{i\pm1}^n - 2r^2u_{i\pm2}^n - 2r^3u_{i\pm3}^n.$$

Substituting the truncated expansions into this, after some calculations we obtain

$$\varepsilon = \boxed{2hu_t} + \boxed{2h^2u_{tt}} + \frac{4}{3}h^3u_{ttt}\,\boxed{-2\alpha hu_{xx}} - 8\left(\frac{\alpha h}{s^2}\right)^3 s^2u_{xx} - 2\frac{\alpha h}{s^2}\frac{s^4}{12}u_{xxxx} - 22\left(\frac{\alpha h}{s^2}\right)^3\frac{s^4}{12}u_{xxxx}\,\boxed{-2\alpha^2h^2u_{xxxx}} - \frac{17}{6}\left(\frac{\alpha h}{s^2}\right)^3 s^4u_{xxxx}.$$

The terms in the boxes again cancel each other, thus the local error is

$$\varepsilon = \frac{4}{3}h^3 u_{ttt} - \frac{1}{6}\alpha h \Delta x^2 u_{xxxx} - 8\alpha^2 h^3 \frac{1}{\Delta x^4} u_t - \frac{14}{3}\alpha h^3 \frac{1}{\Delta x^2} u_{tt},$$

thus the global error is

$$\varepsilon^T = \frac{4}{3}h^2 u_{ttt} - \frac{1}{6}\alpha \Delta x^2 u_{xxxx} - 8\alpha^2 \frac{h^2}{\Delta x^4} u_t - \frac{14}{3}\alpha \frac{h^2}{\Delta x^2} u_{tt}.$$

Applying Eq. (3.55) to Eq. (3.53) we have

$$\varepsilon = u_i^{\mathrm{n}+2} - \left(1 - 4r + 8r^2 - \frac{32}{3}r^3\right)u_i^{\mathrm{n}} - \left(2r - 8r^2 + \frac{52}{3}r^3\right)u_{i\pm1}^{\mathrm{n}} - \left(2r^2 - 6r^3\right)\left(2u_i^{\mathrm{n}} + u_{i\pm2}^{\mathrm{n}}\right).$$

After similar calculations as above, we obtain the global error of the D5 method at even nodes:

$$\varepsilon^T = \frac{4}{3}h^2 u_{ttt} - \frac{1}{6}\alpha \Delta x^2 u_{xxxx} - \frac{20}{3}\alpha^2 \frac{h^2}{\Delta x^4} u_t - \frac{59}{9}\alpha \frac{h^2}{\Delta x^2} u_{tt}.$$

Finally, applying Eq.(3.55) to Eq.(3.54) we obtain

$$\varepsilon = u_i^{\mathrm{n}+2} - \left(1 - 4r + 12r^2 - \frac{92}{3}r^3\right)u_i^{\mathrm{n}} - \left(2r - 8r^2 + \frac{70}{3}r^3\right)u_{i\pm1}^{\mathrm{n}} - \left(2r^2 - 10r^3\right)u_{i\pm2}^{\mathrm{n}} - 2r^3 u_{i\pm3}^{\mathrm{n}},$$

and the final result is the following global error:

$$\varepsilon^T = \frac{4}{3}h^2 u_{ttt} - \frac{1}{6}\alpha \Delta x^2 u_{xxxx} - \frac{4}{3}\alpha^2 \frac{h^2}{\Delta x^4} u_t - \frac{19}{9}\alpha \frac{h^2}{\Delta x^2} u_{tt}.$$

One can see that the global error is second order in the time step size for both methods and both for odd and even cells. □

Remark 2 The first two terms in the global errors are the usual error of space and time discretization. The two other terms are, however, contain the space step size in the denominator. It means that $\frac{h}{\Delta x^2}$ should go to zero to achieve convergence, thus the methods are only conditionally consistent. This is usual for these kind of methods, see the original paper of Gourlay [43] (Theorem 5 and the remark after it), and also in [37].

Remark 3 The analytical solution of Eq. (2.8), in the absense of the heat source, using the $\vec{u}^n$ values as initial conditions is

$$\vec{u}^{n+1} = \mathrm{e}^{2Mh}\vec{u}^n = \left(1 + 2Mh + 2M^2h^2 + \ldots\right)\vec{u}^n.$$

Performing the operations we can obtain for a general element at the end of the doubled time step:

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

This is the same expression up to second order in *r* as the appropriate expressions of the numerical values, thus it is obvious that if one consider fixed spatial discretization, then the numerical

solution unconditionally converges to the exact solution of the obtained ODE system with the order 2.

4. A FAMILIES OF ADAPTIVE TIME STEP CONTROLLERS FOR SOLVING THE NON-STATIONARY HEAT CONDUCTION EQUATION

I systematically test families of explicit adaptive step-size controllers for solving the diffusion or heat equation. After discretizing the space variables as in the conventional method of lines, we are left with a system of ODEs. Different methods for estimating the local error and techniques for changing the step size when solving a system of ODEs were suggested previously by several researchers. In my work [68], those local error estimators and techniques are used to generate different types of adaptive step size controllers. Those controllers are applied to a system of ODEs resulted from discretizing diffusion equation. The performance of the controllers was compared in the case of three different experiments. The first and the second system are heat conduction in homogeneous and inhomogeneous medium, while the third one contains a moving heat source that can correspond to a welding process.

In general, the explicit numerical methods for solving a system of ordinary differential equations use a fixed time step. This kind of approach can perform poorly if the solution changes rapidly in some parts of the integration interval and slowly in other ones. Using a small constant time-step, where the solution changes rapidly, can help to circumvent the problem of poor performance, but this small constant time-step may cause unnecessary computational cost where the solution changes slowly. Using adaptive methods based on automatic time-step selection can be the remedy of the expensive numerical computations [69]. Adapting the time-step size during the integration process is not just a matter of improving the performance of the integrator; it makes the solution of difficult problems practical [70]. There are at least three crucial factors when it comes to designing adaptive step-size integrators: the method of calculating the solution at the end of the actual time step, the method of estimating the local error in each time step, and the approach for changing the time-step [71].

Among a large number of explicit numerical methods available for solving a system of ODEs, our study will be restricted to two fundamentally different types of explicit methods, which are the single step multi-stage Runge-Kutta (RK) method and the recently published LNe3 method [37].

Estimating the local error in Runge-Kutta methods when applied to ordinary differential equations, was studied intensively in the literature of numerical analysis [69]–[79]. The methods of estimating the local error can be classified into two types: the methods which use the information of only a single step, and those which use the information of successive steps. The methods of the second type are out of the scope of this thesis. The most well-known method, of the first type, for estimating the local error is to calculate the dependent variable u first by using a full-time step h, and to recalculate it using two halved time steps $h/2$. The difference between the two values of u represents the local error LE. Another common method is the pseudo-iterative formula which uses a RK formula of order (p), then a RK formula of order $(p+1)$ which uses

the already calculated quantities of the lower order RK formula to spare time, that is why these algorithms are called embedded methods [77]. In his early work [72], Merson has derived a formula that gives a plausible estimation for the local error which is valid only if the ordinary differential equation is linear. Later, Scraton [78] suggested a new formula for estimating the local error but without any restrictions on its validity. The particular schemes of most interest are given in Eqs. (7), (8), and (9) of his paper. However, the formula suggested by Scraton can be applied only to a single differential equation, not to a system of ODEs [76]. To overcome this shortcoming, England [76] introduced a process that can be valid even when it is applied to a general system of ODEs. Also, Shampine [71] proposed a new formula in his work and compared the performance of different error estimators.

The approach for changing the step size in case of ODEs, can be done using the elementary controller:

$$h_{new} = s\left(\frac{TOL}{LE}\right)^{\frac{1}{p}} h_{present}, \tag{4.1}$$

where $s<1$ is the safety factor, TOL is tolerance specified by the user, h is the time step, LE is the estimated local error, and p is the order of the method. The elementary controller changes the step size based on the current estimation of the local error. This elementary controller generally shows a good performance, but there are some exceptions. For instance, the time step size can be limited by the stability properties of the used method, which in turn causes an oscillation in the step size sequences. More details about the shortcomings of the elementary controller can be found in [79]. Based on control theory, Gustafsson [80], [81] introduced the so-called PI controller to overcome the problem of oscillating step size. Its adaptivity algorithm is:

$$h_{new} = \left(\frac{TOL}{LE_n}\right)^{K_I} \left(\frac{LE_{n-1}}{LE_n}\right)^{K_P} h_{present}, \tag{4.2}$$

where K_I and K_P are constants, LE_n is the estimated local error at the current step size, LE_{n-1} is the estimated local error at the previous step size, and h is the step size. Unlike the elementary controller, the PI controller changes the step size based on the past history of the local error estimation. Later, Soderlind [82]–[84] investigated this type of controllers. He developed new strategies for adaptive step size based on digital control theory.

In the references mentioned in this chapter, the authors tested the methods for estimating the local error and the approaches for changing the step size only in the case of small systems of ODEs. Those algorithms might be efficient when they are applied to a single ordinary differential equation or a system of ODEs including limited number of equations. The objective of this paper is to design and extensively test adaptive step size controllers based on the previous mentioned studies then apply those algorithms to equation system (2.8), where the size of matrix M is big.

4.1. The I and PI Step-Size Controllers

Suppose that we applied an explicit numerical method of order p, with step size $h_{present}$, to equation system (2.8) in order to get an approximated solution u_i^{n+1}. Also, assume that we used some method for estimating the local error and the local error estimation, based on that method, is denoted by LE_i. The norm of the error estimation is [85, p. 26]

$$err^{n+1} = \max_{1\le i\le N}\left\{\frac{\left|LE_i\right|}{AbsTol_i + \left|u_i^{n+1}\right| \mathrm{Re}lTol}\right\}, \tag{4.3}$$

where $AbsTol$ and $\mathrm{Re}lTol$ are the relative and absolute tolerances which can be defined by the user.

We note that in Eq. (17) in [69] and Eq. (4.10) in [86], the authors used a different formula for calculating the norm of the error estimation. In their formula they did not only consider the value of u_i^{n+1}, but they also considered the value of u_i^n. Based on our numerical experiments, we do believe that including the value of u_i^n in the calculation does not have significant effect and it only causes extra cost.

Now, after calculating err^{n+1} we can change the step size using the following formula [69]

$$h_{new} = \min\left(f_{\max}, \max\left(f_{\min}, f_s\beta^{n+1}\right)\right)h_{present} \tag{4.4}$$

where β^{n+1} is a function of err^{n+1} and it depends on the type of the step size controller. That function will be defined for each type of controller individually as we will see later. The nonnegative number f_s is a safety factor, and it is used to increase the probability of accepting the step size in the next iteration. The factors $f_{\min}$ and $f_{\max}$ are used to prevent the step size from decreasing or increasing too rapidly. In our codes we set the following values for the factors $f_s = 0.9, f_{\min} = 0.1, f_{\max} = 5$. If $err^{n+1} \le 1$ the step size is accepted and the solution is advanced with u_i^{n+1} and the step size will be modified by (4.4). If $err^{n+1} > 1$ the step size and the solution u_i^{n+1} are rejected and the calculations are repeated with new time step calculated by (4.4).

In elementary controller I (asymptotic) the value of the function β^{n+1} depends on the current estimated error as follows:

$$\beta^{n+1} = \left(err^{n+1}\right)^{\frac{-1}{p}}. \tag{4.5}$$

In PI controller the value of the function β_{n+1} depends on the current and previous estimated errors as follows:

$$\beta^{n+1} = \left(err^{n+1}\right)^{\frac{-k_1}{p}} \left(err^{n}\right)^{\frac{k_2}{p}}. \tag{4.6}$$

Here the values $k_1 = 0.8, k_2 = 0.31$ are taken from [87]. For the first step I considered that $err^n = 1$. To be systematic, I run all our codes for adaptive controller algorithms considering the following decreasing series for the tolerance: $AbsTol = \mathrm{Re}lTol = 2^{-1}, 2^{-2}, ...$

4.2. Description of the Methods

For the sake of simplicity, all the schemes resulted from Runge-Kutta methods and adaptive Runge-Kutta methods are described when they are applied to a single ordinary differential equation. Nevertheless, these schemes can be straightforwardly expanded to solve the system in Eq. (2.8). For initial value problem (IVP) of the form:

$$\left.\begin{aligned} \frac{du}{dt} &= f\left(t,u\right) \\ u\left(t^0\right) &= u^0 \end{aligned}\right\}, \tag{4.7}$$

the general s-stage Runge Kutta method can be written as follows [88]:

$$\left.\begin{aligned} u^{n+1} &= u^n + h\sum_{i=1}^{s} b_i k_i \\ k_i &= f\left(t^n + c_i h, u^n + h\sum_{j=1}^{s} a_{ij} k_j\right), i = 1:s. \end{aligned}\right\}, \tag{4.8}$$

where the a, b and c are constants and can be defined using Butcher array.

4.2.1. Group A: Dormand-Prince Fifth-Order Runge-Kutta Method

In Dormand-Prince method the k functions are evaluated as follows [79]:

$$\left.\begin{aligned}
k_1 &= f\left(t^n, u^n\right) \\
k_2 &= f\left(t^n + \frac{1}{5}h, u^n + \frac{1}{5}hk_1\right) \\
k_3 &= f\left(t^n + \frac{3}{10}h, u^n + \frac{3}{40}hk_1 + \frac{9}{40}hk_2\right) \\
k_4 &= f\left(t^n + \frac{4}{5}h, u^n + \frac{44}{45}hk_1 - \frac{56}{15}hk_2 + \frac{32}{9}hk_3\right) \\
k_5 &= f\left(t^n + \frac{8}{9}h, u^n + \frac{19372}{6561}hk_1 - \frac{25360}{2187}hk_2 + \frac{64448}{6561}hk_3 - \frac{212}{729}hk_4\right) \\
k_6 &= f\left(t^n + h, u^n + \frac{9017}{3168}hk_1 - \frac{355}{33}hk_2 + \frac{46732}{5247}hk_3 + \frac{49}{176}hk_4 - \frac{5103}{18656}hk_5\right)
\end{aligned}\right\}. \quad (4.9)$$

The fifth-order Runge-Kutta formula is:

$$u^{n+1} = u^n + h\left(\frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6\right). \quad (4.10)$$

Estimating the local error requires another formula. To do that Dormand considered an extra evaluation:

$$k_7 = f\left(t^n + h, \frac{35}{384}hk_1 + \frac{500}{1113}hk_3 + \frac{125}{192}hk_4 - \frac{2187}{6784}hk_5 + \frac{11}{84}hk_6\right). \quad (4.11)$$

The embedded formula is:

$$\hat{u}^{n+1} = u^n + h\left(\frac{5179}{57600}k_1 + \frac{7571}{16695}k_3 + \frac{393}{640}k_4 - \frac{92697}{339200}k_5 + \frac{187}{2100}k_6 + \frac{1}{40}k_7\right). \quad (4.12)$$

The coefficients a_{7j} in Eq. (4.11) are designedly chosen to be the same as the coefficients b_i in Eq. (4.10). It means that Eq. (4.11) is equivalent to:

$$k_7 = f\left(t^n + h, u^{n+1}\right). \quad (4.13)$$

Now for the next step (when $err^{n+1} \le 1$), I set $k_1 = k_7$. This trick is called FSAL (first-same-as-last). This means that in the case of acceptance, the evaluation of the function k_7 can be reused again in the following step as k_1 which in turn reduces the cost of computations. The local error estimation can be calculated:

$$LE = \left|\hat{u}^{n+1} - u^{n+1}\right|. \quad (4.14)$$

Our notations hide the fact that the functions LE, k are vectors if the method is applied to a system of ODEs. Since the method is introduced to a single differential equation, as I mentioned

previously, the i index has been dropped from all the formulas of those functions. Also, for the functions of the subsequent subsections, except Subsection 4.2.4, the i index has been dropped as well.

Substituting Eqs. (4.12) and (4.14) into Eq. (4.3), and considering Eqs. (4.4) and (4.5) result in the adaptive step size scheme of the Dormand-Prince method with I controller type, which will be denoted as "DPRK5(4) #I". Considering Eq. (4.6) instead of Eq. (4.5) leads us to the adaptive step size scheme of the Dormand-Prince method, but with PI controller type, which will be denoted as "DPRK5(4) #PI". For the sake of comparison, I will also test the scheme of Eq. (4.10), which is a fifth-order Runge-Kutta method with a fixed step size. This method will be denoted as "non adaptive DPRK5(4)".

Based on Eq. (4.9), we can design adaptive controllers which use the doubling step size technique. First, we take a single step of size h and we use the six stages of Eq. (4.9) in order to calculate the solution u^{n+1} using Eq. (4.10). Second, we take two steps of size $\frac{h}{2}$ to recalculate the solution, denoted again by $\hat{u}^{n+1}$, using Eqs. (4.9) and (4.10) two times. The local error can be simply estimated as in Eq. (4.14), then substituted into Eq. (4.3) to obtain:

$$err^{n+1} = \max\left\{\frac{|LE|}{AbsTol + |u^{n+1}|\,\mathrm{Re}lTol}\right\}. \tag{4.15}$$

Note that instead of $\hat{u}^{n+1}$ now there is u^{n+1} in the denominator. If the error norm is tolerable, the step size is accepted and there are three possibilities to advance the solution. The first possibility is to accept the solution u^{n+1} resulted from taking a single step. With that possibility we can design two controllers based on Eqs. (4.4), (4.5), and (4.6). Those adaptive controllers will be denoted as "RKduplicate 1 # I" and "RKduplicate 1 # PI". The second possibility is to accept the more accurate solution $\hat{u}^{n+1}$ and we will be left again with "RKduplicate 2 # I" and "RKduplicate 2 # PI". The third one is Richardson extrapolation (see Eq. (4.5) from [86]), which combines the solutions u^{n+1} and $\hat{u}^{n+1}$ to produce another, more accurate solution as follows:

$$\tilde{u}^{n+1} = \hat{u}^{n+1} + \left(\frac{\hat{u}^{n+1} - u^{n+1}}{2^p - 1}\right), \tag{4.16}$$

where p is the order of the method, which is four in this scheme. Based on the last formula, we can design two adaptive controllers and they will be denoted as "RKduplicate 3 # I" and "RKduplicate 3 # PI".

4.2.2. Group B: Scraton's Fourth-Order Runge-Kutta Method

In his work [78], Scraton inroduced a Runge-Kutta method with five stages:

$$\left.\begin{aligned}
k_1 &= f\left(t^n, u^n\right)\\
k_2 &= f\left(t^n + \frac{2}{9}h, u^n + \frac{2}{9}hk_1\right)\\
k_3 &= f\left(t^n + \frac{1}{3}h, u^n + \frac{1}{12}hk_1 + \frac{1}{4}hk_2\right)\\
k_4 &= f\left(t^n + \frac{3}{4}h, u^n + \frac{3}{128}\left(23hk_1 - 81hk_2 + 90hk_3\right)\right)\\
k_5 &= f\left(t^n + \frac{9}{10}h, u^n + \frac{9}{10000}\left(-345hk_1 + 2025hk_2 - 1224hk_3 + 544hk_4\right)\right)
\end{aligned}\right\}, \quad (4.17)$$

and the fourth-order scheme is:

$$u^{n+1} = u^n + h\left(\frac{17}{162}k_1 + \frac{81}{170}k_3 + \frac{32}{135}k_4 + \frac{250}{1377}k_5\right). \quad (4.18)$$

To estimate the local error, Scraton evaluated the following functions:

$$\left.\begin{aligned}
q &= -\frac{1}{18}k_1 + \frac{27}{170}k_3 - \frac{4}{15}k_4 + \frac{25}{153}k_5\\
r &= \frac{19}{24}k_1 - \frac{27}{8}k_2 + \frac{57}{20}k_3 - \frac{4}{15}k_4\\
s &= k_3 - k_1
\end{aligned}\right\},$$

then the local error estimation is given by the following nonlinear formula:

$$LE = -\frac{qr}{s}. \quad (4.19)$$

Scraton stated that subtracting the local error calculated in Eq. (4.19) from Eq. (4.18) will increase the order of the scheme to be five:

$$u^{n+1} = u^n + h\left(\frac{17}{162}k_1 + \frac{81}{170}k_3 + \frac{32}{135}k_4 + \frac{250}{1377}k_5\right) + \frac{qr}{s}. \quad (4.20)$$

For easy recognition, I refer to schemes (4.18) as "non adaptive RKSc 1", and (4.20) as "non adaptive RKSc 2". To design an adaptive step size controller based on Scranton's error estimation, we substitute Eqs. (4.18) and (4.19) into Eq. (4.4), then we use Eqs. (4.4) and (4.5). This is an adaptive-Scraton scheme with I controller type. We have two possibilities to advance the solution when the step size is accepted: First, to use Eq. (4.18) and this type will be referred to as "RKSc 1 # I". Second, to use Eq. (4.20), and this type will be referred to as "RKSc 2 # I". If we repeat the

previous steps, but using Eq. (4.6) instead of Eq. (4.5), we will get another two types of adaptive step size controllers which are “RKSc 1 # PI” and “RKSc 2 # PI”.

4.2.3. Group C: England Fourth-Order Runge-Kutta Method

England used four-stage Runge-Kutta method in the first step [71], [76]:

$$\left.\begin{aligned} k_1 &= f\left(t^n, u^n\right) \\ k_2 &= f\left(t^n + \frac{1}{2}h, u^n + \frac{1}{2}hk_1\right) \\ k_3 &= f\left(t^n + \frac{1}{2}h, u^n + \frac{1}{4}\left(hk_1 + hk_2\right)\right) \\ k_4 &= f\left(t^n + h, u^n - hk_2 + 2hk_3\right) \end{aligned}\right\}, \tag{4.21}$$

and the fourth order formula is:

$$u^{n+1} = u^n + \frac{h}{6}\left(k_1 + 4k_3 + k_4\right). \tag{4.22}$$

Unlike in the case of other methods, England started the second time step before estimating the local error, and he used the same stage-formulas as in the first step:

$$\begin{aligned} k_5 &= f\left(t^n + h, u^{n+1}\right) \\ k_6 &= f\left(t^n + \frac{3}{2}h, u^{n+1} + \frac{1}{2}hk_5\right) \\ k_7 &= f\left(t^n + \frac{3}{2}h, u^{n+1} + \frac{1}{4}\left(hk_5 + hk_6\right)\right). \end{aligned}$$

Before completing the second step, an extra evaluation is made which enables us to estimate the local error accumulated in the two steps:

$$k_{extra} = f\left(t^n + 2h, u^n + \frac{h}{6}\left(-k_1 - 96k_2 + 92k_3 - 121k_4 + 144k_5 + 6k_6 - 12k_7\right)\right).$$

The local error estimation according to England is:

$$LE_{En} = \frac{h}{90}\left(-k_1 + 4k_3 + 17k_4 - 23k_5 + 4k_7 - k_{extra}\right). \tag{4.23}$$

Now, we substitute Eqs. (4.22) and (4.23) into Eq. (4.3)

$$err^{n+2} = \max\left\{\frac{\left|LE_{En}\right|}{AbsTol + \left|u^{n+1}\right|\mathrm{Re}lTol}\right\}. \tag{4.24}$$

If $err^{n+2} \le 1$, the evaluation of the last function in the second step can now proceed,

$$k_8 = f\left(t^n + 2h, u^{n+1} - hk_6 + 2hk_7\right),$$

and we accept the following numerical solution:

$$u^{n+2} = u^{n+1} + \frac{h}{6}\left(k_5 + 4k_7 + k_8\right). \tag{4.25}$$

If $err^{n+2} > 1$, we reject the step size without evaluating the function k_8 and we repeat the first step with a new step size. In this case we lose seven function evaluations. When the time step is acceptable, 9 function evaluations will be performed if we consider the calculation of k_{extra} as well. In other words, we make only $4\frac{1}{2}$ function evaluations for each step, while it requires 5 function evaluations per step if we use the logic of the classical adaptive Runge-Kutta such as Fehlberg method. It means that we saved $1/2$ function evaluation for each step. One might argue that saving only a half function evaluation per step cannot compensate the expensive cost of the probability of losing seven function evaluations when the step size is rejected. According to the experience, for a well-designed adaptive controller the probability of a rejected step size is low and the majority of the steps are accepted. It is worthy here to recall that there are more effective tricks which enables us to make only 4 function evaluation per step, for instance, the so called FSAL trick previously described and the local extrapolation technique [89, p. 717].

Nevertheless, in case of either a rejected or an accepted step, we should repeat or proceed the calculations with the new step size. If we use Eq. (4.24) along with Eqs. (4.4) and (4.5), an adaptive controller of type I is applied and it will be denoted as "RKEn #I". If we use Eq. (4.24) along with Eqs. (4.4) and (4.6), a new adaptive controller is applied but of type "PI", and it will be denoted as "RKEn #PI". The simple 4th order RK scheme using only Eq. (4.22) will be referred to as "non adaptive RKEn".

Shampine [71] used the same function evaluations as England used but with a new local error estimator of the form:

$$LE_{Sh} = \frac{h}{180}\left(k_1 - 4k_3 - 17k_4 + 23k_5 - 4k_7 + k_{extra}\right) = -\frac{1}{2}LE_{En}. \tag{4.26}$$

Again, based on Shampine's formula for estimating the local error and using Eqs. (4.3), (4.4), (4.5) , and (4.6) we can design another two types of step size controllers which are "RKSh #I" and "RKSh #PI". Since England used two steps to calculate the numerical solution, then it is fair to compare the "RKEn #" types with that adaptive controller which depends on doubling the step size. First, we take a single step of size h and we use the four stages of Eq. (4.21) in order to calculate the solution u^{n+1} using Eq. (4.22). Second, we take two steps of size $\frac{h}{2}$ to recalculate the solution $\hat{u}^{n+1}$ using Eqs. (4.21) and (4.22) two times. The local error can be simply estimated now as in Eq. (4.14) and then Eq. Is used to calculate err^{n+1}. If the error norm is tolerable, then

the step size is accepted, and we again have the same three possibilities, as in Subsection 4.2.1, to calculate the new solution. The first possibility is to accept the solution u^{n+1} resulted from taking a single step. In this case we can design two controllers based on Eqs. (4.4), (4.5) and (4.6). Those adaptive controllers will be denoted as "RKdoubling 1 # I" and "RKdoubling 1 # PI". The second possibility is to accept the solution $\hat{u}^{n+1}$, and this will be denoted again with "RKdoubling 2 # I" and "RKdoubling 2 # PI". The third one is to use Richardson extrapolation:

$$\tilde{u}^{n+1} = \hat{u}^{n+1} + \left(\frac{\hat{u}^{n+1} - u^{n+1}}{2^p - 1} \right), \tag{4.27}$$

where *p* is the order of the method, which is four in this scheme. Based on the last formula, we can design two adaptive controllers and they will be denoted as "RKdoubling 3 # I" and "RKdoubling 3 # PI".

4.2.4. Group D: Second-Order LNe3 Method:

The LNe3 method has been introduced in Subsection 2.2. The method deals with the spatially discretized heat equation, or generally, any system of first order linear ODEs. Unlike the previous subsections, this method will be explained when it is applied to a system of ODEs instead of single differential equation. The solution produced by Eq. (2.17), which is called LNe2 scheme, will be denoted by u_i^{n+1}, while the solution produced by iterating Eq. (2.17), which is called LNe3, will be denoted by $\hat{u}_i^{n+1}$. The local error can be estimated by the following formula:

$$LE_i = \left| \hat{u}_i^{n+1} - u_i^{n+1} \right|. \tag{4.28}$$

Substituting Eqs. (2.17) and (4.28) into Eq. (4.3), and considering Eqs.(4.4), (4.5), and (4.6), adaptive controllers will be applied and they will be denoted as "ALNe3 #I" and "ALNe3 #PI".

We must note that when Fehlberg introduced his adaptive step size controller, many practitioners questioned the robustness of the method at that time. They thought that it was risky to estimate the local error using the same evaluation points. Later, experiments showed that this concern was not a problem in practice [89, p. 716]. Since the embedded LNe3 method is new, one might have the same concern. As we can see later, our experiments showed a very stable performance for that method.

4.3. Numerical Experiments and Reuslts

The numerical solution and the reference solution are compared only at t_{fin}, which is the final time of the simulation and will be specified later. We measure the accuracy using the global L_∞

error, which is the maximum of the absolute difference between the reference temperature $u_{\mathrm{j}}^{\mathrm{ref}}$ and the temperature $u_{\mathrm{j}}^{\mathrm{num}}$ calculated by our numerical methods at the final time:

$$\mathrm{Error}(L_\infty) = \max_{1\le \mathrm{j} \le N} \left| u_{\mathrm{j}}^{\mathrm{ref}}(t_{\mathrm{fin}}) - u_{\mathrm{j}}^{\mathrm{num}}(t_{\mathrm{fin}}) \right|. \tag{4.29}$$

In the first experiment, I will test the previously described methods in case of a linear diffusion equation in the absence of the heat source, which yields a non-stiff system of ODEs after spatial discretization. In the second experiment a linear diffusion equation in inhomogeneous media will be tested. The third experiment will treat the problem of a moving heat source.

The simulations are conducted using the MATLAB R2020b software on a desktop computer with an Intel Core (TM) i11-11700F. Since the analytical solution does not exist for complicated systems, the reference solution was calculated by the implicit ode15s solver setting very stringent error tolerance ('RelTol' and 'AbsTol' were both 10^{-14}).

4.3.1. Experiment 1: Non-Stiff Linear Diffusion Equation

In this experiment we consider Eq. (2.7) in 2 space dimension $(x, y, t) \in [0,1] \times [0,1] \times \left[0, 2\times 10^{-3}\right]$, subjected to zero Neumann boundary conditions. The space domain was divided into $N = N_x \times N_y = 50 \times 50$, thus we have 2500 cells. The initial conditions were generated randomly using the built-in function 'rand' in MATLAB $u_i(0) = rand$. The 'rand 'function generates random numbers uniformly distributed in the interval $[0,1]$. The resistances and capacities were set as $Rx_i = Rz_i = 1, C_i = 10^{-3}$. The stiffness ratio of the introduced system is roughly 4×10^4 and the CFL limit for the explicit Euler scheme is 2.5×10^{-4} .

For all the groups of methods, Figures 4.1, 4.2, 4.3 and 4.4 show that the adaptive controllers of type (I) achieve approximately the same or slightly better performance as the controllers of type (PI) when both use the same method for estimating the local error. For example – as we can see from Figure (4.1) – the curves of the controllers "RKdoubling 1 #I" and "RKdoubling 1#PI" are almost identical. From Figures 4.1 and 4.3, we can clearly see that using Eq. (4.27), which was suggested in theorem 4.1 in [86], improved the performance of the algorithms based on the step doubling technique. However, the embedded Runge-Kutta-Dormand-Prince adaptive controller showed better performance than all the types of adaptive controllers based on the step doubling technique as we can see in Figure 4.1. It is not a surprising result and was clearly stated in [85, p. 911] that step doubling has been superseded by a more efficient stepsize adjustment algorithm based on embedded Runge-Kutta formulas when it is applied to a system of ODEs. Another important observation is related to England and Shampine methods of group C. Although England stated in his work [76] that his method is valid "when applied to a general system of differential

equations", our numerical experiments appear to contradict his claims. As we can see in Figure 4.3, the adaptive controllers based on England and Shampine showed a poor performance when they were applied to our system (2.7). Scraton suggested a new scheme (4.20), Eq. (9) in his work [78], to increase the order of the method. As we can see in Figure 4.2, the non-adaptive scheme (4.20), which is the red colour curve, showed unstable behaviour. However, the adaptive schemes, which are RKSc 2 # I and RKSc 2 # PI, showed a stable performance but without improving the accuracy if they are compared to "RKSc 1 # I" and "RKSc 1 # PI".

For each group of methods, we can see that the non-adaptive scheme is faster than the adaptive controller. Here a question arises: why do we use the adaptive controller if the non- adaptive scheme is faster? The subsequent discussion will show that the non-adaptive Runge-Kutta scheme is vulnerable, and the stability can be easily violated when small changes in the conditions or parameters of the experiment take place. To illustrate that, the time domain of the experiment 2×10^{-3} was replaced by 2×10^{-1} while all other settings and conditions remained the same. Figure 4.5 shows the performance of the methods of group A. We can see that after we made a small change in the time domain, the behaviour of "non adaptive DPRK5(4)" became unstable and its curve blew up and became discontinuous. Despite of the fact that the behaviour of the adaptive controllers changed after we changed the time domain, their performance remained stable. It does indeed look like that the running time, in case of adaptive controllers, is relatively independent of the required accuracy. This point is out of the scope of this thesis and one can see a good explanation in Appendix D of [90]. So, the advantage of using the adaptive controller is not always about reducing the computational time, but it is sometimes more about providing reliable results when the non-adaptive scheme fails.

The "non adaptive LNe3" method is excluded from the last discussion and in the second experiment I will show that this method remains stable regardless of the conditions of the system. It was proved mathematically, and verified by numerical experiments, that the method is unconditionally stable [37].

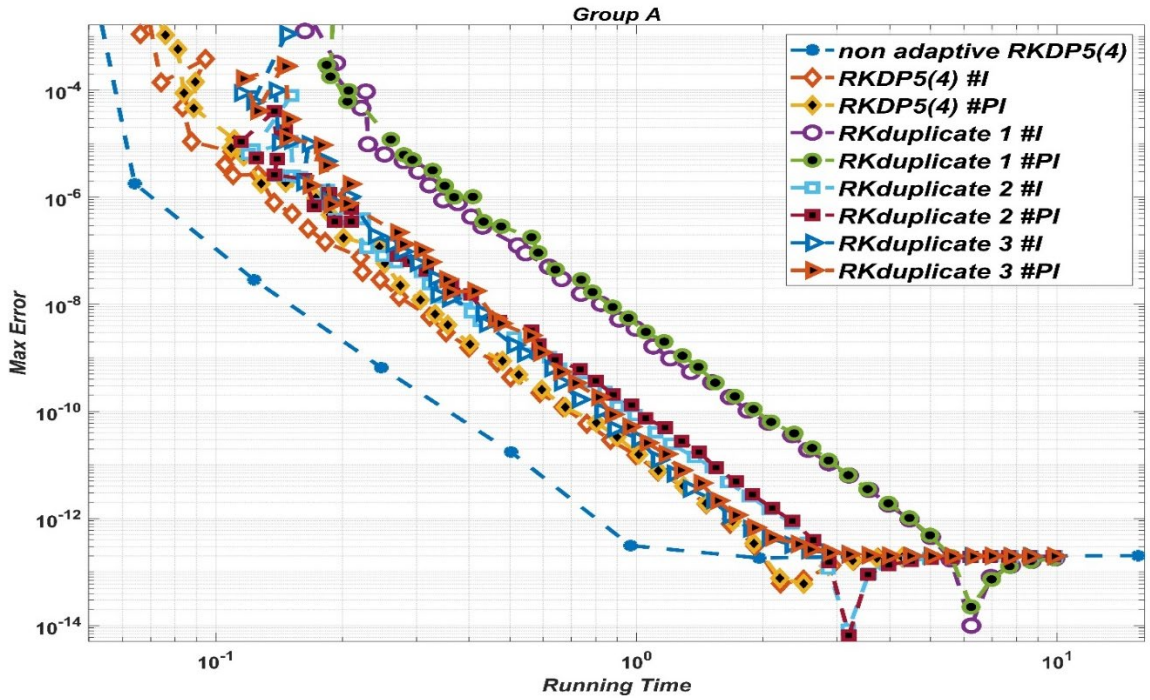

Figure 4.1. Experiment 1: L_∞ errors as a function of the running time for group A.

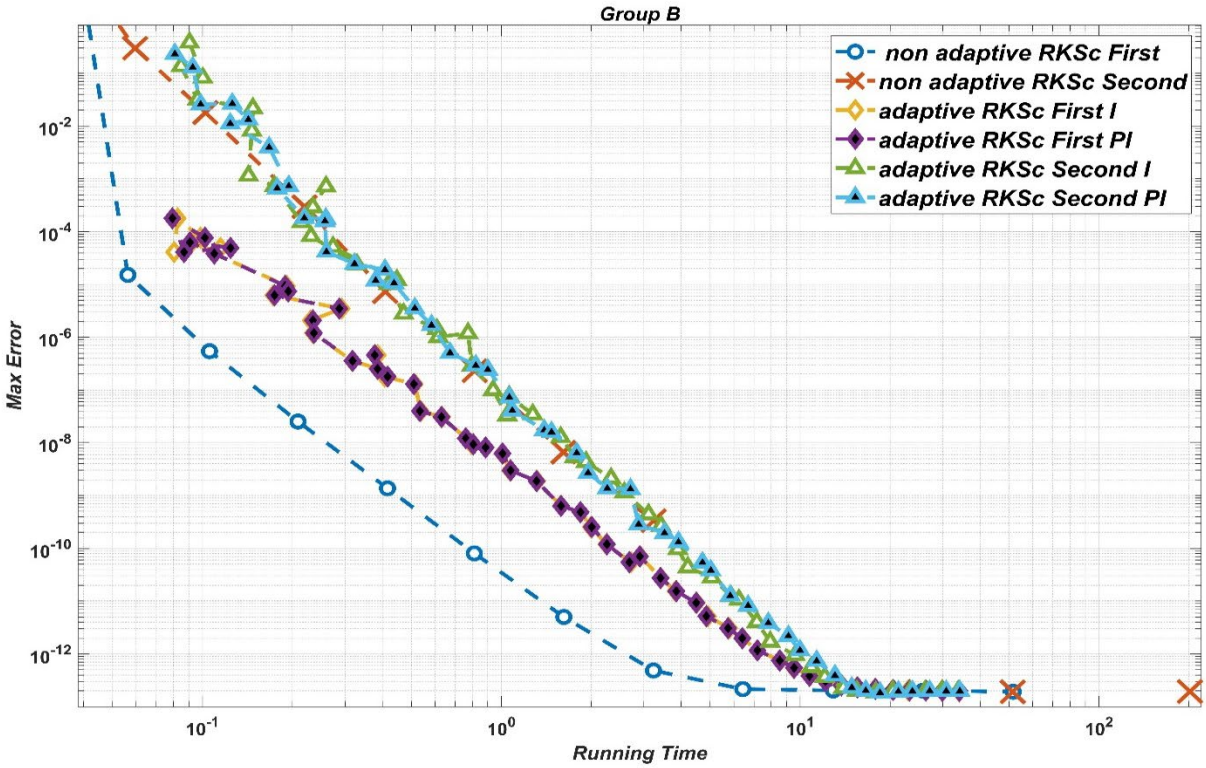

Figure 4.2. Experiment 1: L_∞ errors as a function of the running time for group B.

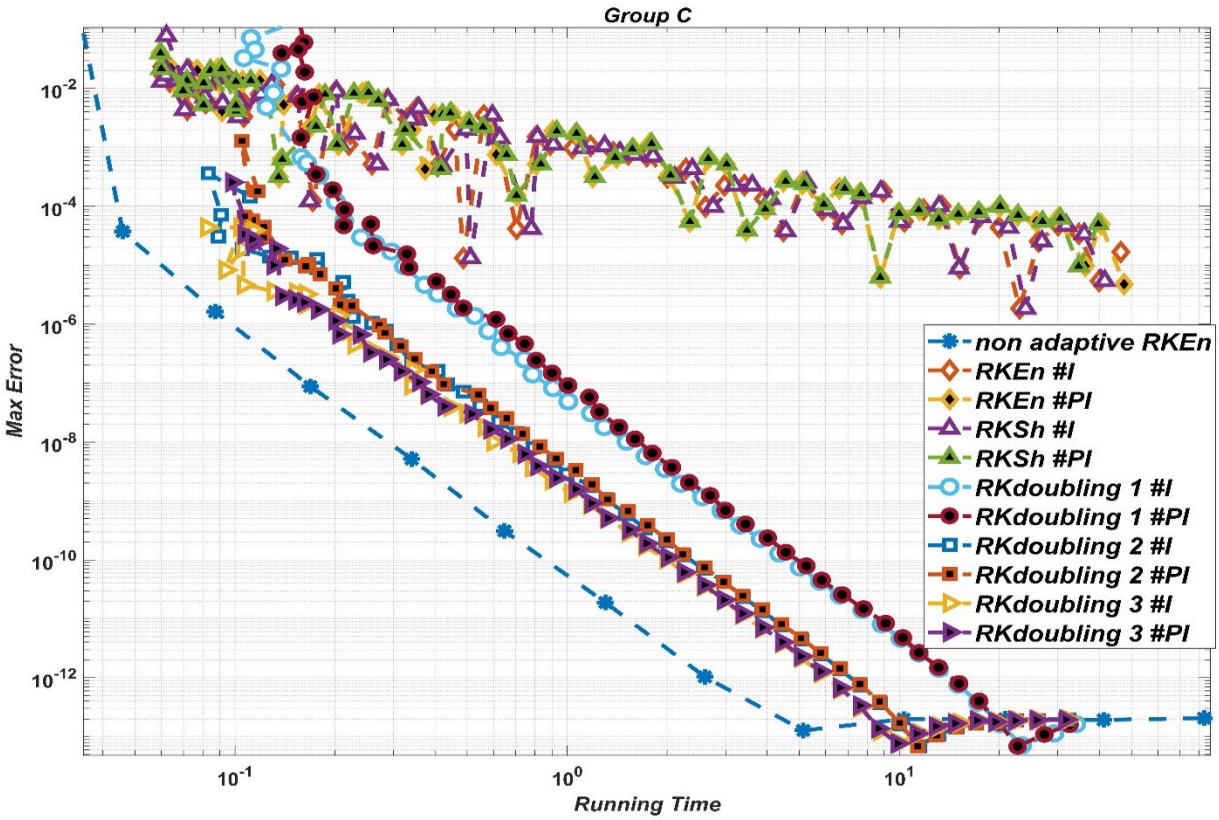

Figure 4.3. Experiment 1: L_∞ errors as a function of the running time for group C.

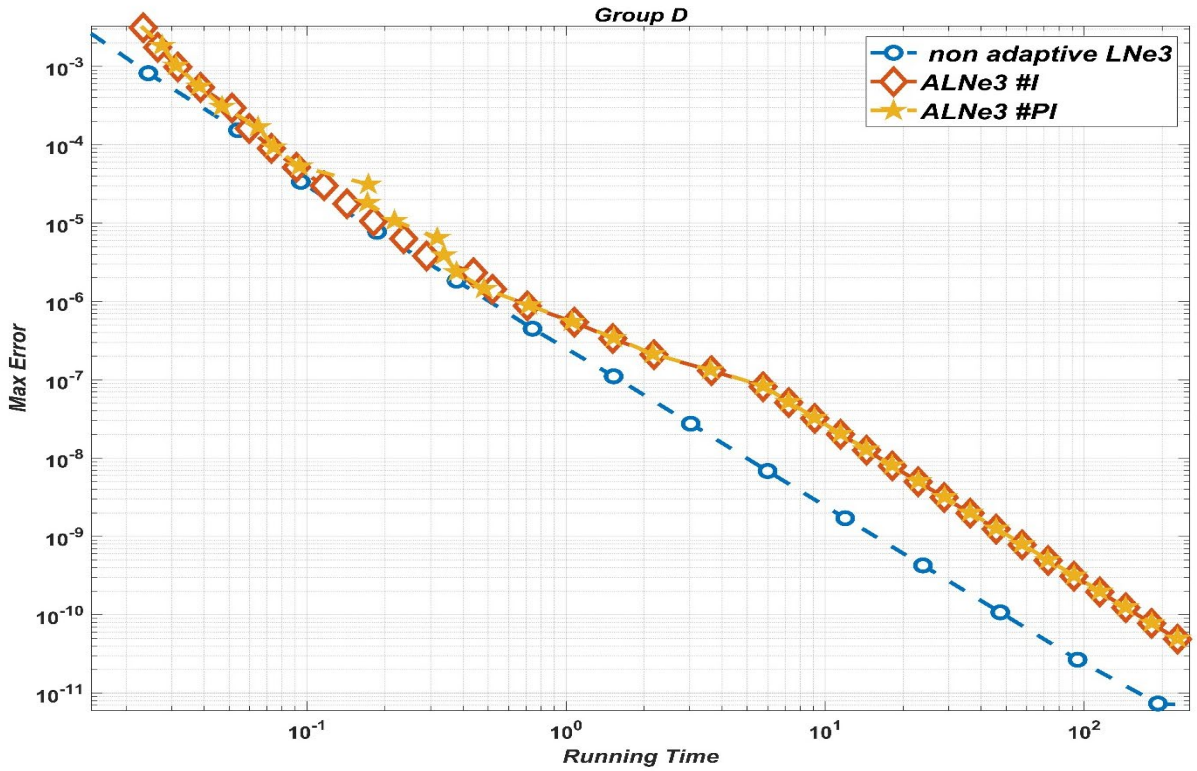

Figure 4.4. Experiment 1: L_∞ errors as a function of the running time for group D

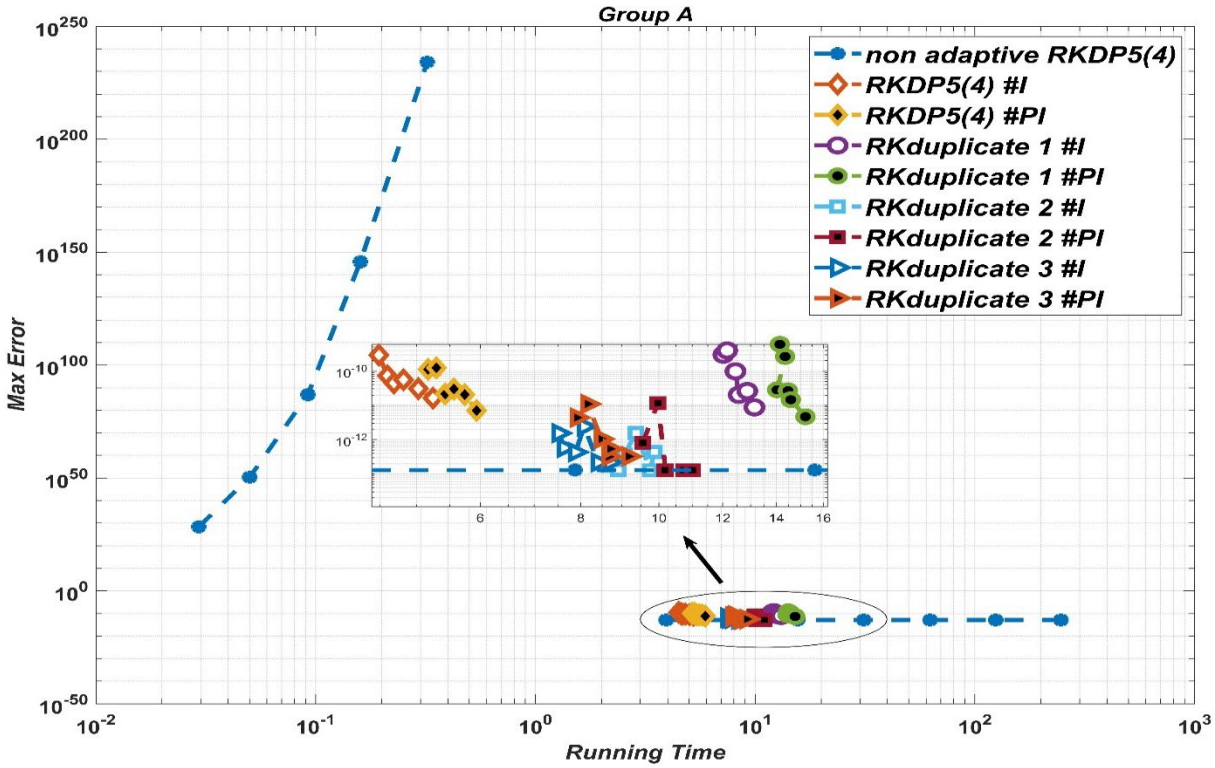

Figure 4.5. Experiment 1: L_∞ errors as a function of the running time for group A when t_{fin} is larger.

4.3.2. Experiment 2: Stiff Linear Diffusion Equation

In this experiment we consider Eq. (2.7) in 2 space-dimensions $(x,y,t)\in[0,1]\times[0,1]\times\left[0,2\times10^{-4}\right]$, subjected to zero Neumann boundary conditions. The space domain was divided into $N=N_x\times N_y=20\times20$, thus we have 400 nodes. The capacity and the resistances obeyed the following formula:

$$C=\left(\left(10^{-4}-1\right)x+1\right)10^{-4},R_x=\left(10^{-4}-1\right)x+1,R_y=y+1.$$

The stiffness ratio of the resulted system is 1.05×10^{9}, while $h_{\max}\simeq9\times10^{-10}$. For such relatively stiff system, all the non-adaptive schemes based on Runge-Kutta showed a poor performance. They can provide a reliable result only when the time step is very small which increases the cost of the computations. The adaptive controllers which used England or Shampine methods for estimating the local error showed also a poor performance when they are compared to those adaptive controllers which used the step doubling technique. As we can see in Figure 4.6, the highest accuracy that the England or Shampine formulas could reach was of the order 10^{-8}, while

it is of the order 10^{-13} if the step doubling technique is used. Also, the performance of the controllers of type (I) was identical and sometimes even better than the performance of those of type (PI). For the sake of comparison, in Figure 4.7 we selected the most accurate methods of groups A, B and C, as well as the methods of group D.

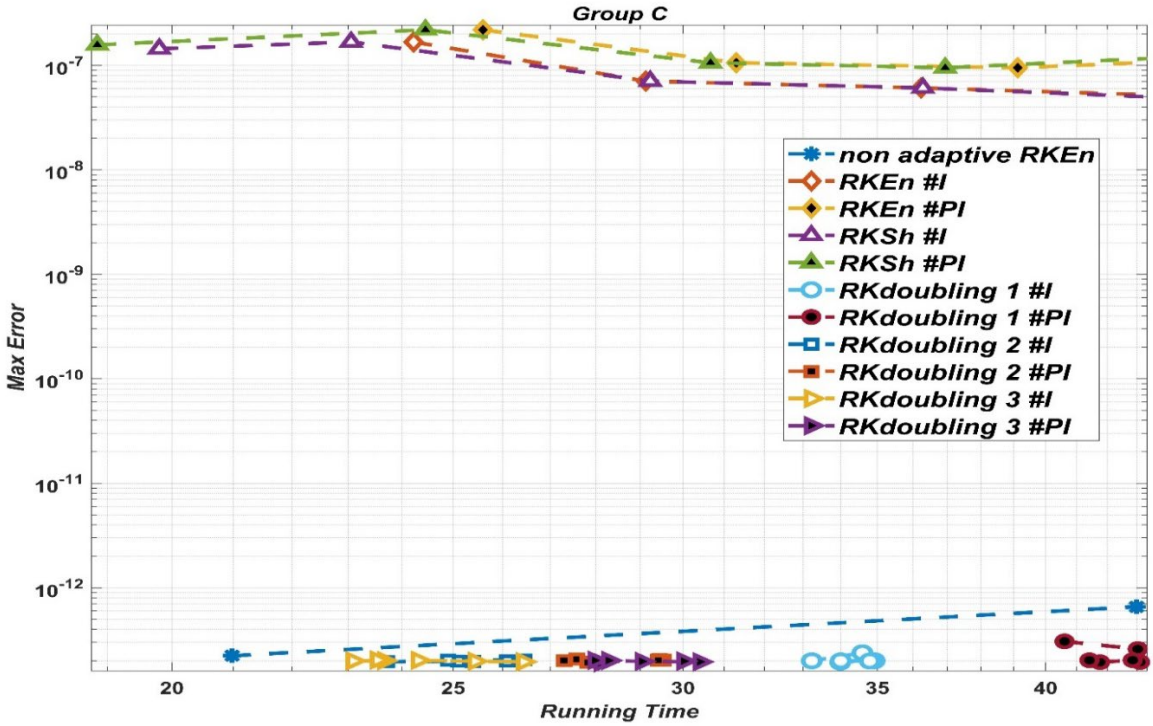

Figure 4.6. Experiment 2: L_∞ errors as a function of the running time for group C.

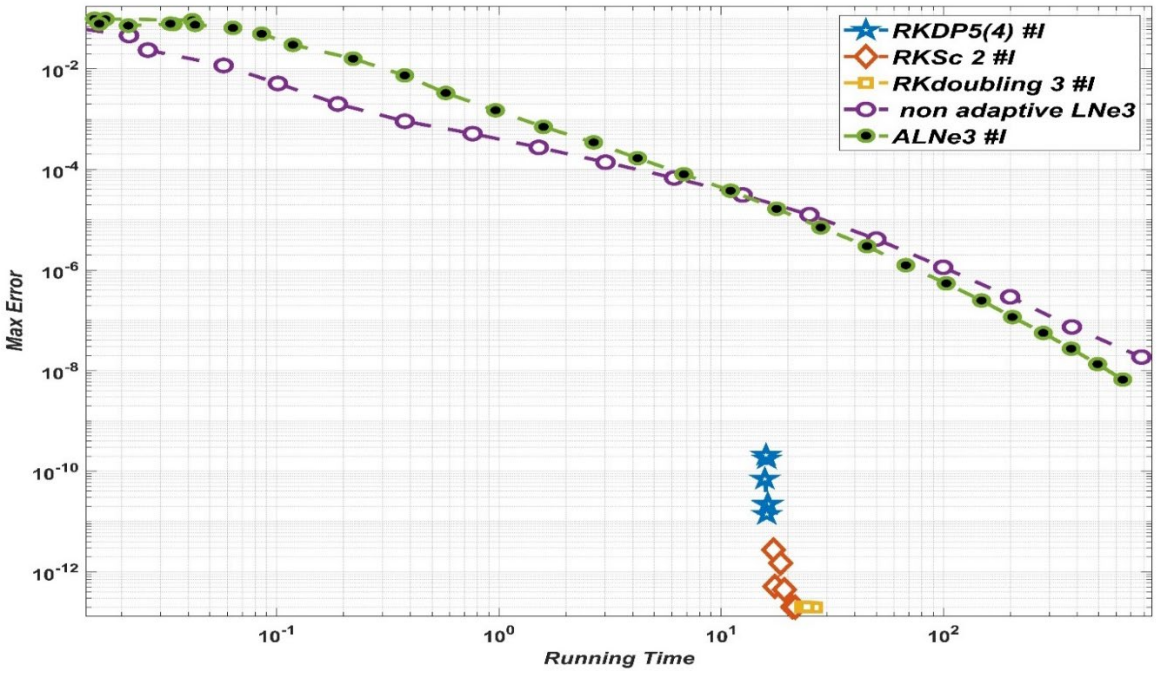

Figure 4.7. Experiment 2: L_∞ errors as a function of the running time for the selected methods.

4.3.3. Experiment 3: Stiff Diffusion Equation with a Moving Heat Source

In this experiment we consider Eq. (2.7) in 2 space-dimensions $(x, y, t) \in [0,1] \times [0,1] \times \left[0, 2 \times 10^{-5}\right]$, subjected to zero Neumann boundary conditions. The space domain was divided into $N = N_x \times N_y = 30 \times 30$, thus we have 900 cells. The initial conditions are generated randomly using the built-in function 'rand' in MATLAB $u_i(0) = rand$. The capacity and the resistances obeyed the following form:

$$C = \left(\left(10^{-6} - 1\right)x + 1\right)10^{-4}, R_x = \left(10^{-6} - 1\right)x + 1, R_y = y + 1.$$

The stiffness ratio of the resulted system is roughly 1.5×10^{11}, while $h_{\max} \simeq 6.4 \times 10^{-12}$. Here we consider a moving Gaussian point heat source which takes the formula:

$$q = q_{\max}\, \mathrm{e}^{-\left\{\frac{\left(y+y_0-v_y t\right)+\left(x+x_0-v_x t\right)}{r^2}\right\}},$$

where $q_{\max}$ is the maximum heat flux at the centre of the heat source, r is the effective heating radius of the heat source, v_x and v_y are the velocities of the heat source in x and z directions respectively and (x_0, y_0) is the initial position of the heat source. The parameters of the heat source are set to be:

$$\left.\begin{array}{l} q_{\max} = 10^6 \\ v_x = 25 \times 10^3, v_y = 0 \\ (x_0, y_0) = (0, -0.5) \end{array}\right\}.$$

It means that heat source will move with a constant speed along the positive direction of the x axis. The effective heating radius is chosen to be $r = 5\Delta x = 5\Delta z$ to ensure that there are at least four nodes inside the effective heating dimeter. Figure. 4.8 shows the contour of the temperature distribution at the end of the time interval and the trace of the heating process refers to the trajectory of the heat source.

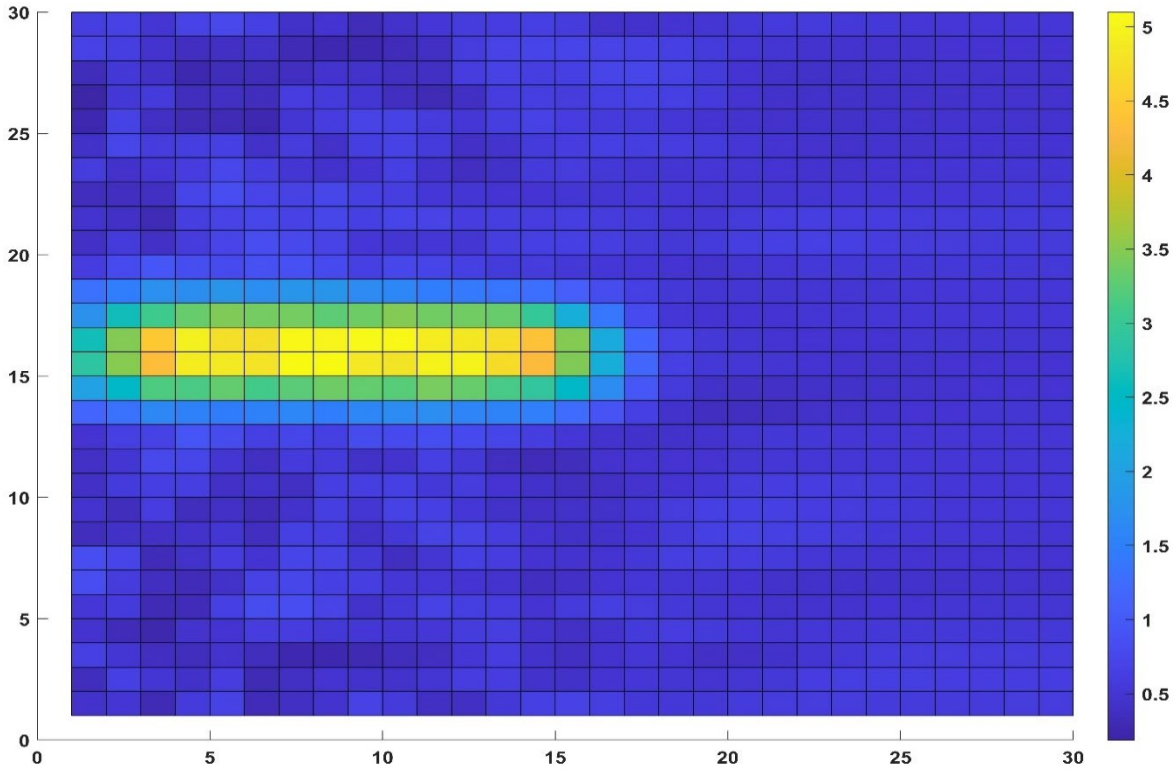

Figure 4.8. Experiment 3: The contour of the temperature distribution at the end of the time interval.

We can clearly see from Figure 4.9 that the LNe3 and its adaptive scheme are much faster than the adaptive schemes of Runge-Kutta when high accuracy is not required, while adaptive Runge-Kutta schemes are more applicable when the desired accuracy goes beyond a certain level (which is of order 10^{-6} in our experiment).

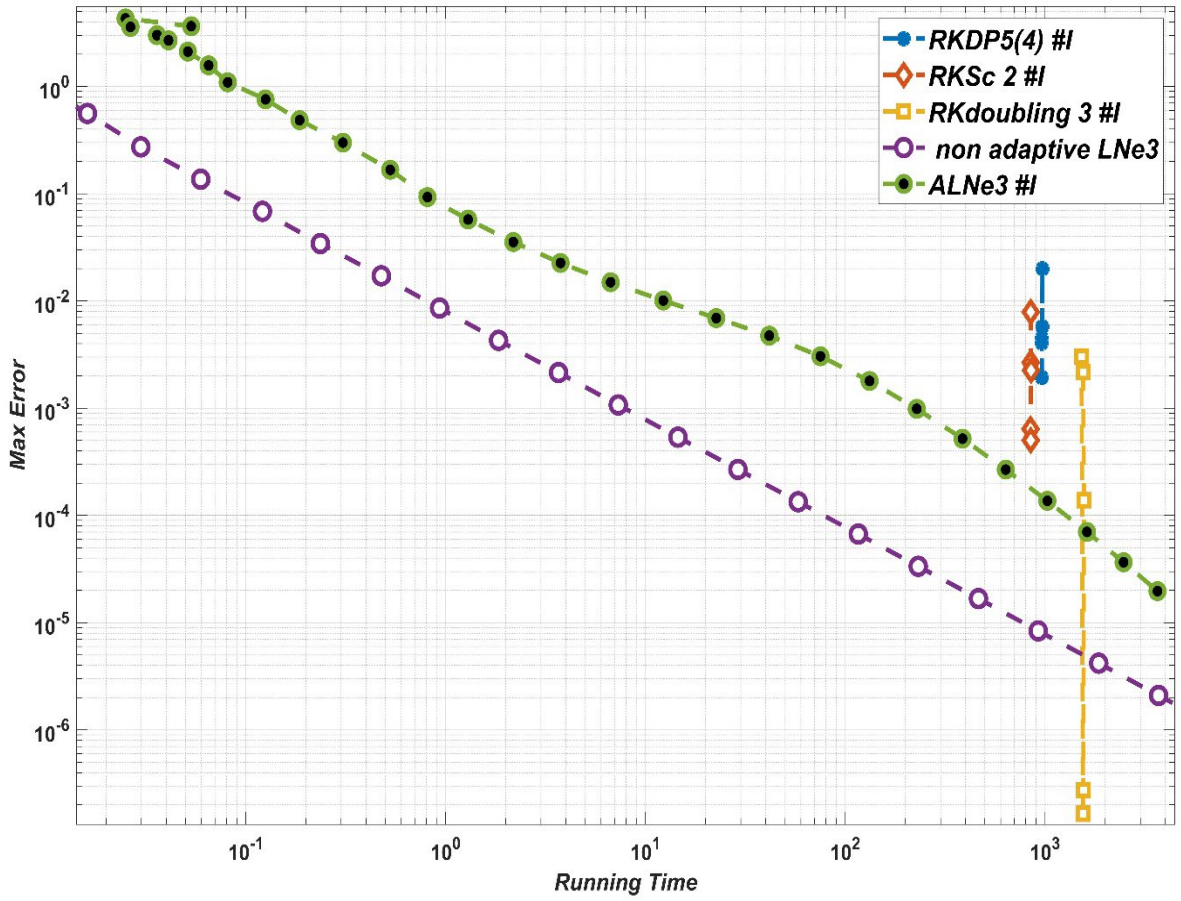

Figure 4.9. Experiment 3: L_∞ errors as a function of the running time.

5. FAMILIES OF ADAPTIVE TIME STEP CONTROLLERS FOR THE TRANSIENT DIFFUSION EQUATION WITH DIFFUSION COEFFICIENT DEPENDING ON BOTH SPACE AND TIME

In this chapter I deal with the time-dependent diffusion equation in one dimension, where the diffusion coefficient itself depends simultaneously on space and time. I introduce and design several adaptive time-step controllers to solve numerically that equation. The simplest regular diffusion PDE in one space dimension is

$$\frac{\partial u(x,t)}{\partial t} = \alpha \frac{\partial^2 u(x,t)}{\partial x^2}, \quad u(x,t=0) = u^0(x). \tag{5.1}$$

The boundary conditions will be discussed in the concrete analytical and numerical examples. In this work, we modify Eq. (5.1) to have a diffusion coefficient which is non-constant in two senses. We introduce a new variable, which is a combination of the space and time variable: $\eta = \frac{x}{\sqrt{t}} \in \mathbb{R}$. The diffusion coefficient has the simplest power law dependence on this variable: $\bar{\alpha}(\eta) = \alpha\eta^m$, where α is a constant, whose physical dimension depends on the concrete value of m. Inserting it into the diffusion equation we obtain

$$\frac{\partial u(x,t)}{\partial t} = \alpha \frac{\partial}{\partial x}\left(\eta^m \frac{\partial u(x,t)}{\partial x}\right) = \alpha\left(m\eta^{m-1}\frac{\partial \eta}{\partial x}\frac{\partial u(x,t)}{\partial x} + \eta^m \frac{\partial^2 u(x,t)}{\partial x^2}\right). \tag{5.2}$$

In our published work [91], Ferenc Barna has introduced the analytical solution of Eq. (5.2) which will be used as a reference solution in my numerical experiments. Since the analytical procedures are out of the scope of my research, I will introduce here only the final formula of the solution without going through the details:

$$u(x,t) = c\sqrt{\frac{t^{1/2-2\gamma}}{x}} \cdot \mathrm{e}^{\frac{\left(x/\sqrt{t}\right)^{2-m}}{4(m-2)}} \cdot W_{\frac{4\gamma-1}{2m-4},\frac{m-1}{2m-4}}\left(\frac{\left(x/\sqrt{t}\right)^{2-m}}{2(m-2)}\right), \tag{5.3}$$

where γ is a real number, and c is a normalization constant. W is the Whittaker functions [92] given by the following formula:

$$W_{\kappa,\mu}(z) = \mathrm{e}^{-\frac{z}{2}} z^{\mu+\frac{1}{2}} U\left(\mu-\kappa+\frac{1}{2}, 1+2\mu; z\right),$$

where U is kummer function. It can be shown with careful parameter analysis that for negative values of γ, the solutions have an exponential growth at large time and spatial coordinates which may be considered non-physical, thus we exclude them from further numerical investigations.

5.1. The Sapce-Temporal Discretization and the Applied Schemes

We consider Eq. (2.3) in the absence of the heat source:

$$c(x)\rho(x)\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}\left(k(x)\frac{\partial u}{\partial x}\right). \tag{5.4}$$

For simplicity, we consider $c(x,t)\equiv 1$ and $\rho(x,t)\equiv 1$ in this work, and all the space- and time-dependence of the diffusivity will be incorporated into the conductivity $k(x,t)$. To discretize Eq. (5.4), we follow the same procedures in Subsection 2.1 to obtain a system of ODEs:

$$\frac{du_i}{dt}=\frac{u_{i-1}-u_i}{R^n_{i,i-1}C_i}+\frac{u_{i+1}-u_i}{R^n_{i,i+1}C_i}. \tag{5.5}$$

Unlike Eq. (2.7), the resistance in Eq. (5.5) contains the superscript n which means that resistance is not only a function of the space but also the time. The the resisance in this case is given by the following formula:

$$R^n_{i,i+1}=\frac{\Delta x}{D\left(x_{i,i+1}/\sqrt{t^n}\right)^m},\quad i=1,...,N-1. \tag{5.6}$$

In the last equation, the index i of the variable Δx has been dropped since we are dealing with equidistance grid.

Now we can introduce the adaptive controllers which we will use to reperduce the solution of Eq. (5.2) numerically. In Chapter 4, we came out with the result that the adaptive schemes using (PI) controller do not have any advantage compared to the same schemes using (I) controller. So, we use only the adaptive controllers of type (I) in this chapter. The quantities in Eq. (3.27) will be introduced here again taking into consideration that the diffusion coefficient is function of the time and space:

$$r^n_i=\frac{h}{C_i}\left(\frac{1}{R^n_{i,i-1}}+\frac{1}{R^n_{i,i+1}}\right)\ \text{and}\ A^n_i=\frac{h}{C_i}\left(\frac{u^n_{i-1}}{R^n_{i,i-1}}+\frac{u^n_{i+1}}{R^n_{i,i+1}}\right),\ i=1,...,N,\ n=0,...,T. \tag{5.7}$$

A) The apdative LNe3 schemes

As I introduced in Chapter 2, the LNe3 method consists of three stages. In the first stage we use the CNe scheme. Considering the new notations in Eq. (5.7), the CNe scheme has the following formula:

$$u^{n+1}_i=u^n_i\cdot e^{-r^n_i}+\frac{A^n_i}{r^n_i}\left(1-e^{-r^n_i}\right). \tag{5.8}$$

In the second stage, we use the scheme in Eq. (5.8) as predictor in order to calculate new A_i^{new} values:

$$A_i^{\text{new}} = \frac{h}{C_i}\left(\frac{u_{i-1}^{\text{pred}}}{R_{i,\,i-1}^n} + \frac{u_{i+1}^{\text{pred}}}{R_{i,\,i+1}^n}\right). \tag{5.9}$$

The LNe2 scheme has the following formula:

$$u_i^{n+1} = u_i^n e^{-r_i^n} + \left(A_i^n - \frac{A_i^{\text{new}} - A_i^n}{r_i^n}\right)\frac{1-e^{-r_i^n}}{r_i^n} + \frac{A_i^{\text{new}} - A_i^n}{r_i^n}. \tag{5.10}$$

In the third stage, the values in Eq. (5.10) one can first recalculate A_i^{new} again, then repeat Eq. (5.10) to obtain the LNe3 scheme.

Each stage provides values of the unknown function u at the end of the actual time step. It means that there are three possibilities to compare these values with one another in order to estimate the local error. The first possibility means that the difference between the numerical solutions calculated in the first and second stages is used as a local error estimator as follows:

$$LE_{C1L2} = \left|\hat{u}^{n+1} - u^{n+1}\right|, \tag{5.11}$$

where $\hat{u}^{n+1}$ and u^{n+1} are the solutions calculated by Eqs. (5.8) and (5.10) respectively. The indices C1 and L2 in the last nomenclature LE_{C1L2} refer to the stages used to estimate the local error. Now we substitute LE_{C1L2} and u^{n+1} into Eq. (4.3) to obtain the norm of the local error estimation. Considering the previous calculations and Eqs. (4.4) and (4.5), an adaptive time step controller is constructed, and it is denoted by ALNe3-C1L2. Repeating the same step as in the previous lines, another adaptive time step controller is obtained, and it is denoted by ALNe3-C1L3. The third possibility is when the local error is estimated based on the first and the third stages and the applied controller will be denoted by ALNe3-L2L3 which used in Susection 4.2.4.

B) The apdative CLL schemes

The CLL scheme [93] is a modification of the LNe3 algorithm in order to achieve third order temporal convergence. It consists of three stages. It uses fractional time steps during the first and second stages and a full time step in the third stage. Generally, the length at the first stage is $h_1 = ph$, $2/3 \le p < 2$, but at the second stage it is always $h_2 = 2h/3$. In the first stage, the CNe formula is employed to calculate new predictor values:

$$u_i^{\text{C}} = u_i^n e^{-pr_i^n} + \frac{A_i^n}{r_i^n}\left(1 - e^{-pr_i^n}\right). \tag{5.12}$$

In the second stage, we use formulas similar to Eq. (5.10), but with an $h_2 = 2h/3$ time step size to obtain the first corrector values. The new A_i^{new} values are calculated as in Eq. (5.9), i.e. $A_i^C = \frac{h}{C_i}\left(\frac{u_{i-1}^{\text{C}}}{R_{i,\,i-1}^n} + \frac{u_{i+1}^{\text{C}}}{R_{i,\,i+1}^n}\right)$. Using these the corrector step is as follows:

$$u_i^{n+1} = u_i^n e^{-2r_i^n/3} + \left(A_i^n - \frac{A_i^{\text{C}} - A_i^n}{p r_i^n}\right)\frac{1-e^{-2r_i^n/3}}{r_i^n} + \frac{A_i^{\text{C}} - A_i^n}{r_i^n}. \tag{5.13}$$

In the third stage, a full time step is taken with the LNe formula:

$$u_i^{n+1} = u_i^n e^{-r_i^n} + \left(A_i^n - \frac{A_i^{\text{CL}} - A_i^n}{2r_i^n/3}\right)\frac{1-e^{-r_i^n}}{r_i^n} + \frac{A_i^{\text{CL}} - A_i^n}{2r_i^n/3}, \tag{5.14}$$

where $A_i^{\text{CL}} = \frac{h}{C_i}\left(\frac{u_{i-1}^{\text{CL}}}{R_{i,\,i-1}^n} + \frac{u_{i+1}^{\text{CL}}}{R_{i,\,i+1}^n}\right)$.

If we take $p = 2/3$ in the first stage, then an error estimation can be made as in Eq. (5.11), where $\hat{u}^{n+1}$ is calculated by Eq. (5.12), considering that $p = 2/3$, while u^{n+1} is calculated by Eq. (5.13). Substituting Eq. (5.11) into Eq. (4.3), then considering Eqs. (4.4) and (4.5), an adaptive controller can be implemented, and it will be denoted by ACLL-C1L2. If $p = 1$, another local error estimation can be considered as follows:

$$LE_{C1L3} = \left|\hat{u}^{n+1} - u^{n+1}\right|, \tag{5.15}$$

where $\hat{u}^{n+1}$ is calculated by Eq. (5.12), taking $p = 1$, while u^{n+1} is calculated by Eq. (5.14). Substituting Eq. (5.15) into Eq. (4.3), then considering Eqs. (4.4) and (4.5), an adaptive controller will be implemented, and it will be denoted by ACLL-C1L3.

C) Runge-Kutta Cash-Karp Method RKCK

Since it is a well-known method, and explained in detail in [79, p. 717], I think that it is not necessary to describe the tedious processes of implementing the method. The local error estimation in Eq. (16.2.6) and fourth order solution in Eq. (16.2.5) in [79] can be plugged in our Eq. (4.3) to obtain the norm of the local error estimation. Again, using Eqs. (4.4) and (4.5), the Runge Kutta Cash-Karp is obtained, and it is denoted by RKCK.

D) Runge-Kutta-Fehlberg Method

Plenty of references discussed and implemented this method. Here I will refer to [94], where the authors show how to estimate the local error using Eq. (5.55) in that reference. The numerical solution generated by Eq. (5.53), along with the local error estimated by Eq. (5.55) in that reference, can be substituted into our Eq. (4.3) to obtain the norm of error estimation. That norm

can be used to adapt the time step size using Eq. (4.4), resulting in the so-called Runge-Kutta-Fehlberg 4(5) or RKF45 method and it will be referred to as RKF in our paper.

5.2. Numerical Experiments and Results

The numerical solution and the reference solution are compared only at t^{fin}, which is the final time of the simulation and will be specified later. We measure the accuracy using the global L_∞ error described in (4.29).

The command kummerU, in MATLAB, has been used to calculate the Kummer U function (confluent hypergeometric function of the second kind). Since the calculation of the values of the boundary conditions for a given time point is orders of magnitude more time-consuming than performing the steps of the numerical schemes for all the nodes of the grid, we applied a trick to minimize running time. The boundary conditions have been calculated only in 4000 time points, and linear interpolation between the two appropriate times of the pre-calculated boundary values has been used to evaluate the boundary conditions at the actual time of the simulation. Of course, we always checked that the error due to this approximation is always much smaller than the errors of the numerical algorithms at the intermediate space points.

Two numerical experiments are conducted to check the performance of these adaptive controllers and to compare their performances. The numerical computations are carried out using the MATLAB and desktop computer which were used in the experiments of Chapter 4.

5.2.1. Experiment 1

In this experiment, the following parameters are used:

$$m = 2.4,\ \alpha = 3.1,\ c = 5.96 \cdot 10^{-13},\ N = 1000, x_0 = 0.055,\ \Delta x = 3 \cdot 10^{-4},\ t^0 = 0.5,\ t^{\text{fin}} = 1.5. \quad (5.16)$$

The errors as a function of the running time are presented in a log-log diagram in Figure. 5.1. From the figure, it is evident that the adaptive LNe3 controllers and the adaptive CLL controllers are significantly faster than the RKF an RKCK when the desired accuracy is not very high. RKF and RKCK can achieve the same accuracy of the adaptive LNe and the adaptive CLL families with the same running time, only when the error is 1.4×10^{-6}. However, none of the adaptive controllers can go beyond this accuracy due to the space discretization error. It does indeed look like the error, in the case of the RKF and RKCK, is relatively independent of the running time. According to our previous experience, this is not uncommon behavior in the case of explicit adaptive solvers, if the method used for designing the controller is only conditionally stable, such as some of the built-in ODE solvers of MATLAB [95].

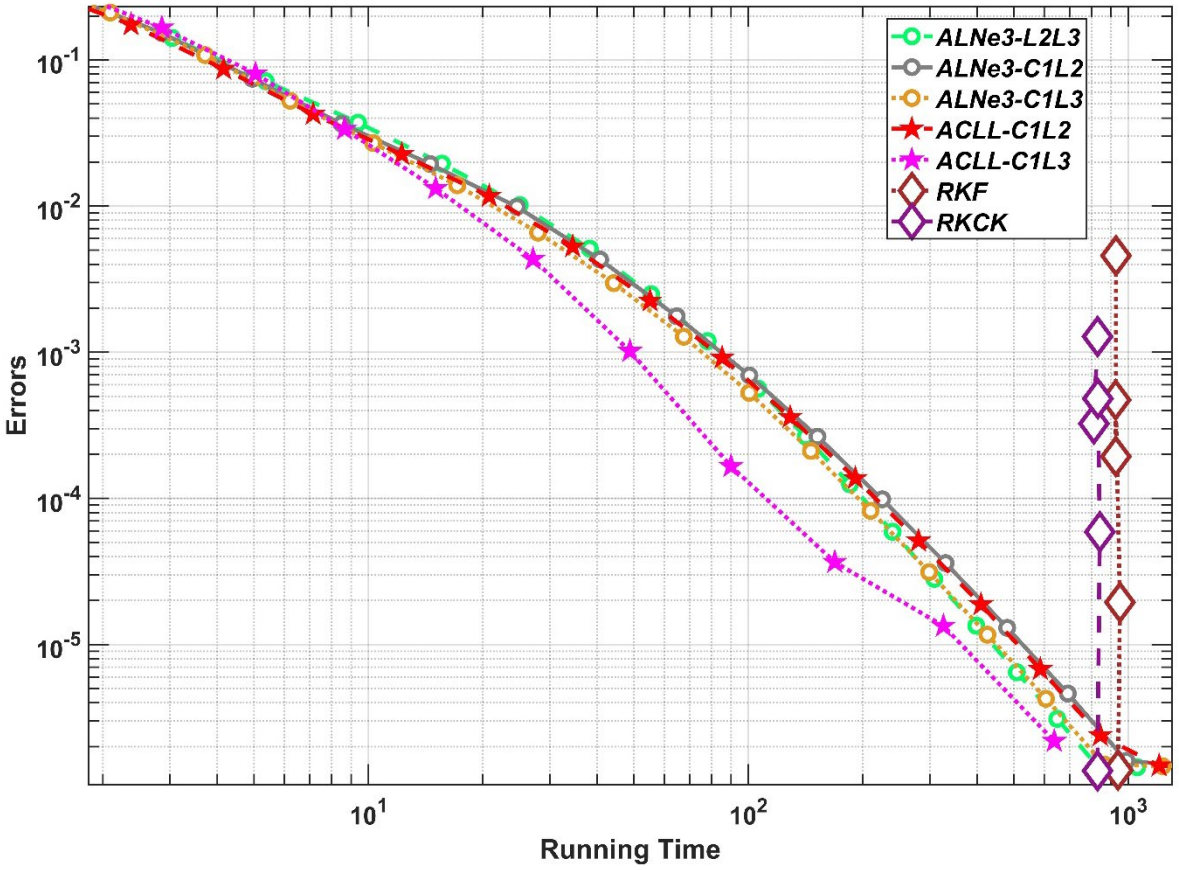

Figure. 5.1. Experiment 1: The L_∞ errors as a function of the running times

5.2.1. Experiment 2

In this experiment, the following parameters are used:

$$m = 7.2,\ \alpha = 11.4,\ c = 0.0042,\ N = 500, x_0 = 0.48,\ \Delta x = 5 \cdot 10^{-4},\ t^0 = 0.9,\ t^{\text{fin}} = 1.5.$$

Figure 5.2 shows the errors as a function of the running time in a log-log diagram. This experiment shows that the adaptive LNe controllers and the adaptive CNe controllers are again faster than controllers designed based on the Runge-Kutta method. As we mentioned previously, the CFL limit is changing with respect to time, and it can be calculated for the explicit Euler method as $h_{\text{CFL}}^{\text{EE}} = \left| 2 / \lambda_{\text{MAX}} \right|$. That limit was calculated in this experiment at six selected points in time as follows

$$\text{time point} \in \left\{ t^0 + i\left(t^{fin} - t^0 \right) \right\},\ i \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}.$$

I plotted this limit as a function of time with a dashed blue line in Figure 5.3. Approximately at the same level of accuracy, when the produced error was of order 10^{-4}, the history of the time step size was also registered for each adaptive controller in order to check if they can follow the trend

of the $h_{\mathrm{CFL}}^{\mathrm{EE}}$. Figure 5.3 shows that the LNe3 controllers and CLL controllers could roughly follow the trend of the CFL limit. It means that they could detect the changes in the CFL limit and modify the step size. The Runge-Kutta controllers could follow the general trend very well, but they suffer from fluctuating step size. The zoomed area of Figure 5.3 shows the behavior of the time-step size of RKCK during very short time (0.06% of the total time). On other hand, the time-step size in the case adaptive LNe3-L2L3 remained roughly constant. The reason behind the fluctuation in the case of the RK solvers is the conditional stability: when the time step size h is below the CFL limit (which is slightly larger for RK4 than for the first order explicit Euler), the error is very small and the time step size is increased. When the time step size exceeds the stability limit, errors are starting to be amplified exponentially. This exponential increase can be very slow at the beginning if h is still close to the limit, which may yield further time step size elevation. Once the increasing error is detected, h is suddenly decreased to let the errors diffuse away. Then the errors will be very small again, thus the cycle starts again. This fluctuation is time consuming and therefore undesirable. It is among the reasons why adaptive Runge-Kutta controllers are slower than other solvers in our experiments.

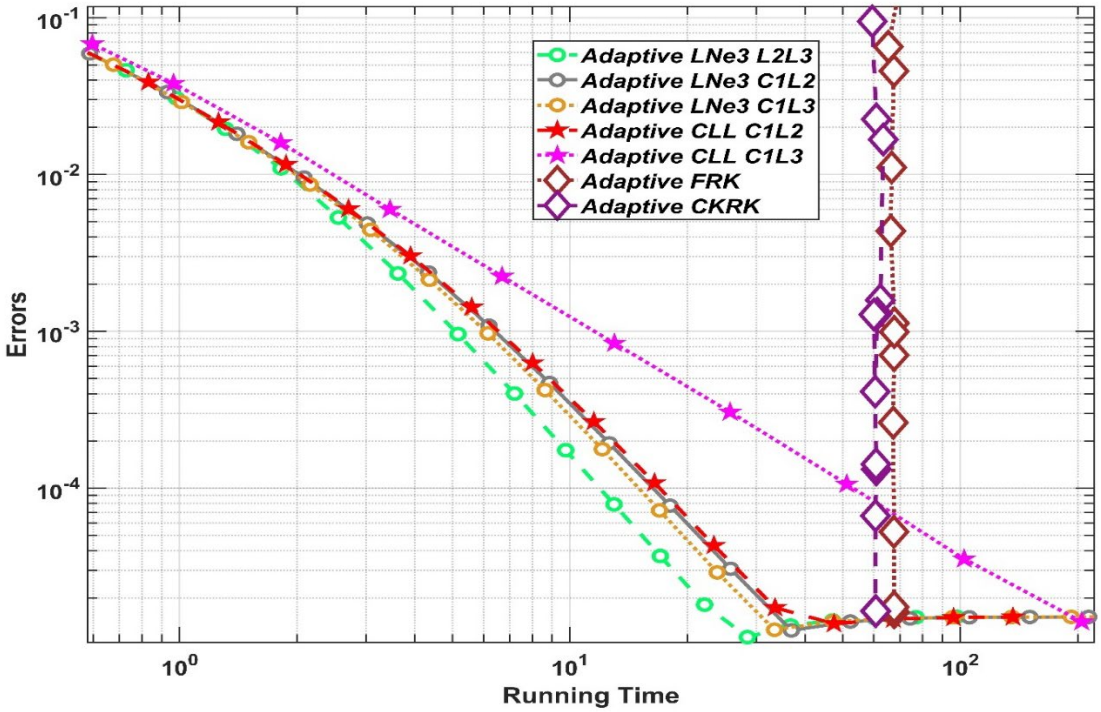

Figure 5.2. Experiment 2: The L_∞ errors as a function of the running times.

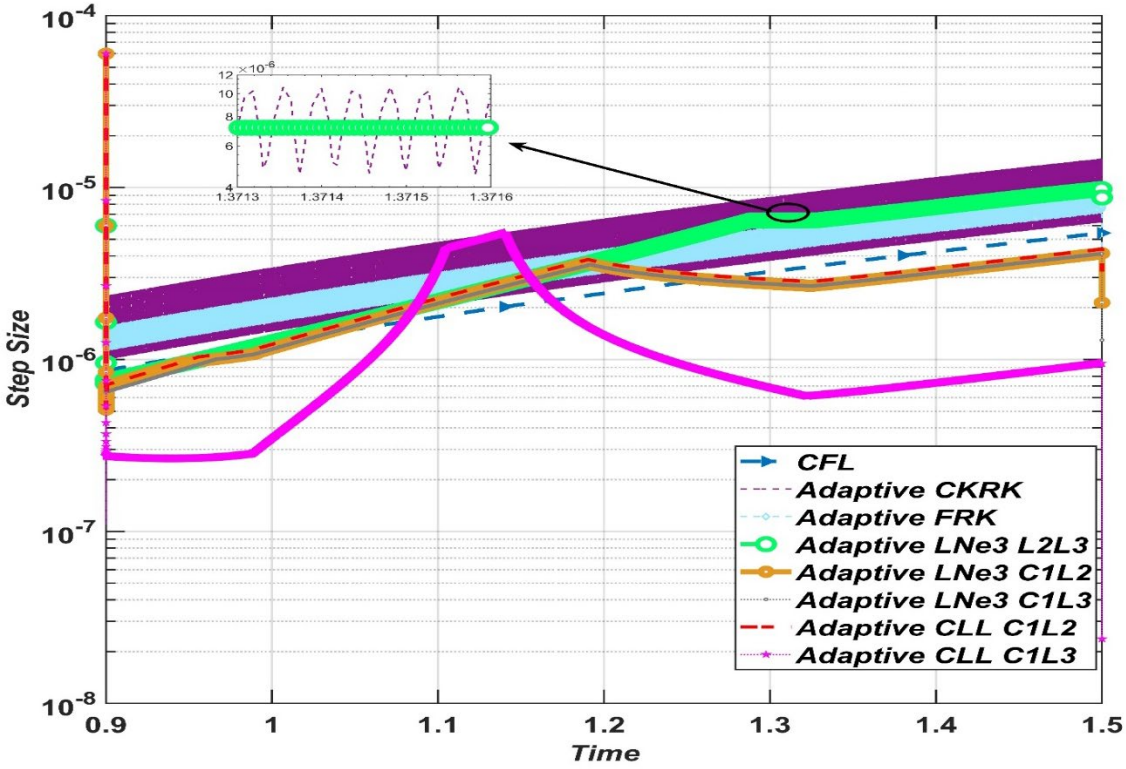

Figure 5.3. The time step size as a function time for the examined solvers.

6. THESES – NEW SCIENTIFIC RESULTS

T1. I constructed 24 schemes by combining conventional and non-conventional schemes within the odd-even hopscotch structure to obtain two stage methods. Then I produced preliminary numerical results and based on these I chose the 6 most efficient methods for further investigation. However, the original hopscotch method (A2) was one of these six methods. The performance of the selected methods was examined in the case of two 2-dimensional systems containing 10000 cells with very inhomogeneous randomly generated parameters and initial conditions. I showed that the proposed methods are competitive as they can give results with acceptable accuracy orders of magnitude faster than the well-optimized MATLAB routines.

T2. The results showed that our novel hopscotch-based methods B1, C4, C5, D4, D5 are faster than the original hopscotch method (A2) when they are applied to a linear system with relatively high stiffness ratio. However, B1 method has the best performance comparing to all selected methods, and its advantage becomes larger when the system becomes more stiff. Based on the different numerical results, I selected those two methods which were proven to have the most valuable properties, namely, the reversed hopscotch (B1) and the CNe-CNe hopscotch (D5) algorithms. Then I analytically proved that their stability is guaranteed for the linear diffusion equation and that their convergence is second order in the time step size.

T3. Our novel hopscotch-based methods were applied to Fisher's equation. The results showed that the performance of the original hopscotch method (A2) is very poor when it is compared to performances of our novel hopscotch-based methods. I could prove analytically, in case of the CNe-CNe hopscotch (D5) scheme, that the way I treated the nonlinear reaction term guaranteed that the values of the unkown varaible will remin in the unit interval if the intial values of that unknown are in unit interval, which in turn implies the positivity preserving property.

T4. I systematically designed and tested families of adaptive time step controllers, based on PI and I controllers, for solving a system of ODEs resulted from spatially discretized linear diffusion equation. Several studies claimed that the adaptive step controllers of type PI are better than I type for solving a system of ODEs. Those studies compared the two types of controllers when they are applied to a single ODE or a small system of ODEs, and in the literature, I could not find any study which compare them when they are applied to a big system of ODEs. However, our result showed that adaptive schemes using PI controller do not have any advantage compared to the same schemes using I controller when they are applied to a big system of ODEs resulted from discretizing the space variable in the linear diffusion equation.

T5. Using the linear neighbour LNe3 method, I designed a novel adaptive time step controller of type I and applied it to a system of ODEs resulted from spatially discretized linear diffusion equation in the absence and in the presence of the heat source. In both cases, I conducted the numerical experiments in inhomogeneous media with relatively high stiffness ratio. The result

showed that the adaptive LNe3 is much faster than the adaptive schemes of Runge-Kutta when high accuracy is not required. The adaptive Runge-Kutta schemes are faster at the level of accuracy which is not required in the engineering application.

T6. Using LNe3 and CLL methods, families of novel adaptive time step controllers of type I are constructed. I treated the non-steady-state linear diffusion equation, where the diffusion coefficient itself depends simultaneously on space and time. I discretized the space variable in that equation to obtain a system of ODEs, then I used our adaptive controller to solve that system of equations. The numerical experiments showed that the performances of our adaptive controllers severely outperform the widely used schemes, which are Fehlberg Runge-Kutta and Cash-Karp Runge-Kutta. Recall that a lot of efforts have been made to improve traditional solvers by using the so-called PI and PID controllers. The LNe3 and the CLL-based adaptive controllers could change the time-step size smoothly using only the elementary controller without any need to implement the PI controller. I consider this as another advantage of these methods.

7. SUMMARY

…

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my Supervisor, Doctor Endre Kovács, who introduced me to this intersting field of research. Thank you for your constant and patient help, encouragement, and excellent advice. Apart from research, I am glad to meet such a friendly and elegant person.

I would like also to thank my team mate Ádám Nagy and my friend Modar Wannous who helped me to overcome the difficulties related to programming. Without your help my task would be more complicated

Thanks for my team mates and colleagues, János Majár, Issa Omle, Kareem Humam, Habeeb Ali, Pszota Gabor, Tamás Jenyó who helped me in many different ways.

REFERENCES

[1] Frank P. Incropera, David P. DeWitt, Theodore L. Bergman, and Adrienne S. Lavine, *Fundamentals of Heat and Mass Transfer*, 6th ed. John Wiley & Sons, 2006.

[2] L. M. Jiji, *Heat Conduction*, 3rd ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-01267-9.

[3] P. Nithiarasu, R. W. Lewis, and K. N. Seetharamu, "Fundamentals of the Finite Element Method for Heat and Mass Transfer Second Edition."

[4] YUNUS A. ÇENGEL and AFSHIN J. GHAJAR, *H E A T A N D M A S S TRANSFER FUNDAMENTALS & APPLICATIONS*, 5th ed. New York: McGraw-Hill Education, 2015.

[5] D. W. Hahn and M. N. Özişik, *Heat Conduction*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2012. doi: 10.1002/9781118411285.

[6] D. W. Hahn and M. Necati. Özışık, *Heat conduction.*, 3rd ed. Wiley, 2012.

[7] H.-C. Huang and A. S. Usmani, *Finite Element Analysis for Heat Transfer*, 1st ed. London: Springer London, 1994. doi: 10.1007/978-1-4471-2091-9.

[8] C. Fradin, "On the importance of protein diffusion in biological systems: The example of the Bicoid morphogen gradient," *Biochimica et Biophysica Acta (BBA) - Proteins and Proteomics*, vol. 1865, no. 11, pp. 1676–1686, Nov. 2017, doi: 10.1016/j.bbapap.2017.09.002.

[9] David Fisher, *Defects and Diffusion in Carbon Nanotubes*. Trans Tech Publications, 2014.

[10] I. Pisarenko and E. Ryndin, "Numerical Drift-Diffusion Simulation of GaAs p-i-n and Schottky-Barrier Photodiodes for High-Speed AIIIBV On-Chip Optical Interconnections," *Electronics (Basel)*, vol. 5, no. 4, p. 52, Sep. 2016, doi: 10.3390/electronics5030052.

[11] Robert Zimmerman, *The Imperial College Lectures in Petroleum Engineering: Fluid Flow in Porous Media*. World Scientific Publishing Europe Ltd, 2018.

[12] H. Yu *et al.*, "The Moisture Diffusion Equation for Moisture Absorption of Multiphase Symmetrical Sandwich Structures," *Mathematics*, vol. 10, no. 15, p. 2669, Jul. 2022, doi: 10.3390/math10152669.

[13] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton, "Implicit-Explicit Methods for Time-Dependent Partial Differential Equations," *SIAM J Numer Anal*, vol. 32, no. 3, pp. 797–823, Jun. 1995, doi: 10.1137/0732037.

[14] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, 3rd ed. Clarendon Press, 1986.

[15] Jui-Ling Yu, "A FULLY EXPLICIT OPTIMAL TWO-STAGE NUMERICAL SCHEME FOR SOLVING REACTION-DIFFUSION-CHEMOTAXIS SYSTEMS," Michigan State University, 2005.

[16] D. A. Anderson, J. C. Tannehill, R. H. Pletcher, M. Ramakanth, and V. Shankar, *Computational Fluid Mechanics and Heat Transfer*. Fourth edition. | Boca Raton, FL : CRC Press, 2020. | Series: Computational and physical processes in mechanics and thermal sciences: CRC Press, 2020. doi: 10.1201/9781351124027.

[17] E. Kovács, Á. Nagy, and M. Saleh, "A Set of New Stable, Explicit, Second Order Schemes for the Non-Stationary Heat Conduction Equation," *Mathematics*, vol. 9, no. 18, p. 2284, Sep. 2021, doi: 10.3390/math9182284.

[18] P. O. Appau, O. K. Dankwa, and E. T. Brantson, "A comparative study between finite difference explicit and implicit method for predicting pressure distribution in a petroleum reservoir," *International Journal of Engineering, Science and Technology*, vol. 11, no. 4, pp. 23–40, Oct. 2019, doi: 10.4314/ijest.v11i4.3.

[19] A. Moncorgé, H. A. Tchelepi, and P. Jenny, "Modified sequential fully implicit scheme for compositional flow simulation," *J Comput Phys*, vol. 337, pp. 98–115, May 2017, doi: 10.1016/j.jcp.2017.02.032.

[20] A. Costa-Solé, E. Ruiz-Gironés, and J. Sarrate, "High-order hybridizable discontinuous Galerkin formulation with fully implicit temporal schemes for the simulation of two-phase flow through porous media," *Int J Numer Methods Eng*, vol. 122, no. 14, pp. 3583–3612, Jul. 2021, doi: 10.1002/nme.6674.

[21] M. Mascagni, "The Backward Euler Method for Numerical Solution of the Hodgkin–Huxley Equations of Nerve Conduction," *SIAM J Numer Anal*, vol. 27, no. 4, pp. 941–962, Aug. 1990, doi: 10.1137/0727054.

[22] S. Manaa and M. Sabawi, "Numerical Solution and Stability Analysis of Huxley Equation," *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 2, no. 1, pp. 85–97, Jun. 2005, doi: 10.33899/csmj.2005.164070.

[23] S. Y. Kadioglu and D. A. Knoll, "A fully second order implicit/explicit time integration technique for hydrodynamics plus nonlinear heat conduction problems," *J Comput Phys*, vol. 229, no. 9, pp. 3237–3249, May 2010, doi: 10.1016/j.jcp.2009.12.039.

[24] H. Chen, J. Kou, S. Sun, and T. Zhang, "Fully mass-conservative IMPES schemes for incompressible two-phase flow in porous media," *Comput Methods Appl Mech Eng*, vol. 350, pp. 641–663, Jun. 2019, doi: 10.1016/j.cma.2019.03.023.

[25] S. H. Lee, M. Ţene, S. Du, X. Wen, and Y. Efendiev, "A conservative sequential fully implicit method for compositional reservoir simulation," *J Comput Phys*, vol. 428, p. 109961, Mar. 2021, doi: 10.1016/j.jcp.2020.109961.

[26] F. Gagliardi, M. Moreto, M. Olivieri, and M. Valero, "The international race towards Exascale in Europe," *CCF Transactions on High Performance Computing*, vol. 1, no. 1, pp. 3–13, May 2019, doi: 10.1007/s42514-019-00002-y.

[27] I. Z. Reguly and G. R. Mudalige, "Productivity, performance, and portability for computational fluid dynamics applications," *Comput Fluids*, vol. 199, p. 104425, Mar. 2020, doi: 10.1016/j.compfluid.2020.104425.

[28] B. M. Chen-Charpentier and H. v. Kojouharov, "An unconditionally positivity preserving scheme for advection–diffusion reaction equations," *Math Comput Model*, vol. 57, no. 9–10, pp. 2177–2185, May 2013, doi: 10.1016/j.mcm.2011.05.005.

[29] G. F. Sun, G. R. Liu, and M. Li, "An Efficient Explicit Finite-Difference Scheme for Simulating Coupled Biomass Growth on Nutritive Substrates," *Math Probl Eng*, vol. 2015, pp. 1–17, 2015, doi: 10.1155/2015/708497.

[30] A. R. Appadu, "Performance of UPFD scheme under some different regimes of advection, diffusion and reaction," *Int J Numer Methods Heat Fluid Flow*, vol. 27, no. 7, pp. 1412–1429, Jul. 2017, doi: 10.1108/HFF-01-2016-0038.

[31] M. K. Kolev, M. N. Koleva, and L. G. Vulkov, "An Unconditional Positivity-Preserving Difference Scheme for Models of Cancer Migration and Invasion," *Mathematics*, vol. 10, no. 1, p. 131, Jan. 2022, doi: 10.3390/math10010131.

[32] A. Chertock and A. Kurganov, "High-Resolution Positivity and Asymptotic Preserving Numerical Methods for Chemotaxis and Related Models," 2019, pp. 109–148. doi: 10.1007/978-3-030-20297-2_4.

[33] P. Gordon, “Nonsymmetric Difference Equations,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 13, no. 3, pp. 667–673, Sep. 1965, doi: 10.1137/0113044.
[34] A. R. GOURLAY, “Hopscotch: a Fast Second-order Partial Differential Equation Solver,” *IMA J Appl Math*, vol. 6, no. 4, pp. 375–390, 1970, doi: 10.1093/imamat/6.4.375.
[35] H. Liu and S. Leung, “An alternating direction explicit method for time evolution equations with applications to fractional differential equations,” *Methods and Applications of Analysis*, vol. 26, no. 3, pp. 249–268, 2019, doi: 10.4310/MAA.2019.v26.n3.a3.
[36] J. H. Ferziger, M. Perić, and R. L. Street, *Computational Methods for Fluid Dynamics*. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-319-99693-6.
[37] E. Kovács, “A class of new stable, explicit methods to solve the non-stationary heat equation,” *Numer Methods Partial Differ Equ*, vol. 37, no. 3, pp. 2469–2489, May 2021, doi: 10.1002/num.22730.
[38] ENDRE KOVÁCS and ANDRÁS GILICZ, “New stable method to solve heat conduction problems in extremely large systems,” *Design of Machines and Structures*, vol. 8, pp. 30–38, Aug. 2019.
[39] E. Kovács, “New stable, explicit, first order method to solve the heat conduction equation,” *Journal of Computational and Applied Mechanics*, vol. 15, no. 1, pp. 3–13, 2020, doi: 10.32973/jcam.2020.001.
[40] B. M. Chen-Charpentier and H. v. Kojouharov, “An unconditionally positivity preserving scheme for advection–diffusion reaction equations,” *Math Comput Model*, vol. 57, no. 9–10, pp. 2177–2185, May 2013, doi: 10.1016/j.mcm.2011.05.005.
[41] P. Gordon, “Nonsymmetric Difference Equations,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 13, no. 3, pp. 667–673, Sep. 1965, doi: 10.1137/0113044.
[42] A. R. GOURLAY and G. R. MCGUIRE, “General Hopscotch Algorithm for the Numerical Solution of Partial Differential Equations,” *IMA J Appl Math*, vol. 7, no. 2, pp. 216–227, 1971, doi: 10.1093/imamat/7.2.216.
[43] A. R. GOURLAY, “Hopscotch: a Fast Second-order Partial Differential Equation Solver,” *IMA J Appl Math*, vol. 6, no. 4, pp. 375–390, 1970, doi: 10.1093/imamat/6.4.375.
[44] “Some recent methods for the numerical solution of time-dependent partial differential equations,” *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 323, no. 1553, pp. 219–235, Jun. 1971, doi: 10.1098/rspa.1971.0099.
[45] A. Al-Bayati, S. Manaa, and A. Al-Rozbayani, “Comparison of Finite Difference Solution Methods for Reaction Diffusion System in Two Dimensions,” *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 8, no. 1, pp. 21–36, Jul. 2011, doi: 10.33899/csmj.2011.163605.
[46] C. Harley, “Hopscotch method: The numerical solution of the Frank-Kamenetskii partial differential equation,” *Appl Math Comput*, vol. 217, no. 8, pp. 4065–4075, Dec. 2010, doi: 10.1016/j.amc.2010.10.020.
[47] J. H. M. ten Thije Boonkkamp, “The Odd-Even Hopscotch Pressure Correction Scheme for the Incompressible Navier–Stokes Equations,” *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 2, pp. 252–270, Mar. 1988, doi: 10.1137/0909016.
[48] J. H. M. ten Thije Boonkkamp and J. G. Verwer, “On the odd-even hopscotch scheme for the numerical integration of time-dependent partial differential equations,” *Applied Numerical Mathematics*, vol. 3, no. 1–2, pp. 183–193, May 1987, doi: 10.1016/0168-9274(87)90011-0.
[49] J. Xu, S. Shao, and H. Tang, “Numerical methods for nonlinear Dirac equation,” *J Comput Phys*, vol. 245, pp. 131–149, Jul. 2013, doi: 10.1016/j.jcp.2013.03.031.
[50] E. D. de Goede and J. H. M. ten Thije Boonkkamp, “Vectorization of the Odd–Even Hopscotch Scheme and the Alternating Direction Implicit Scheme for the Two-

Dimensional Burgers Equations," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 2, pp. 354–367, Mar. 1990, doi: 10.1137/0911021.

[51] S. Maritim, J. K. Rotich, and J. K. Bitok, "Hybrid hopscotch Crank-Nicholson-Du Fort and Frankel (HP-CN-DF) method for solving two dimensional system of Burgers' equation," *Applied Mathematical Sciences*, vol. 12, no. 19, pp. 935–949, 2018, doi: 10.12988/ams.2018.8798.

[52] Simeon Kiprono Maritim and John Kimutai Rotich, "Hybrid Hopscotch Method for Solving Two Dimensional System of Burgers' Equation," *International Journal of Science and Research (IJSR)*, vol. 9, no. 8, Aug. 2019.

[53] M. Saleh, Á. Nagy, and E. Kovács, "Construction and investigation of new numerical algorithms for the heat equation : Part 2," *Multidiszciplináris tudományok*, vol. 10, no. 4, pp. 339–348, 2020, doi: 10.35925/j.multi.2020.4.37.

[54] M. Saleh, Á. Nagy, and E. Kovács, "Construction and investigation of new numerical algorithms for the heat equation : Part 1," *Multidiszciplináris tudományok*, vol. 10, no. 4, pp. 323–338, 2020, doi: 10.35925/j.multi.2020.4.36.

[55] M. Saleh, Á. Nagy, and E. Kovács, "Construction and investigation of new numerical algorithms for the heat equation : Part 3," *Multidiszciplináris tudományok*, vol. 10, no. 4, pp. 349–360, 2020, doi: 10.35925/j.multi.2020.4.38.

[56] M. Saleh, E. Kovács, and Á. Nagy, "New stable, explicit, second order hopscotch methods for diffusion-type problems," *Math Comput Simul*, vol. 208, pp. 301–325, Jun. 2023, doi: 10.1016/j.matcom.2023.01.029.

[57] Mark H. Holmes, *Introduction to Numerical Methods in Differential Equations*, vol. 52. 2007.

[58] Y. Li, P. van Heijster, R. Marangell, and M. J. Simpson, "Travelling wave solutions in a negative nonlinear diffusion–reaction model," *J Math Biol*, vol. 81, no. 6–7, pp. 1495–1522, Dec. 2020, doi: 10.1007/s00285-020-01547-1.

[59] J. H. MERKIN, D. J. NEEDHAM, and S. K. SCOTT, "Coupled reaction-diffusion waves in an isothermal autocatalytic chemical system," *IMA J Appl Math*, vol. 50, no. 1, pp. 43–76, 1993, doi: 10.1093/imamat/50.1.43.

[60] D. Campos, J. E. Llebot, and J. Fort, "Reaction–diffusion pulses: a combustion model," *J Phys A Math Gen*, vol. 37, no. 26, pp. 6609–6621, Jul. 2004, doi: 10.1088/0305-4470/37/26/001.

[61] M. BASTANI and D. K. SALKUYEH, "A highly accurate method to solve Fisher's equation," *Pramana*, vol. 78, no. 3, pp. 335–346, Mar. 2012, doi: 10.1007/s12043-011-0243-8.

[62] K. M. Agbavon, A. R. Appadu, and M. Khumalo, "On the numerical solution of Fisher's equation with coefficient of diffusion term much smaller than coefficient of reaction term," *Adv Differ Equ*, vol. 2019, no. 1, p. 146, Dec. 2019, doi: 10.1186/s13662-019-2080-x.

[63] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal, *Fundamentals of Convex Analysis*. Berlin: Springer, 2001.

[64] J. H. M. ten Thije Boonkkamp and J. G. Verwer, "On the odd-even hopscotch scheme for the numerical integration of time-dependent partial differential equations," *Applied Numerical Mathematics*, vol. 3, no. 1–2, pp. 183–193, May 1987, doi: 10.1016/0168-9274(87)90011-0.

[65] "LinearAlgebra: Norm, MatrixNorm, VectorNorm." https://www.maplesoft.com/support/help/maple/view.aspx?path=LinearAlgebra/Norm. (accessed Feb. 19, 2023).

[66] N. A. Mbroh and J. B. Munyakazi, “A robust numerical scheme for singularly perturbed parabolic reaction-diffusion problems via the method of lines,” *Int J Comput Math*, vol. 99, no. 6, pp. 1139–1158, Jun. 2022, doi: 10.1080/00207160.2021.1954621.
[67] C. A. J. Fletcher, Ed., *Computational Techniques for Fluid Dynamics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988. doi: 10.1007/978-3-642-97071-9.
[68] M. Saleh, E. Kovács, and N. Kallur, “Adaptive step size controllers based on Runge-Kutta and linear-neighbor methods for solving the non-stationary heat conduction equation,” *Networks and Heterogeneous Media*, vol. 18, no. 3, pp. 1059–1082, 2023, doi: 10.3934/nhm.2023046.
[69] I. Fekete, S. Conde, and J. N. Shadid, “Embedded pairs for optimal explicit strong stability preserving Runge–Kutta methods,” *J Comput Appl Math*, vol. 412, p. 114325, Oct. 2022, doi: 10.1016/j.cam.2022.114325.
[70] L. F. Shampine, “Error estimation and control for ODEs,” *J Sci Comput*, vol. 25, no. 1–2, pp. 3–16, Nov. 2005, doi: 10.1007/bf02728979.
[71] L. F. Shampine and H. A. Watts, “Comparing Error Estimators for Runge-Kutta Methods,” *Math Comput*, vol. 25, no. 115, p. 445, Jul. 1971, doi: 10.2307/2005206.
[72] R. h. Merson, “An operational methods for study of integration processes,” 1957.
[73] L. F. Shampine, “Local Extrapolation in the Solution of Ordinary Differential Equations,” *Math Comput*, vol. 27, no. 121, p. 91, Jan. 1973, doi: 10.2307/2005249.
[74] J. C. Butcher and P. B. Johnston, “Estimating local truncation errors for Runge-Kutta methods,” *J Comput Appl Math*, vol. 45, no. 1–2, pp. 203–212, Apr. 1993, doi: 10.1016/0377-0427(93)90275-G.
[75] J. H. Verner, “Explicit Runge–Kutta Methods with Estimates of the Local Truncation Error,” *SIAM J Numer Anal*, vol. 15, no. 4, pp. 772–790, Aug. 1978, doi: 10.1137/0715051.
[76] R. England, “Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations,” *Comput J*, vol. 12, no. 2, pp. 166–170, Feb. 1969, doi: 10.1093/comjnl/12.2.166.
[77] A. S. Chai, “Error estimate of a fourth-order Runge-Kutta method with only one initial derivative evaluation,” in *Proceedings of the April 30--May 2, 1968, spring joint computer conference on - AFIPS ’68 (Spring)*, New York, New York, USA: ACM Press, 1968, p. 467. doi: 10.1145/1468075.1468144.
[78] R. E. Scraton, “Estimation of the truncation error in Runge-Kutta and allied processes,” *Comput J*, vol. 7, no. 3, pp. 246–248, Mar. 1964, doi: 10.1093/comjnl/7.3.246.
[79] William H. Press, *Numerical Recipes The Art of Scientific Computing*, vol. 3. Cambridge University Press, 2007. [Online]. Available: www.cambridge.org/9780521880688
[80] K. Jell Gustafsson, M. Lundh, G. St, and) Derlind, “A PI STEPSIZE CONTROL FOR THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS,” *BIT*, vol. 28, pp. 270–287, 1988.
[81] K. Gustafsson, “Control Theoretic Techniques for Stepsize Selection in Explicit Runge-Kutta Methods”.
[82] G. Söderlind, “Automatic control and adaptive time-stepping,” *Numer Algorithms*, vol. 31, pp. 281–310, 2002.
[83] G. Söderlind, “Digital Filters in Adaptive Time-Stepping,” 2003.
[84] G. Söderlind and L. Wang, “Adaptive time-stepping and computational stability,” *J Comput Appl Math*, vol. 185, no. 2, pp. 225–243, Jan. 2006, doi: 10.1016/j.cam.2005.03.008.
[85] T. Ritschel, “Numerical Methods For Solution of Differential Equations,” 2013.
[86] *Solving Ordinary Differential Equations I*, vol. 8. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993. doi: 10.1007/978-3-540-78862-1.

[87] S. Conde, I. Fekete, and J. N. Shadid, “Embedded error estimation and adaptive step-size control for optimal explicit strong stability preserving Runge--Kutta methods,” Jun. 2018, doi: 10.1016/j.cam.2022.114325.
[88] David F. Griffiths and Desmond J. Higham, *Numerical Methods for Ordinary Differential Equations*. Springer, 2010. [Online]. Available: www.springer.com/series/3423
[89] W. H. Press, *Numerical recipes in C : the art of scientific computing*. Cambridge University Press, 1992.
[90] J. Feldman, A. Rechnitzer, and E. Yeager, “CLP-2 INTEGRAL CALCULUS.” [Online]. Available: https://LibreTexts.org
[91] M. Saleh, E. Kovács, and I. F. Barna, “Analytical and Numerical Results for the Transient Diffusion Equation with Diffusion Coefficient Depending on Both Space and Time,” *Algorithms*, vol. 16, no. 4, p. 184, Mar. 2023, doi: 10.3390/a16040184.
[92] Frank W. Olver, Daniel W. Lozier, Ronald Boisvert, and Charles W. Clark, *The NIST Handbook of Mathematical Functions*, vol. 1. Cambridge University Press, New York, NY, 2010.
[93] E. Kovács, Á. Nagy, and M. Saleh, “A New Stable, Explicit, Third-Order Method for Diffusion-Type Problems,” *Adv Theory Simul*, vol. 5, no. 6, p. 2100600, Jun. 2022, doi: 10.1002/adts.202100600.
[94] K. E. Atkinson, W. Han, and D. Stewart, *Numerical Solution of Ordinary Differential Equations*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2009. doi: 10.1002/9781118164495.
[95] Á. Nagy, I. Omle, H. Kareem, E. Kovács, I. F. Barna, and G. Bognar, “Stable, Explicit, Leapfrog-Hopscotch Algorithms for the Diffusion Equation,” *Computation*, vol. 9, no. 8, p. 92, Aug. 2021, doi: 10.3390/computation9080092.

LIST OF PUBLICATIONS RELATED TO THE TOPIC OF THE RESEARCH FIELD

(1) M. Saleh, E. Kovács, and I. F. Barna, "Analytical and Numerical Results for the Transient Diffusion Equation with Diffusion Coefficient Depending on Both Space and Time," *Algorithms*, vol. 16, no. 4, p. 184, Mar. 2023, doi: 10.3390/a16040184.

(2) M. Saleh, E. Kovács, and N. Kallur, "Adaptive step size controllers based on Runge-Kutta and linear-neighbor methods for solving the non-stationary heat conduction equation," *Networks and Heterogeneous Media*, vol. 18, no. 3, pp. 1059–1082, 2023, doi: 10.3934/nhm.2023046.

(3) M. Saleh, E. Kovács, and Á. Nagy, "New stable, explicit, second order hopscotch methods for diffusion-type problems," *Math Comput Simul*, vol. 208, pp. 301–325, Jun. 2023, doi: 10.1016/j.matcom.2023.01.029.

(4) Endre Kovács, Mahmoud Saleh, Imre Ferenc Barna, László Mátyás, "New Analytical Results and Numerical Schemes for Irregular Diffusion Processes," DIFFUSION FUNDAMENTALS 35 pp. 1-15. , 15 p. (2022)

(5) M. Saleh, E. Kovács, I. F. Barna, and L. Mátyás, "New Analytical Results and Comparison of 14 Numerical Schemes for the Diffusion Equation with Space-Dependent Diffusion Coefficient," *Mathematics*, vol. 10, no. 15, p. 2813, Aug. 2022, doi: 10.3390/math10152813.

(6) E. Kovács, Á. Nagy, and M. Saleh, "A New Stable, Explicit, Third-Order Method for Diffusion-Type Problems," *Adv Theory Simul*, vol. 5, no. 6, p. 2100600, Jun. 2022, doi: 10.1002/adts.202100600.

(7) E. Kovács, Á. Nagy, and M. Saleh, "A Set of New Stable, Explicit, Second Order Schemes for the Non-Stationary Heat Conduction Equation," *Mathematics*, vol. 9, no. 18, p. 2284, Sep. 2021, doi: 10.3390/math9182284.

(8) M. Saleh and E. Kovács, "Drag coefficient calculation of modified Myring-Savonius wind turbine with numerical simulations," *Design of Machines and Structures*, vol. 10, no. 2, pp. 73–84, 2020, doi: 10.32972/dms.2020.017.

(9) Á. Nagy, M. Saleh, I. Omle, H. Kareem, and E. Kovács, "New Stable, Explicit, Shifted-Hopscotch Algorithms for the Heat Equation," *Mathematical and Computational Applications*, vol. 26, no. 3, p. 61, Aug. 2021, doi: 10.3390/mca26030061.

(10) M. Saleh, Á. Nagy, and E. Kovács, "Construction and investigation of new numerical algorithms for the heat equation: Part 1," *Multidiszciplináris tudományok*, vol. 10, no. 4, pp. 323–338, 2020, doi: 10.35925/j.multi.2020.4.36.

(11) M. Saleh, Á. Nagy, and E. Kovács, "Construction and investigation of new numerical algorithms for the heat equation: Part 2," *Multidiszciplináris tudományok*, vol. 10, no. 4, pp. 339–348, 2020, doi: 10.35925/j.multi.2020.4.37.

(12) M. Saleh, Á. Nagy, and E. Kovács, "Construction and investigation of new numerical algorithms for the heat equation: Part 3," *Multidiszciplináris tudományok*, vol. 10, no. 4, pp. 349–360, 2020, doi: 10.35925/j.multi.2020.4.38.

(13) M. Saleh, E. Kovács, and G. Pszota, "Testing and improving a non-conventional unconditionally positive finite difference method," *Multidiszciplináris tudományok*, vol. 10, no. 4, pp. 206–213, 2020, doi: 10.35925/j.multi.2020.4.24.

Under Review:

Endre Kovács, János Majár, Mahmoud Saleh, "Unconditionally positive, explicit, fourth order method for the diffusion- and Nagumo-type diffusion-reaction equations,". Submitted to the "Journal of Scientific Computing D1" on 27 Sep 2022.

APPENDICES