UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

CENTRO DE CIENCIAS BÁSICAS
DEPARTAMENTO DE SISTEMAS ELECTRÓNICOS

TESIS

A COMPUTATIONALLY EFFICIENT ALGORITHM TO SOLVE SOME SYSTEMS OF NONLINEAR FRACTIONAL PARTIAL DIFFERENTIAL EQUATIONS

PRESENTA

Jorge Eduardo Herrera Serrano

PARA OPTAR POR EL GRADO DE MAESTRO EN CIENCIAS EN COMPUTACIÓN

TUTOR

Dr. Jorge Eduardo Macías-Díaz

COMITÉ TUTORAL

Dr. Juan Manuel Gómez Reynoso (co-tutor)
Dr. José Antonio Guerrero Díaz de León (asesor)
Dr. Francisco Javier Álvarez Rodríguez (asesor)

Aguascalientes, Ags., 6 de febrero de 2020

DICTAMEN DE LIBERACIÓN ACADÉMICA PARA INICIAR LOS TRÁMITES DEL EXAMEN DE GRADO

UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES

POSGRADOS UAA

Fecha de dictaminación dd/mm/aa: 20/01/2020

NOMBRE: JORGE EDUARDO HERRERA SERRANO **ID** 138041

PROGRAMA: MAESTRÍA EN CIENCIAS CON OPCIONES A LA COMPUTACIÓN, MATEMÁTICAS APLICADAS **LGAC (del posgrado):** COMPUTACIÓN

TIPO DE TRABAJO: (X) Tesis ( ) Trabajo práctico

TITULO: A COMPUTATIONALLY EFFICIENT ALGORITHM TO SOLVE SOME SYSTEMS OF NONLINEAR FRACTIONAL PARTIAL DIFFERENTIAL EQUATIONS

IMPACTO SOCIAL (señalar el impacto logrado): ______

INDICAR SI/NO SEGÚN CORRESPONDA:

Elementos para la revisión académica del trabajo de tesis o trabajo práctico:

X	El trabajo es congruente con las LGAC del programa de posgrado
X	La problemática fue abordada desde un enfoque multidisciplinario
X	Existe coherencia, continuidad y orden lógico del tema central con cada apartado
X	Los resultados del trabajo dan respuesta a las preguntas de investigación o a la problemática que aborda
X	Los resultados presentados en el trabajo son de gran relevancia científica, tecnológica o profesional según el área
X	El trabajo demuestra más de una aportación original al conocimiento de su área
	Las aportaciones responden a los problemas prioritarios del país
X	Generó transferecia del conocimiento o tecnológica

El egresado cumple con lo siguiente:

X	Cumple con lo señalado por el Reglamento General de Docencia
X	Cumple con los requisitos señalados en el plan de estudios (créditos curriculares, optativos, actividades complementarias,estancia, predoctoral, etc)
X	Cuenta con los votos aprobatorios del comité tutoral, en caso de los posgrados profesionales si tiene solo tutorpodrá liberar solo el tutor
	Cuenta con la carta de satisfacción del Usuario
X	Coincide con el título y objetivo registrado
X	Tiene congruencia con cuerpos académicos
X	Tiene el CVU del Conacyt actualizado
	Tiene el artículo aceptado o publicado y cumple con los requisitos institucionales (en caso que proceda)

En caso de Tesis por artículos científicos publicados:

	Aceptación o Publicación de los articulos según el nivel del programa
	El estudiante es el primer autor
	El autor de correspondencia es el Tutor del Núcleo Académico Básico
	En los artículos se ven reflejados los objetivos de la tesis, ya que son producto de este trabao de investigación.
	Los artículos integran los capítulos de la tesis y se presentan en el idioma en que fueron publicados
	La aceptación o publicación de los artículos en revistas indexadas de alto impacto

Con base a estos criterios, se autoriza se continúen con los trámites de titulación y programación del examen de grado **Sí** X **No** ____

FIRMAS

Elaboró:

* NOMBRE Y FIRMA DEL CONSEJERO SEGÚN LA LGAC DE ADSCRIPCION: [illegible] JORGE EDUARDO MACÍAS DÍAZ

NOMBRE Y FIRMA DEL SECRETARIO TÉCNICO: [illegible] DR. HERMILO SÁNCHEZ CRUZ

* En caso de conflicto de intereses, firmará un revisor miembro del NAB de la LGAC correspondiente distinto al tutor o miembro del comité tutoral, asignado por el Decano.

Revisó:

NOMBRE Y FIRMA DEL SECRETARIO DE INVESTIGACIÓN Y POSGRADO: [illegible] DRA. HAYDEE MARTÍNEZ RUVALCABA

Autorizó:

NOMBRE Y FIRMA DEL DECANO: [illegible] M. en C. JORGE MARTÍN ALFEREZ CHÁVEZ

Nota: procede el trámite para el Depto. de Apoyo al Posgrado

En cumplimiento con el Art. 105C del Reglamento General de Docencia que a la letra señala entre las funciones del Consejo Académico: .... Cuidar la eficiencia terminal del programa de posgrado y el Art. 105F las funciones del Secretario Técnico, llevar el seguimiento de los alumnos.

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. Jorge Martín Alférez Chávez
DECANO DEL CENTRO DE CIENCIAS BÁSICAS
PRESENTE

Por medio de la presente, en mi calidad de tutor designado del estudiante **JORGE EDUARDO HERRERA SERRANO** con ID 138041 quien realizó la tesis titulada: **A COMPUTATIONALLY EFFICIENT ALGORITHM TO SOLVE SOME SYSTEMS OF NONLINEAR FRACTIONAL PARTIAL DIFFERENTIAL EQUATIONS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirla, y así continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y, sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE
"Se Lumen Proferre"
Aguascalientes, Ags., a 27 de enero de 2020

[illegible]

Dr. Jorge Eduardo Macías-Díaz

c.c.p.- Interesado
c.c.p.- Secretaría de Investigación y Posgrado
c.c.p.- Consejero Académico
c.c.p.- Minuta Secretario Técnico

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. Jorge Martín Alférez Chávez
DECANO DEL CENTRO DE CIENCIAS BÁSICAS
PRESENTE

Por medio de la presente, en mi calidad de sinodal designado del estudiante **JORGE EDUARDO HERRERA SERRANO** con ID 138041 quien realizó la tesis titulada: **A COMPUTATIONALLY EFFICIENT ALGORITHM TO SOLVE SOME SYSTEMS OF NONLINEAR FRACTIONAL PARTIAL DIFFERENTIAL EQUATIONS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirla, y así continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y, sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE
"Se Lumen Proferre"
Aguascalientes, Ags., a 27 de enero de 2020

[illegible]

Dr. Juan Manuel Gómez Reynoso

c.c.p.- Interesado
c.c.p.- Secretaría de Investigación y Posgrado
c.c.p.- Consejero Académico
c.c.p.- Minuta Secretario Técnico

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. Jorge Martín Alférez Chávez
DECANO DEL CENTRO DE CIENCIAS BÁSICAS
PRESENTE

Por medio de la presente, en mi calidad de sinodal designado del estudiante **JORGE EDUARDO HERRERA SERRANO** con ID 138041 quien realizó la tesis titulada: **A COMPUTATIONALLY EFFICIENT ALGORITHM TO SOLVE SOME SYSTEMS OF NONLINEAR FRACTIONAL PARTIAL DIFFERENTIAL EQUATIONS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirla, y así continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y, sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE
"Se Lumen Proferre"
Aguascalientes, Ags., a 27 de enero de 2020

[illegible]

Dr. José Antonio Guerrero Díaz de León

c.c.p.- Interesado
c.c.p.- Secretaría de Investigación y Posgrado
c.c.p.- Consejero Académico
c.c.p.- Minuta Secretario Técnico

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. Jorge Martín Alférez Chávez
DECANO DEL CENTRO DE CIENCIAS BÁSICAS
PRESENTE

Por medio de la presente, en mi calidad de sinodal designado del estudiante **JORGE EDUARDO HERRERA SERRANO** con ID 138041 quien realizó la tesis titulada: **A COMPUTATIONALLY EFFICIENT ALGORITHM TO SOLVE SOME SYSTEMS OF NONLINEAR FRACTIONAL PARTIAL DIFFERENTIAL EQUATIONS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirla, y así continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y, sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE
"Se Lumen Proferre"
Aguascalientes, Ags., a 27 de enero de 2020

[illegible]

Dr. Francisco Javier Álvarez Rodríguez

c.c.p.- Interesado
c.c.p.- Secretaría de Investigación y Posgrado
c.c.p.- Consejero Académico
c.c.p.- Minuta Secretario Técnico

Acknowledgments

First of all, I want to thank my tutors, Dr. Jorge Eduardo Macías-Díaz and Dr. Juan Manuel Gómez Reynoso because they were the main pillars for completing this thesis. To both of them I thank you very much for your dedication, support and effort that you put into making this research going forward. Without your help I would not have been unable to finish it. I am also grateful that, despite the circumstances, they did not abandon me and for keep working with me. I will take all your advice and words of encouragement with me forever. Thank you for all you did for me.

I also always want to thank my parents for supporting me and being there in those difficult times, my brothers for always encouraging me to continue and, above all, I want to thank my wife and daughter because they were always the main reasons for me to giving my best. Thank you all for being my motivation.

I also want to thank CONACYT and the Autonomous University of Aguascalientes for giving me this great opportunity to study a master's degree. . . .

Jorge Eduardo Herrera Serrano

Contents

List of Tables

List of Figures

Resumen

Es bien sabido que la simulación de sistemas fraccionarios es una tarea difícil desde todos los puntos de vista. En particular, la implementación por computadora de algoritmos numéricos para simular sistemas fraccionarios de ecuaciones diferenciales parciales en tres dimensiones es una tarea difícil que no se ha resuelto satisfactoriamente. En este trabajo, proponemos un método numérico para resolver sistemas de ecuaciones diferenciales parciales hiperbólicas (fraccionarias o no fraccionarias) que generalizan varios modelos conocidos de física, química y biología. El esquema es una técnica explícita que tiene la ventaja de ser fácil de implementar para cualquier científico con un conocimiento mínimo sobre programación científica. Proponemos una implementación eficiente que explota las ventajas del álgebra matricial ya disponibles en Fortran y otros lenguajes. El algoritmo se presenta matemáticamente y también en pseudocódigo, y en el apéndice se proporciona una implementación básica en Fortran del algoritmo informático. Este código es susceptible de ser compilado en paralelo usando OpenMP, de donde se deduce que el tiempo de la computadora se puede reducir sustancialmente. Como aplicación, proporcionamos algunas simulaciones ilustrativas sobre la formación de patrones de Turing en un sistema tridimensional de sustancias inhibidoras-activadoras. Los gráficos se obtuvieron utilizando funciones de Matlab con las salidas numéricas generadas por nuestro código Fortran. En el método computacional, se programó el método numérico obtenido previamente. Un total de 3 diferentes algoritmos fueron desarrollados para obtener el mejor de ellos, en cuanto a tiempo de ejecución. Una vez que el resultado fue favorable, se procedió a realizar un sistema completo, el cual contiene una interfaz gráfica para manipular los parámetros del algoritmo de manera más fácil. El sistema fue probado bajo una rúbrica de evaluación y se diseñó un nuevo sistema a partir de los resultados obtenidos en la evaluación.

Abstract

It is well know that the simulation of fractional systems is a difficult task from all points of view. In particular, the computer implementation of numerical algorithms to simulate fractional systems of partial differential equations in three dimensions is a hard task which has no been solved satisfactorily. In this work, we propose a numerical method to solve systems of hyperbolic (fractional o non-fractional) partial differential equations that generalize various known models from physics, chemistry and biology. The scheme is an explicit technique which has the advantage of being easy to implement for any scientist with minimal knowledge on scientific programming. We propose a computer implementation which exploits the advantages of the efficient matrix algebra already available in Fortran and other languages. The algorithm is presented mathematically as well as in pseudo-code, and a raw Fortran implementation of the computer algorithm is provided in the appendix. This code is susceptible to be compiled in parallel using OpenMP, whence it follows that the computer time can be substantially reduced. As application, we provide some illustrative simulations on the formation of Turing patterns in a three-dimensional system of inhibitor-activator substances in physics. The graphs were obtained using functions of Matlab with the numerical outputs generated by our Fortran code. In the computational method, the previously obtained numeric method was programmed. A total of 3 different algorithms were developed to get the best of them, in terms of runtime. Once the result was favorable, a complete system was carried out, in which it would contain a graphical interface to manipulate the algorithm parameters more easily. The system was tested under an evaluation rubric and a new system was designed based on the results obtained in the evaluation.

Introduction

Background

The investigation on the conditions under which Turing patterns appear in physical systems has been a highly transited avenue of research from the mathematical, the numerical and the physical points of view. It is well know that many nonlinear systems exhibit Turing patterns under suitable conditions on the parameters of the model and the initial data. Some diffusion-reaction systems exhibit the presence of Turing patterns, like some coupled systems which describe the interaction between inhibitor and activator substances in chemistry [1, 2], especially in chemical reactions with chlorine dioxide, iodine, and malonic acid [3]. In fact, several different Turing patterns are also found in the chlorine-iodide-malonic acid reaction, including hexagonal and striped [4], and oscillatory structures [5]. Outside the chemical sciences, there is also evidence of the presence of this kind of nonlinear behavior. For instance, there is experimental evidence on the presence of azimuthal Turing patterns in Kerr combs generated by whispering-gallery-mode resonators [6], and the theory suggests that diffusion-reaction systems may be used to understand the formation of this type of patterns in biology [7]. Even the labyrinthine structure of the cerebral cortex has been identified as a complex three-dimensional Turing pattern [8], among other applications [9]. Moreover, it is well known that the Brusselator is a system which exhibits the presence of that type of structures [10, 11].

The physical models described in the paragraph above are diffusive equations. It is well known that these systems have the physical limitation that any perturbation is instantaneously propagated throughout the system. For that reason, hyperbolic forms of these equations are physically preferred. It turns out that a wide variety of Turing patterns have been found also in hyperbolic systems. For example, Turing instabilities have been investigated in diffusive predator-prey models with hyperbolic mortality [12, 13], in hyperbolic models for locally interacting cell systems [14], in the hyperbolic chaos of standing-wave patterns generated by a modulated pump source [15] and in the spreading of infectious diseases in hyperbolic susceptible-infected-removed models [16]. Also, the presence of various wave and Turing patterns has been studied in the context of hyperbolic forms of the Brusselator [17, 18], in some hyperbolic models for self-organized biological aggregations and movement [19], in network systems through collective patterns and single differentiated nodes [20], in predator-prey reaction-diffusion models with spatio-temporal delays [21] and in hyperbolic vegetation models for semiarid environments [22], just to mention some systems.

In recent years, fractional derivatives have been introduced to mathematical models in order to provide more realistic descriptions of the physical phenomena. For instance, many fractional systems have been obtained as the continuous limit of discrete systems of particles with long-range interactions [23, 24]. However, independently of that, fractional derivatives have been successfully used in the theory of viscoelasticity [25], the theory of thermoelasticity [26], financial problems under a continuous time frame [27], self-similar protein dynamics [28] and quantum mechanics [29]. Moreover, some distributed-order fractional diffusion-wave equations are used in the modeling of groundwater flow to and from wells [30, 31]. As expected, the complexity of fractional problems is considerably higher than that of integer-order models, whence the need to design reliable numerical techniques to approximate the solutions is pragmatically justified [32]. In this direction, the literature reports on various methods to approximate the solutions of fractional systems. For example, some numerical methods have been proposed to solve fractional partial differential equations using fractional centered differences [33, 34], the time-fractional diffusion equation [35], the fractional Schrödinger equation in multiple spatial dimensions [36], the nonlinear fractional Korteweg–de Vries–Burgers equation [37], the fractional FitzHugh–Nagumo monodomain model in two spatial dimensions [38], distributed-order time-fractional diffusion-wave equations on bounded domains [39] and some Hamiltonian hyperbolic fractional differential equations that generalize various well-known equations from quantum field theory [40].

In light of these facts, the investigation on the presence of Turing patterns in fractional systems (in both the parabolic and hyperbolic types) has been also an interesting topic of research in recent years. Some reports have studied the presence of these patterns in simple models with fractional diffusion [41], in fractional reaction-diffusion systems with indices of different order [42] and with multiple homogeneous states [43], in hyperbolic inhibitor-activator systems with fractional diffusion [44], in two-species fractional reaction-diffusion systems [45] and in reaction models with sub-diffusion [46]. At the same time, various numerical methods have been proposed in the literature to investigate complex nonlinear models [47, 48, 49]. However, it is worth pointing out that most of the computational methods proposed in the literature are computationally slow and highly demanding [50]. This is a consequence of the fact that the calculation of fractional derivatives require the use of global information of the system at each time step [51, 52, 53, 54]. The present thesis is intended to alleviate partially such shortcomings. Moreover, it is important to point out that, as opposed to some previous efforts [33, 40, 55], the present approach is not Hamiltonian and considers a family consisting of more than one hyperbolic partial differential equation. As a consequence, various physical systems can be simulated using the numerical technique reported in this thesis.

Despite the great advances and discoveries that have been made in all areas of science, there are still existing problems that need to be solved, or in some cases, generate better solutions. In the area of mathematics intersecting with the area of researching environmental issues there are many problems that has to be addressed. One of such problems requires the use of Turing Patterns, which demand intensive and high-volume computational power. Therefore, this area presents the opportunity to conduct a new research.

The present research intends to solve this issue. For this purpose, an algorithm was developed that solves the numerical method previously proposed, which consists of a chemical model of activator and inhibitory substances. However, in order to find the optimal solution, it was considered necessary to implement such algorithm by developing a software application. In order to do this, three different approaches were developed that are capable of executing the proposed numerical method. A first solution was developed using Fortran as programming language, which was able to solve the numerical method, however, it has a considerable weakness: the response time required for each iteration was too large, to the extent of having to wait more than two days for heavy sets (100,000 iterations). This solution uses the traditional sequential programming approach. Thus, it was necessary to search for an improved solution. We found that the programmed algorithm was possible to be parallelized. Thus, the new solution proved to be a lot faster than its predecessor (about 300 times faster). However, we considered necessary to search for an enhanced solution. Finally, we discovered that the cycles nested within the algorithm could be changed by using matrix algebra. Hence, this solution turned out to be the best amongst the three performed for the present research and, consequently, the final selection.

The implemented solution was efficient; however, it was difficult to manipulate for inexperienced or not savvy users in the area of computer programming. For instance, if an end- user wants to study a particular data set, he must to modify that source code, which not all end- users are capable of performing. Therefore, a graphical user interface environment that integrates the algorithm's final solution was developed. This approach allows end-user to manipulate the desired input values for the application. In addition, this would address the mains software engineering requirements that any software system must be easy to use, easy to understand and easy to learn. Therefore, both the graphical interface and the algorithm would now work together under the same scheme and the result would be a complete software system.

The development of the system for the present research was performed based on a software methodology known as Rapid Application Development (RAD), which allows perfectly to understand and discover the requirements. Thus, developers are able to develop an ideal software solution.

After completing the development of the system, a set of tests we performed in order to understand whether end-user requirements are really met. The tests were performed using a questionnaire developed and tested in a previous research [56]. In addition to the questionnaire based on standards by the International Standard Organization (ISO), which describes about the quality factors that any software system must comply to. After the tests were completed, the results were interpreted to discover deficiencies in the software application, their possible causes and their corresponding corrective actions. After completing the tests, a new and enhanced version was developed that addresses all issues discovered during the testing phase. Finally, it is important to mention that the first implementation demanded approximately 48 hours to process a case of 100,000 iterations; which was reduced to just about 9 minutes in the second version; to finally, using about only 2 minutes for the final version. In addition, of all the quality variables studied, it was found that the factors of understanding and attractiveness had the minimum accepted quality (90%) value, and that the usability, operability and learning factors needed to be modified to achieve get the minimum. With the results obtained, it can be said that the solution is efficient at the moment, but perhaps in the future it could be further improved and perhaps develop an optimal solution (if there is).

Finally, it is important to mention that the first implementation took approximately 48 hours to process a case of 100,000 iterations; which was reduced to 9 minutes in the second version, to finally require only 2 minutes. In addition, of all the quality variables studied, it was found that the factors of understanding and attractivity had the minimum accepted quality (90%), and that the usability, operability and learning factors needed to be modified to achieve get the minimum. With the results obtained, it can be said that the solution is efficient at the moment, but perhaps in the future it could be further improved and perhaps get an optimal solution (if any).

Aims and scope

The purpose of this thesis is to propose a parallel (and, thus, computationally economic [57]) methodology to investigate general coupled hyperbolic systems with Riesz space-fractional diffusion. In order to generalize our physical model as much as possible, the system investigated in this thesis will consider an arbitrary finite number p of spatial dimensions, as well as an arbitrary finite number q of dependent variables. We will employ fractional centered differences to approximate the fractional partial derivatives of the model in view of their mathematical and computational simplicity, and an explicit four-step finite-difference discretization of our physical model will be proposed. Obviously, the explicit nature of the scheme will is an advantage when considering a high number of spatial variables. We will establish rigorously the main numerical features of our scheme, namely, its second-order consistency, stability, boundedness and quadratic convergence. The computational implementation of the scheme will be carried out using a convenient reformulation of the numerical method using vector algebra when approximating the fractional derivatives. Various computer languages have efficient implementations of vector algebra operations, including Fortran, C++ and Matlab. Our implementation is carried out using Gfortran on a Linux operating system, and we will increase the speed of the calculations using the multi-processing package `OpenMP`. As an illustrative application of our computer code, we will exhibit the presence of Turing patterns in some two- and three-dimensional inhibitor-activator systems. However, we will note that the methodology proposed in this thesis can be applied to a wide variety of physical models.

Document organization

This thesis is sectioned as follows.

- In Chapter 1, we consider a general multidimensional system of hyperbolic partial differential equations with fractional diffusion of the Riesz type, constant damping and coupled nonlinear reaction terms. The system generalizes many particular models from the physical sciences (including inhibitor-activator models in chemistry, diffusive nonlinear systems in population dynamics and relativistic wave equations), and considers the presence of an arbitrary number of both spatial dimensions and dependent variables. Motivated by the wide range of applications, we propose an explicit four-step finite-difference methodology to approximate the solutions of the continuous system. The properties of stability, boundedness and convergence of the scheme are proved rigorously using a discrete form of the fractional energy method.

- Chapter 2 describes efficient computational implementations of the finite-difference scheme presented in Chapter 1. It is important to recall that algorithms for space-fractional systems are computationally highly demanding. To alleviate this problem, a parallel implementation of our scheme is proposed using a vector reformulation of the numerical method. We provide some illustrative simulations of the formation of complex patterns in the two-dimensional scenario, and even in the computationally intense three-dimensional case. For the sake of convenience, an algorithmic presentation of our computational model is provided therein. Moreover, a simplified form of the numerical method is proposed, and some simulations using that method are provided in that stage.

- In Chapter 3, the first section presents an introduction to the types of existing software that can be found nowadays, as an introduction for the software developed as a product of this thesis. The following subtopics show the different versions of the algorithms used throughout this work, explaining the significant differences of each of them and, in turn, proposing the best algorithm due to its qualities in terms of the time of execution. In addition, these subtopics include the programming approaches used for each algorithm, as well as their advantages in terms of processing and runtime.

- Chapter 4 introduces the development of the software for the present research. First, the software methodology used for system development is presented. After that, it is shown the work performed for designing a set of the graphical user interfaces and how they and the algorithm interact with each other to work under the same scope for the end-user. Finally, a set of screenshots are shown for the different interfaces included in the application as well as its corresponding explanation of how it works are included.

- In the last chapter of the thesis, the performed evaluation of the developed application is presented. For this, a previously tested questionnaire based on ISO standards was used. The chapter shows some tables with the statistical results obtained, as well as their interpretation and some improvements included for the final version of the application developed. Finally, some screenshots are shown comparing the previous version with the final version of the application. In addition, the improvements were made based on the obtained evaluation results.

1. Mathematical models

This chapter is organized as follows. In Section 1.1, we provide the definition of Riesz partial fractional partial derivative, and introduce the mathematical system of interest. We note therein that various models from physics are particular cases of our model, and we close that stage providing a vector form of the mathematical model, considering some parameter simplifications. In Section 1.2, we introduce the discrete nomenclature and operators which will be used throughout this manuscript, including the concept of fractional centered differences, which is crucial in our investigation. An explicit four-step finite-difference scheme to solve the continuous problem of interest is introduced in that stage. The discrete model is then analyzed rigorously in Section 1.3. Concretely, we establish that the scheme is consistent of the second order, and we use a discrete form of the fractional energy method to show that the model is stable and quadratically convergent under appropriate parameter conditions.

1.1 Mathematical model

Throughout this thesis, we will use the symbol $\mathcal{I}_n$ to represent the set of indexes $\{1, \ldots, n\}$, and define $\overline{\mathcal{I}}_n = \mathcal{I}_n \cup \{0\}$, for each $n \in \mathbb{N}$. Moreover, if $x \in \mathbb{R}^p$ then we convey that $x = (x_1, \ldots, x_p)$, for any $p \in \mathbb{N}$. The present section is devoted to introduce the continuous nomenclature and the mathematical model under investigation in this manuscript. In particular, the following concepts are standard in the literature. They are the cornerstone to describe our continuous model, and they are recalled here for the sake of convenience. Throughout, Γ denotes the usual Gamma function.

Definition 1.1 (Podlubny [58]). Let $f : \mathbb{R} \to \mathbb{R}$ be a function, and let $n \in \mathbb{N} \cup \{0\}$ and $\alpha \in \mathbb{R}$ satisfy $n - 1 < \alpha < n$. The *Riesz fractional derivative* of f of order α at $x \in \mathbb{R}$ is defined (when it exists) as

$$\frac{d^\alpha f(x)}{d|x|^\alpha} = \frac{-1}{2\cos(\frac{\pi\alpha}{2})\Gamma(n-\alpha)} \frac{d^n}{dx^n} \int_{-\infty}^{\infty} \frac{f(\xi)d\xi}{|x-\xi|^{\alpha+1-n}}. \tag{1.1}$$

Definition 1.2 (Podlubny [58]). Assume that $p \in \mathbb{N}$ and $i \in \mathcal{I}_p$. Let $\alpha \in \mathbb{R}$ and $n \in \mathbb{N}$ satisfy $n - 1 < \alpha < n$, and suppose that $u : \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}$ is a function. Then the *left* and the *right Riemann–Liouville fractional partial derivative* of u of order α with respect to x_i at the point $(x, t) \in \mathbb{R}^p \times \mathbb{R}$ are defined, respectively, by

$$_{-\infty}D^\alpha_{x_i} u(x,t) = \frac{1}{\Gamma(n-\alpha)} \frac{\partial^n}{\partial x_i^n} \int_{-\infty}^{x} (x_i - \xi)^{n-1-\alpha} u(x_1, \ldots, x_{i-1}, \xi, x_{i+1}, \ldots, x_p, t) d\xi, \tag{1.2}$$

$$_{x_i}D^\alpha_{\infty} u(x,t) = \frac{(-1)^n}{\Gamma(n-\alpha)} \frac{\partial^n}{\partial x_i^n} \int_{x}^{\infty} (\xi - x_i)^{n-1-\alpha} u(x_1, \ldots, x_{i-1}, \xi, x_{i+1}, \ldots, x_p, t) d\xi, \tag{1.3}$$

whenever they exist. For the remainder of this thesis, we will consider differentiation orders satisfying $1 < \alpha < 2$. In that case, we define the *Riesz partial fractional derivative* of u of order α with respect to x_i at the point (x, t) as

$$\frac{\partial^\alpha u(x,t)}{\partial |x_i|^\alpha} = -\frac{1}{2\cos(\pi\alpha/2)} \left({}_aD^\alpha_{x_i} u(x,t) + {}_{x_i}D^\alpha_b u(x,t) \right). \tag{1.4}$$

Moreover, for convenience, we will agree that the Riesz partial fractional derivative of u of order 1 and 2 with respect to x_i coincide with the usual first- and second-order partial derivative of u with respect to x_i, respectively. Finally, the *Riesz fractional Laplacian* of u of order α at (x, t) will be defined as

$$\Delta^\alpha u(x,t) = \sum_{i=1}^{p} \frac{\partial^\alpha u(x,t)}{\partial |x_i|^\alpha}. \tag{1.5}$$

For the remainder of this paper, we let $p \in \mathbb{N}$ and assume that $a_i, b_i \in \mathbb{R}$ satisfy $a_i < b_i$, for each $i \in \mathcal{I}_p$. Let $T \in \mathbb{R}^+$, and fix the spatial and the spatial-temporal domains

$$B = \prod_{i=1}^{p} (a_i, b_i), \qquad \Omega = B \times (0, T), \tag{1.6}$$

respectively. Obviously, the sets B and Ω are open in $\mathbb{R}^{p+1}$, their respective closures will be denoted by $\overline{B}$ and $\overline{\Omega}$, and we will use the notation ∂B to represent the boundary of B. Also, we will fix $q \in \mathbb{N}$, and we will suppose that $u_j : \overline{\Omega} \to \mathbb{R}$ is a function, for each $j \in \mathcal{I}_q$. For convenience, we will extend the definition of u_j to all the space $\mathbb{R}^p \times [0, T]$, by letting $u_j(x, t) = 0$ for each $(x, t) \in (\mathbb{R}^p \setminus B) \times [0, T]$ and $j \in \mathcal{I}_q$.

Let $F_j : \mathbb{R}^q \to \mathbb{R}$ be a function for each $j \in \mathcal{I}_q$, and $u : \overline{\Omega} \to \mathbb{R}^q$ by $u(x, t) = (u_1(x, t), \ldots, u_q(x, t))$, for each $(x, t) \in \overline{\Omega}$. Moreover, we will agree that γ_j is a nonnegative constant, τ_j and d_j are positive, $\alpha_j \in (1, 2]$ and $\phi_j, \psi_j : B \to \mathbb{R}$ are smooth functions, for each $j \in \mathcal{I}_q$. We will investigate computationally the solution of the following problem governed by a system of hyperbolic space-fractional partial differential equations with coupled nonlinear reaction terms:

$$\begin{aligned} &\tau_j \frac{\partial^2 u_j(x,t)}{\partial t^2} + \gamma_j \frac{\partial u_j(x,t)}{\partial t} = d_j \Delta^{\alpha_j} u_j(x,t) - F_j(u(x,t)), \qquad \forall (x,t) \in \Omega,\ \forall j \in \mathcal{I}_q, \\ &\qquad \text{such that } \begin{cases} u_j(x,0) = \phi_j(x), & \forall x \in B,\ \forall j \in \mathcal{I}_q, \\ \dfrac{\partial u_j(x,0)}{\partial t} = \psi_j(x), & \forall x \in B,\ \forall j \in \mathcal{I}_q, \\ u_j(x,t) = 0, & \forall (x,t) \in \partial B \times [0,T],\ \forall j \in \mathcal{I}_q. \end{cases} \end{aligned} \tag{1.7}$$

More precisely, the system of partial differential equations in (1.7) can be explicitly described as

$$\begin{cases} \tau_1 \dfrac{\partial^2 u_1(x,t)}{\partial t^2} + \gamma_1 \dfrac{\partial u_1(x,t)}{\partial t} = d_1 \Delta^{\alpha_1} u_1(x,t) - F_1(u_1(x,t), u_2(x,t), \ldots, u_q(x,t)), & \forall (x,t) \in \Omega, \\ \tau_2 \dfrac{\partial^2 u_2(x,t)}{\partial t^2} + \gamma_2 \dfrac{\partial u_2(x,t)}{\partial t} = d_2 \Delta^{\alpha_2} u_2(x,t) - F_2(u_1(x,t), u_2(x,t), \ldots, u_q(x,t)), & \forall (x,t) \in \Omega, \\ \qquad\vdots & \\ \tau_q \dfrac{\partial^2 u_q(x,t)}{\partial t^2} + \gamma_q \dfrac{\partial u_q(x,t)}{\partial t} = d_q \Delta^{\alpha_q} u_q(x,t) - F_q(u_1(x,t), u_2(x,t), \ldots, u_q(x,t)), & \forall (x,t) \in \Omega. \end{cases} \tag{1.8}$$

It is important to point out that (1.7) generalizes various mathematical models used in the physical sciences. The following are some of examples where (1.7) finds interesting applications.

Example 1.3. Suppose that $q = 1$ in model (1.7), and let $u = u_1$. In this case, the system consists of a unique hyperbolic partial differential equation with fractional diffusion. If $F_1(u(x, t)) = \sin(u(x, t))$ then the resulting model is a fractional form of the damped sine-Gordon equation, which is a well-known system of mathematical physics. Other forms of F_1 readily lead to models like the (linear or nonlinear) Klein–Gordon equation and the double sine-Gordon equation, which are systems appearing in relativistic quantum mechanics. □

Example 1.4. The system (1.7) is a hyperbolic and fractional generalization of the ratio-dependent predator-prey system with Michaelis–Menten-type functional response [59]. In that model, $q = 2$, and $u_1(x,t)$ and $u_2(x,t)$ represent the prey and the predator densities, respectively, as functions of the spatial variable x and the time t. In the classical parabolic setting, $\alpha_1 = \alpha_2 = 2$, $\gamma_1 = \gamma_2 = 1$ and

$$F_1(u_1,u_2) = ru_1\left(1-\frac{u_1}{K}\right) - \frac{\alpha u_1 u_2}{u_2+\alpha\eta u_1}, \qquad \forall u_1,u_2 \in \mathbb{R}^+, \tag{1.9}$$

$$F_2(u_1,u_2) = \frac{\gamma\alpha u_1 u_2}{u_2+\alpha\eta u_1} - \mu u_2, \qquad \forall u_1,u_2 \in \mathbb{R}^+. \tag{1.10}$$

Here, r represents the maximal growth rate of the prey, γ denotes the conversion efficiency, μ is the predator death rate, K is the carrying capacity, α represents the capture rate and η is the handling time. After a scaling analysis, the system (1.7) with reactions (1.9)-(1.10) may be expressed in dimensionless form using the functions

$$F_1(u_1,u_2) = Ru_1\left(1-\frac{u_1}{S}\right) - \frac{Su_1u_2}{u_2+Su_1}, \qquad \forall u_1,u_2 \in \mathbb{R}^+, \tag{1.11}$$

$$F_2(u_1,u_2) = \frac{Su_1u_2}{u_2+Su_1} - Qu_2, \qquad \forall u_1,u_2 \in \mathbb{R}^+, \tag{1.12}$$

where the constants R, S and Q are positive. $\square$

Example 1.5. The model (1.7) is also a hyperbolic and fractional generalization of some systems which describe the interaction between some activator and inhibitor substances in chemistry [1]. Such is the case when $q = 2$ and

$$F_1(u_1,u_2) = u_1 - au_2 + bu_1u_2 - u_1^3, \tag{1.13}$$

$$F_2(u_1,u_2) = u_1 - cu_2. \tag{1.14}$$

In that context, u_1 and u_2 represent the amounts of the activator and inhibitor substances, respectively. All the parameters are positive, and the differentiation orders are equal to 2 in the original parabolic models. $\square$

Like these examples, there are various other models in the physical sciences which are generalized by (1.7) in various ways and, for that reason, its numerical investigation is an important topic of research. For simplification purposes and for the remainder of this thesis, convey that $\tau_j = 1$, $\gamma = \gamma_j \in \mathbb{R}^+ \cup \{0\}$, $d = d_j \in \mathbb{R}^+$ and $\alpha = \alpha_j \in (1,2]$, for each $j \in \mathcal{I}_q$. It is worth pointing out that our analysis would be similar for the general scenario, but we have chosen these simplifications for the sake of easiness. Moreover, we will observe the vector notation

$$\frac{\partial^2 u(x,t)}{\partial t^2} = \left(\frac{\partial^2 u_1(x,t)}{\partial t^2}, \frac{\partial^2 u_2(x,t)}{\partial t^2}, \ldots, \frac{\partial^2 u_q(x,t)}{\partial t^2}\right), \qquad \forall (x,t) \in \Omega. \tag{1.15}$$

$$\frac{\partial u(x,t)}{\partial t} = \left(\frac{\partial u_1(x,t)}{\partial t}, \frac{\partial u_2(x,t)}{\partial t}, \ldots, \frac{\partial u_q(x,t)}{\partial t}\right), \qquad \forall (x,t) \in \Omega. \tag{1.16}$$

$$\Delta^\alpha u(x,t) = \left(\Delta^\alpha u_1(x,t), \Delta^\alpha u_2(x,t), \ldots, \Delta^\alpha u_q(x,t)\right), \qquad \forall (x,t) \in \Omega, \tag{1.17}$$

$$F(x,t) = \left(F_1(u(x,t)), F_2(u(x,t)), \ldots, F_q(x,t)\right), \qquad \forall (x,t) \in \Omega. \tag{1.18}$$

Also, let $\phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_q(x))$ and $\psi(x) = (\psi_1(x), \psi_2(x), \ldots, \psi_q(x))$, for each $x \in B$. With this notation and the simplifications proposed above, the mathematical model (1.7) can be expressed alternatively in vector form as

$$\begin{aligned} &\frac{\partial^2 u(x,t)}{\partial t^2} + \gamma\frac{\partial u(x,t)}{\partial t} = d\Delta^\alpha u(x,t) - F(u(x,t)), \qquad \forall (x,t) \in \Omega, \\ &\text{such that } \begin{cases} u(x,0) = \phi(x), & \forall x \in B, \\ \dfrac{\partial u(x,0)}{\partial t} = \psi(x), & \forall x \in B, \\ u(x,t) = 0, & \forall (x,t) \in \partial B \times [0,T]. \end{cases} \end{aligned} \tag{1.19}$$

Finally, we will observe the following assumption for the remainder of this thesis:

(H) There exists a smooth function $G = (G_1, \ldots, G_q) : \mathbb{R}^q \to \mathbb{R}^q$ with $G_1, \ldots, G_q : \mathbb{R}^q \to \mathbb{R}$, such that

$$F_j(u) = \frac{\partial G_j(u)}{\partial u_j}, \qquad \forall j \in \mathcal{I}_q,\ \forall u \in \mathbb{R}^q. \tag{1.20}$$

As we will note in Section 2.3, the models described in Examples 1.3–1.5 satisfy this hypothesis. Moreover, it is also important to mention that a system (1.19) which satisfies this assumption need not be Hamiltonian. This fact implies that the type of systems satisfying **(H)** is substantially large. As expected, the finite-difference scheme proposed in the following section will be also general enough to consider not necessarily Hamiltonian models, like those nonlinear systems in which complex patterns appear.

1.2 Numerical model

The present section will be devoted to describe the numerical method to approximate the solutions of (1.19). Agree that h_i and τ are positive step-sizes, for each $i \in \mathcal{I}_p$. Let $N = T/\tau$ and $M_i = (b_i - a_i)/h_i$ be natural numbers, for each $i \in \mathcal{I}_p$. Consider uniform partitions of $[a_i, b_i]$ and $[0, T]$ given by

$$x_{i,k_i} = a_i + k_i h_i, \qquad \forall i \in \mathcal{I}_p,\ \forall k_i \in \overline{\mathcal{I}}_{M_i}, \tag{1.21}$$

$$t_n = n\tau, \qquad \forall n \in \overline{\mathcal{I}}_N, \tag{1.22}$$

$$t_{n+\frac{1}{2}} = t_n + \tau/2, \qquad \forall n \in \overline{\mathcal{I}}_{N-1}. \tag{1.23}$$

We introduce the discrete sets

$$\mathcal{K} = \prod_{i=1}^{p} \mathcal{I}_{M_i - 1}, \qquad \overline{\mathcal{K}} = \prod_{i=1}^{p} \overline{\mathcal{I}}_{M_i}, \tag{1.24}$$

and define $x_k = (x_{1,k_1}, \ldots, x_{p,k_p})$ for each multi-index $k = (k_1, \ldots, k_p) \in \overline{\mathcal{K}}$. Let $\partial\mathcal{K}$ represent the boundary of the mesh $\overline{\mathcal{K}}$, that is, let $\partial\mathcal{K}$ be the set of multi-indexes $k \in \overline{\mathcal{K}}$ such that $x_k \in \partial B$. In this manuscript, we will convey that

$$h = (h_1, \ldots, h_p), \qquad h_* = h_1 \cdots h_p, \tag{1.25}$$

and $\mathcal{R}_h$ will represent the spatial mesh $\{x_k : k \in \overline{\mathcal{K}}\}$. Moreover, $\mathcal{V}_h$ will denote the vector space of all real functions defined on $\mathcal{R}_h$, which vanish at those points on the boundary of B. For simplification purposes, we will agree that $w_k = w(x_k)$, for each $w \in \mathcal{V}_h$ and $k \in \overline{\mathcal{K}}$.

For each $j \in \mathcal{I}_q$ and $(k, n) \in \overline{\mathcal{K}} \times \overline{\mathcal{I}}_N$, let $u_{j,k}^n = u_j(x_k, t_n)$, and let $\mathbf{u}_{j,k}^n$ represent an approximation to the exact value of $u_{j,k}^n$. If $\mathbf{w} \in \mathcal{V}_h$ then we will use the notation $\mathbf{w} = (\mathbf{w}_k)_{k \in \overline{\mathcal{K}}}$. As a consequence, note that $\mathbf{u}_j^n \in \mathcal{V}_h$ can be represented as $(\mathbf{u}_{j,k}^n)_{k \in \overline{\mathcal{K}}}$, for each $j \in \mathcal{I}_q$. Moreover, we will use the convention $\mathbf{u}_j = (\mathbf{u}_j^n)_{n \in \overline{\mathcal{I}}_N}$ for each $j \in \mathcal{I}_q$, and we will set $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_q)$. Similarly, notice that $u_j^n \in \mathcal{V}_h$ can be represented as $u_j^n = (u_{j,k}^n)_{k \in \overline{\mathcal{K}}}$ for each $j \in \mathcal{I}_q$. Also, we let $u_j = (u_j^n)_{n \in \overline{\mathcal{I}}_N}$ for each $j \in \mathcal{I}_q$, and $u = (u_1, u_2, \ldots, u_q)$.

Definition 1.6. Define the *inner product* $\langle \cdot, \cdot \rangle : \mathcal{V}_h \times \mathcal{V}_h \to \mathbb{R}$ and the *norm* $\| \cdot \|_1 : \mathcal{V}_h \to \mathbb{R}$ by

$$\langle \mathbf{u}, \mathbf{v} \rangle = h_* \sum_{k \in \overline{\mathcal{K}}} \mathbf{u}_k \mathbf{v}_k, \quad \|\mathbf{u}\|_1 = h_* \sum_{k \in \overline{\mathcal{K}}} |\mathbf{u}_k|, \qquad \forall \mathbf{u}, \mathbf{v} \in \mathcal{V}_h. \tag{1.26}$$

The Euclidean norm induced by $\langle \cdot, \cdot \rangle$ will be denoted by $\| \cdot \|_2$. Meanwhile, $\| \cdot \|_\infty$ will be the usual infinity norm in $\mathcal{V}_h$.

Definition 1.7. For each sequence $(\mathbf{v}^n)_{n=0}^N \subseteq \mathcal{V}_h$ and $(k, n) \in \overline{\mathcal{K}} \times \mathcal{I}_{N-1}$, we introduce the discrete linear operators:

$$\begin{cases} \delta_t \mathbf{v}_k^n = \dfrac{\mathbf{v}_k^{n+1} - \mathbf{v}_k^n}{\tau}, & \delta_t^{(1)} \mathbf{v}_k^n = \dfrac{\mathbf{v}_k^{n+1} - \mathbf{v}_k^{n-1}}{2\tau}, \\ \delta_t^{(2)} \mathbf{v}_k^n = \dfrac{\mathbf{v}_k^{n+1} - 2\mathbf{v}_k^n + \mathbf{v}_k^{n-1}}{\tau^2}, & \mu_t \mathbf{v}_k^n = \dfrac{\mathbf{v}_k^{n+1} + \mathbf{v}_k^n}{2}. \end{cases} \tag{1.27}$$

It is well known that these difference operators provide consistent approximations to some differential operators under suitable analytical conditions. Moreover, we will set $\hat{\mathbf{v}}_k^n = \mu_t \mathbf{v}_k^n$ for briefness.

Definition 1.8 (Ortigueira [60]). For any function $f : \mathbb{R} \to \mathbb{R}$, any $h > 0$ and $\alpha > -1$, the *fractional centered difference* of order α of f at the point x is defined as

$$\Delta_h^{(\alpha)} f(x) = \sum_{k=-\infty}^{\infty} g_k^{(\alpha)} f(x - kh), \qquad \forall x \in \mathbb{R}, \tag{1.28}$$

whenever the double series at the right-hand side exists. Here,

$$g_k^{(\alpha)} = \frac{(-1)^k \Gamma(\alpha + 1)}{\Gamma(\frac{\alpha}{2} - k + 1)\Gamma(\frac{\alpha}{2} + k + 1)}, \qquad \forall k \in \mathbb{Z}. \tag{1.29}$$

Lemma 1.9 (Çelik and Duman [61]). *If $1 < \alpha \leq 2$ then the coefficients $(g_k^{(\alpha)})_{k=-\infty}^{\infty}$ satisfy:*

(a) $g_0^{(\alpha)} > 0$,

(b) $g_k^{(\alpha)} = g_{-k}^{(\alpha)} \leq 0$ *for all* $k \neq 0$, *and*

(c) $\displaystyle\sum_{k=-\infty}^{\infty} g_k^{(\alpha)} = 0$. *As a consequence, it follows that* $g_0^{(\alpha)} = -\displaystyle\sum_{\substack{k=-\infty \\ k\neq 0}}^{\infty} g_k^{(\alpha)}$.

Lemma 1.10 (Çelik and Duman [61]). *Let $f \in \mathcal{C}^5(\mathbb{R})$ and assume that all its derivatives up to order five are integrable. If $1 < \alpha \leq 2$ then, for almost all x,*

$$-\frac{\Delta_h^\alpha f(x)}{h^\alpha} = \frac{d^\alpha f(x)}{d|x|^\alpha} + \mathcal{O}(h^2). \tag{1.30}$$

Definition 1.11. For each sequence $(\mathbf{v}^n)_{n=0}^N \subseteq \mathcal{V}_h$, $(k, n) \in \overline{\mathcal{K}} \times \mathcal{I}_{N-1}$ and $i \in \mathcal{I}_p$, define the linear operator

$$\delta_{x_i}^{(\alpha)} \mathbf{v}_k^n = -\frac{1}{h_i^\alpha} \sum_{l=0}^{M_i} g_{k_i - l}^{(\alpha)} \mathbf{v}_{k_1,\ldots,k_{i-1},l,k_{i+1},\ldots,k_p}^n. \tag{1.31}$$

In light of Lemma 1.10, the operators (1.31) provide quadratically consistent approximations to the Riesz partial derivative of u with respect to x_i of order α at $(x_{1,k_1}, \ldots, x_{i-1,k_{i-1}}, x_{i,k_i}, x_{i+1,k_{i+1}}, \ldots, x_{p,k_p})$ and time t_n. Meanwhile, the fractional Laplacian will be estimated with a quadratic order of consistency, using the expression

$$\delta_x^{(\alpha)} \mathbf{v}_k^n = \sum_{i=1}^p \delta_{x_i}^{(\alpha)} \mathbf{v}_k^n. \tag{1.32}$$

The following results will be needed in our study.

Lemma 1.12 (Macías-Díaz [40]). *For each $i \in \mathcal{I}_p$ and $\alpha \in (1, 2]$, there exists a unique positive self-adjoint (square-root) linear operator $\Lambda_{x_i}^{(\alpha)} : \mathcal{V}_h \to \mathcal{V}_h$, such that $\langle -\delta_{x_i}^{(\alpha)} \mathbf{u}, \mathbf{v}\rangle = \langle \Lambda_{x_i}^{(\alpha)} \mathbf{u}, \Lambda_{x_i}^{(\alpha)} \mathbf{v}\rangle$, for each $\mathbf{u}, \mathbf{v} \in \mathcal{V}_h$.*

Lemma 1.13 (Macías-Díaz [62]). *Let $\mathbf{v} \in \mathcal{V}_h$ and $i \in \mathcal{I}_p$. Let $\alpha \in (1, 2]$ and define the constants*

$$g_h^{(\alpha)} = 2 g_0^{(\alpha)} h_* \sqrt{\sum_{i=1}^p h_i^{-2\alpha}}, \qquad \mathbf{g}_h^{(\alpha)} = 2 g_0^{(\alpha)} h_* \sum_{i=1}^p h_i^{-\alpha}. \tag{1.33}$$

Then

(a) $\|\Lambda_{x_i}^{(\alpha)}\mathbf{v}\|_2^2 \leq 2g_0^{(\alpha)}h_* h_i^{-\alpha}\|\mathbf{v}\|_2^2$.

(b) $\|\delta_{x_i}^{(\alpha)}\mathbf{v}\|_2^2 = \|\Lambda_{x_i}^{(\alpha)}\Lambda_{x_i}^{(\alpha)}\mathbf{v}\|_2^2$.

(c) $\|\delta_{x_i}^{(\alpha)}\mathbf{v}\|_2^2 \leq 4\left(g_0^{(\alpha)}h_* h_i^{-\alpha}\right)^2\|\mathbf{v}\|_2^2$.

(d) $\displaystyle\sum_{i\in\mathcal{I}_p}\|\delta_{x_i}^{(\alpha)}\mathbf{v}\|_2^2 \leq \left(g_h^{(\alpha)}\|\mathbf{v}\|_2\right)^2$ *and* $\displaystyle\sum_{i\in\mathcal{I}_p}\|\Lambda_{x_i}^{(\alpha)}\mathbf{v}\|_2^2 \leq \mathbf{g}_h^{(\alpha)}\|\mathbf{v}\|_2^2$.

Definition 1.14. Suppose that $G:\mathbb{R}^q\to\mathbb{R}$ is a differentiable function, and let $\mathbf{u}_j = (\mathbf{u}_j^n)_{n\in\overline{\mathcal{I}}_N}\subseteq\mathcal{V}_h$, for each $j\in\mathcal{I}_q$. For each $j\in\mathcal{I}_q$ and $(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_{N-1}$, we define the nonlinear operator

$$\delta_{t,u_j}G_{j,k}^n(\mathbf{u}) = \frac{G_j\left(\hat{\mathbf{u}}_{1,k}^n,\ldots,\hat{\mathbf{u}}_{j-1,k}^n,\mathbf{u}_{j,k}^{n+1},\hat{\mathbf{u}}_{j+1,k}^n,\ldots,\hat{\mathbf{u}}_{q,k}^n\right) - G_j\left(\hat{\mathbf{u}}_{1,k}^n,\ldots,\hat{\mathbf{u}}_{j-1,k}^n,\mathbf{u}_{j,k}^n,\hat{\mathbf{u}}_{j+1,k}^n,\ldots,\hat{\mathbf{u}}_{q,k}^n\right)}{\mathbf{u}_{j,k}^{n+1}-\mathbf{u}_{j,k}^n}, \tag{1.34}$$

if $\mathbf{u}_{j,k}^{n+1}\neq\mathbf{u}_{j,k}^n$. Otherwise,

$$\delta_{t,u_j}G_{j,k}^n(\mathbf{u}) = \frac{\partial G(u(x_k,t_{n+\frac{1}{2}}))}{\partial u_j}. \tag{1.35}$$

Next, we extend dimensionally the operators of Definitions 1.7, 1.11 and 1.14

Definition 1.15. For each $(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_N$, we let $\overline{\mathbf{u}}_k^n = (\mathbf{u}_{1,k}^n,\mathbf{u}_{2,k}^n,\ldots,\mathbf{u}_{q,k}^n)$. We define

$$\delta_t\overline{\mathbf{u}}_k^n = \left(\delta_t\mathbf{u}_{1,k}^n,\delta_t\mathbf{u}_{2,k}^n\ldots,\delta_t\mathbf{u}_{q,k}^n\right),\qquad\forall(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_{N-1}, \tag{1.36}$$

$$\delta_t^{(1)}\overline{\mathbf{u}}_k^n = \left(\delta_t^{(1)}\mathbf{u}_{1,k}^n,\delta_t^{(1)}\mathbf{u}_{2,k}^n\ldots,\delta_t^{(1)}\mathbf{u}_{q,k}^n\right),\qquad\forall(k,n)\in\overline{\mathcal{K}}\times\mathcal{I}_{N-1}, \tag{1.37}$$

$$\delta_t^{(2)}\overline{\mathbf{u}}_k^n = \left(\delta_t^{(2)}\mathbf{u}_{1,k}^n,\delta_t^{(2)}\mathbf{u}_{2,k}^n\ldots,\delta_t^{(2)}\mathbf{u}_{q,k}^n\right),\qquad\forall(k,n)\in\overline{\mathcal{K}}\times\mathcal{I}_{N-1}, \tag{1.38}$$

$$\mu_t\overline{\mathbf{u}}_k^n = \left(\mu_t\mathbf{u}_{1,k}^n,\mu_t\mathbf{u}_{2,k}^n,\ldots,\mu_t\mathbf{u}_{q,k}^n\right),\qquad\forall(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_{N-1}, \tag{1.39}$$

$$\delta_{x_i}^{(\alpha)}\overline{\mathbf{u}}_k^n = \left(\delta_{x_i}^{(\alpha)}\mathbf{u}_{1,k}^n,\delta_{x_i}^{(\alpha)}\mathbf{u}_{2,k}^n,\ldots,\delta_{x_i}^{(\alpha)}\mathbf{u}_{q,k}^n\right),\qquad\forall(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_N,\ \forall i\in\mathcal{I}_p, \tag{1.40}$$

$$\delta_{t,u}G_k^n(\mathbf{u}) = \left(\delta_{t,u_1}G_{1,k}^n(\mathbf{u}),\delta_{t,u_2}G_{2,k}^n(\mathbf{u}),\ldots,\delta_{t,u_q}G_{q,k}^n(\mathbf{u})\right),\qquad\forall(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_{N-1}. \tag{1.41}$$

As expected, we will set

$$\delta_x^{(\alpha)}\overline{\mathbf{u}}_k^n = \sum_{i=1}^p\delta_{x_i}^{(\alpha)}\overline{\mathbf{u}}_k^n,\qquad\forall(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_N. \tag{1.42}$$

In an analogous fashion, we let $\overline{u}_k^n = (u_{1,k}^n,u_{2,k}^n,\ldots,u_{q,k}^n)$ for each $(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_N$, and we can readily define the operators (1.36)–(1.42) for $\overline{u}$ as above.

Using the hypothesis **(H)** and the notation introduced in the present section, the finite-difference scheme used in this thesis to solve (1.19) is the set of difference equations

$$\begin{gathered}\mu_t\delta_t^{(2)}\overline{\mathbf{u}}_k^n+\gamma\delta_t\overline{\mathbf{u}}_k^n = d\mu_t\delta_x^{(\alpha)}\overline{\mathbf{u}}_k^n-\delta_{t,u}G_k^n(\mathbf{u}),\qquad\forall(k,n)\in\mathcal{K}\times\mathcal{I}_{N-2},\\ \text{subject to}\begin{cases}\overline{\mathbf{u}}_k^0 = \phi(x_k), & \forall k\in\mathcal{K},\\ \overline{\mathbf{u}}_k^1 = \psi(x_k), & \forall k\in\mathcal{K},\\ \overline{\mathbf{u}}_k^2 = \chi(x_k), & \forall k\in\mathcal{K},\\ \overline{\mathbf{u}}_k^n = 0, & \forall(k,n)\in\partial\mathcal{K}\times\overline{\mathcal{I}}_N.\end{cases}\end{gathered} \tag{1.43}$$

Note that we chose to define the initial data exactly by means of the functions $\phi,\psi,\chi:\Omega\to\mathbb{R}^q$. Also, it is clear that this system is a fully explicit model, whence the existence and uniqueness of solutions is an immediate consequence. Indeed, for each $(k,n)\in\mathcal{K}\times\mathcal{I}_{N-1}$, the difference equation of (1.43) can be rewritten as

$$\overline{\mathbf{u}}_k^{n+2} = \overline{\mathbf{u}}_k^{n+1}+\overline{\mathbf{u}}_k^n-\overline{\mathbf{u}}_k^{n-1}+2\tau^2\left[d\mu_t\delta_x^{(\alpha)}\overline{\mathbf{u}}_k^n-\delta_{t,u}G_k^n(\mathbf{u})-\gamma\delta_t\overline{\mathbf{u}}_k^n\right] \tag{1.44}$$

The discrete model (1.43) is, thus, a four-step numerical scheme. For the sake of convenience, Figure 1.1 provides the forward-difference stencil of the discrete model in the case when $p = q = 1$.

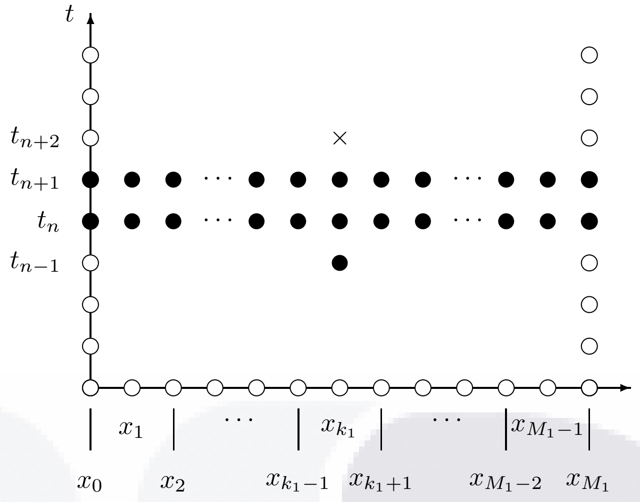


Figure 1.1: Forward-difference stencil for the approximation to the exact solution of the one dimensional form of (1.7) at the time t_n, using the finite-difference scheme (1.43) when $p = q = 1$. The black circles represent the known approximations at the times t_{n-1}, t_n and t_{n+1}, while the cross denotes the unknown approximation at the time t_{n+2}.

1.3 Numerical properties

The properties of consistency, stability, boundedness and convergence of this scheme will be mathematically established in this section. In a first stage, we wish to prove the quadratic consistency of the scheme. To that end, we will consider the continuous operator $\mathcal{L}$ and the discrete operator L, which are respectively defined as

$$\mathcal{L}_u(u(x,t)) = \frac{\partial^2 u(x,t)}{\partial t^2} + \gamma\frac{\partial u(x,t)}{\partial t} - d\Delta^\alpha u(x,t) + F(u(x,t)), \qquad \forall (x,t) \in \Omega, \tag{1.45}$$

$$L_u(\overline{u}_k^n) = \mu_t\delta_t^{(2)}\overline{u}_k^n + \gamma\delta_t\overline{u}_k^n - d\mu_t\delta_x^{(\alpha)}\overline{u}_k^n + \delta_{t,u}G_k^n(u), \qquad \forall (k,n) \in \mathcal{K}\times\mathcal{I}_{N-2}. \tag{1.46}$$

Throughout, we will suppose that **(H)** is satisfied.

Theorem 1.16 (Consistency). *If $u \in \mathcal{C}_{x,t}^{5,4}(\overline{\Omega})$ and $F \in L^\infty(\mathbb{R}^q)$ then there exists a constant C which is independent of h and τ, such that $\|\mathcal{L}_u(u(x_k,t_{n+\frac{1}{2}})) - L_u(\overline{u}_k^n)\|_\infty \leq C(\tau^2 + \|h\|_2^2)$, for each $(k,n) \in \overline{\mathcal{K}}\times\mathcal{I}_{N-2}$.*

Proof. Note that the smoothness of u and the essential boundedness of F imply that $u_j \in \mathcal{C}_{x,t}^{5,4}(\overline{\Omega})$ and $F_j \in L^\infty(\mathbb{R}^q)$, for each $j \in \mathcal{I}_q$. The consistency of the discrete operators in Definitions 1.7, 1.11 and 1.14 along with Taylor's theorem guarantee now that there exist constants C_1^j, C_2^j, $C_{3,i}^j$ and C_4^j which are independent of h and τ, such that

$$\left|\frac{\partial^2 u_j(x_k,t_{n+\frac{1}{2}})}{\partial t^2} - \mu_t\delta_t^{(2)}u_j(x_k,t_n)\right| \leq C_1^j\tau^2, \qquad \forall (k,n)\in\mathcal{K}\times\mathcal{I}_{N-2},\ \forall j\in\mathcal{I}_q, \tag{1.47}$$

$$\left|\frac{\partial u_j(x_k,t_{n+\frac{1}{2}})}{\partial t} - \delta_t u_j(x_k,t_n)\right| \leq C_2^j\tau^2, \qquad \forall (k,n)\in\mathcal{K}\times\mathcal{I}_{N-1},\ \forall j\in\mathcal{I}_q, \tag{1.48}$$

$$\left|\frac{\partial^\alpha u_j(x_k,t_{n+\frac{1}{2}})}{\partial|x_i|^\alpha} - \mu_t\delta_{x_i}^{(\alpha)}u_j(x_k,t_n)\right| \leq C_{3,i}^j(h_i^2+\tau^2), \qquad \forall (k,n)\in\mathcal{K}\times\mathcal{I}_{N-1},\ \forall (i,j)\in\mathcal{I}_p\times\mathcal{I}_q, \tag{1.49}$$

$$\left|F_j(x_k,t_{n+\frac{1}{2}}) - \delta_{t,u_j}G_k^n(u)\right| \leq C_4^j\tau^2, \qquad \forall (k,n)\in\mathcal{K}\times\mathcal{I}_{N-1},\ \forall j\in\mathcal{I}_q. \tag{1.50}$$

Let $C_l = \max\{C_l^j : j \in \mathcal{I}_q\}$ for each $l = 1, 2, 4$, and define $C_3 = \max\{C_{3,i}^j : (i,j) \in \mathcal{I}_q \times \mathcal{I}_q\}$. Clearly, these constants are independent of h and τ. Moreover, using the triangle inequality, it follows that

$$\begin{aligned}\left\|\mathcal{L}_u(u(x_k,t_{n+\frac{1}{2}})) - L_u(\overline{u}_k^n)\right\|_\infty &\leq \left\|\frac{\partial^2 u(x_k,t_{n+\frac{1}{2}})}{\partial t^2} - \mu_t\delta_t^{(2)}\overline{u}_k^n\right\|_\infty + \gamma\left\|\frac{\partial u(x_k,t_{n+\frac{1}{2}})}{\partial t} - \delta_t\overline{u}_k^n\right\|_\infty \\ &\quad + d\left\|\Delta^\alpha u(x_k,t_{n+\frac{1}{2}}) - \mu_t\delta_x^{(\alpha)}\overline{u}_k^n\right\|_\infty + \left\|F(u(x_k,t_{n+\frac{1}{2}})) - \delta_{t,u}G_k^n(u)\right\|_\infty \\ &\leq C_1\tau^2 + \gamma C_2\tau^2 + dC_3(\|h\|_2^2 + \tau^2) + C_4\tau^2,\end{aligned} \tag{1.51}$$

for each $(k,n) \in \mathcal{K} \times \mathcal{I}_{N-2}$. The conclusion readily follows with $C = \max\{C_1, \gamma C_2, dC_3, C_4\}$. □

Lemma 1.17 (Macías-Díaz [40]). *Let $G : \mathbb{R}^q \to \mathbb{R}^q$ be such that $G \in \mathcal{C}^2(\mathbb{R}^q)$ and $D^2G \in L^\infty(\mathbb{R})$, and let $(\mathbf{u}_j^n)_{n=0}^N$, $(\mathbf{v}_j^n)_{n=0}^N$ and $(\mathbf{R}_j^n)_{n=0}^N$ be sequences in $\mathcal{V}_h$, for each $j \in \mathcal{I}_q$. Let $\mathbf{e}_j^n = \mathbf{v}_j^n - \mathbf{u}_j^n$ and $\widetilde{G}_j^n = \delta_{v_j,t}G^n(\mathbf{v}) - \delta_{u_j,t}G^n(\mathbf{u})$, for each $n \in \overline{\mathcal{I}}_{N-1}$ and $j \in \mathcal{I}_q$. There exist constants $C_1, C_2, C_3 \in \mathbb{R}^+$ which depend only on G, such that, for each $j \in \mathcal{I}_q$*

$$\|\widetilde{G}_j^n\|_2^2 \leq C_1(\|\mathbf{e}_j^{n+1}\|_2^2 + \|\mathbf{e}_j^n\|_2^2), \qquad \forall n \in \overline{\mathcal{I}}_{N-1}, \tag{1.52}$$

$$2\tau\sum_{n=1}^m \left|\langle \mathbf{R}_j^n - \widetilde{G}_j^n, \delta_t\mathbf{e}_j^n\rangle\right| \leq 2\tau\sum_{n=0}^m \|\mathbf{R}_j^n\|_2^2 + C_2\|\mathbf{e}_j^0\|_2^2 + C_3\tau\sum_{n=0}^m \|\delta_t\mathbf{e}_j^n\|_2^2, \qquad \forall m \in \mathcal{I}_{N-1}, \tag{1.53}$$

$$m\tau^2\sum_{n=1}^m \|\widetilde{G}_j^n\|_2^2 \leq 4C_1T^2\|\mathbf{e}_j^0\|_2^2 + 4C_1T^3\tau\sum_{n=0}^m \|\delta_t\mathbf{e}_j^n\|_2^2, \qquad \forall m \in \mathcal{I}_{N-1}. \tag{1.54}$$

Lemma 1.18 (Macías-Díaz [40]). *Let G, $(\mathbf{u}_j^n)_{n=0}^N$, $(\mathbf{v}_j^n)_{n=0}^N$ and $(\mathbf{R}_j^n)_{n=0}^N$ be as in Lemma 1.17. Let $\mathbf{e}_j^n = \mathbf{v}_j^n - \mathbf{u}_j^n$ and $\widetilde{G}_j^n = \delta_{v_j,t}G^n(\mathbf{v}) - \delta_{u_j,t}G^n(\mathbf{u})$, for each $n \in \overline{\mathcal{I}}_{N-1}$ and $j \in \mathcal{I}_q$. If*

$$\mu_t\delta_t^{(2)}\mathbf{e}_j^n + \gamma\delta_t\mathbf{e}_j^n - d\mu_t\delta_x^{(\alpha)}\mathbf{e}_j^n + \widetilde{G}_j^n = \mathbf{R}_j^n, \qquad \forall n \in \mathcal{I}_{N-1},\ \forall j \in \mathcal{I}_q, \tag{1.55}$$

then for each $m \in \mathcal{I}_{N-1}$ and $j \in \mathcal{I}_q$,

$$\begin{aligned}\tau^2\|\delta_t^{(2)}\mathbf{e}_j^{m+1}\|_2^2 &\leq 160C_1T^2\|\mathbf{e}_j^0\|_2^2 + 20\mu_t\|\delta_t\mathbf{e}_j^0\|_2^2 + 5p\tau^2 g_h^{(\alpha)} d^2\sum_{i=1}^p \left[\|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^1\|_2^2 + \|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^{m+1}\|_2^2\right] \\ &\quad + 40T\tau\sum_{n=1}^m \|\mathbf{R}_j^n\|_2^2 + 20(8C_1T^2 + \gamma^2)T\tau\sum_{n=0}^m \|\delta_t\mathbf{e}_j^n\|_2^2.\end{aligned} \tag{1.56}$$

The following discrete version of Gronwall's inequality will be needed in the sequel.

Lemma 1.19 (Pen-Yu [63]). *Let $(\omega^n)_{n=0}^N$ and $(\rho^n)_{n=0}^N$ be finite sequences of nonnegative mesh functions, and suppose that there exists $C \geq 0$ such that*

$$\omega^k \leq \rho^k + C\tau\sum_{n=0}^{k-1}\omega^k, \quad \forall k \in \mathcal{I}_{N-1}. \tag{1.57}$$

Then $\omega^n \leq \rho^n e^{Cn\tau}$ for each $n \in \overline{I}_N$. □

Theorem 1.20 (Stability). *Let $G : \mathbb{R}^q \to \mathbb{R}^q$ be such that $G \in \mathcal{C}^2(\mathbb{R}^q)$ and $D^2G \in L^\infty(\mathbb{R})$, and suppose that $\mathbf{u}$ and $\mathbf{v}$ are solutions of (1.43) corresponding to the initial conditions (ϕ_u, ψ_u, χ_u) and (ϕ_v, ψ_v, χ_v), respectively. Let $\mathbf{e} = \mathbf{u} - \mathbf{v}$, and assume that $\frac{5}{2}p\tau^2 g_h^{(\alpha)} < 1$. Then there exists a constant $C_0 \in \mathbb{R}^+$ which is independent of τ, h, $\mathbf{u}$ and $\mathbf{v}$, and a constant $0 < \eta_0 < 1$ independent of $\mathbf{u}$ and $\mathbf{v}$, such that*

$$\frac{1}{2}\|\delta_t\mathbf{e}_j^n\|_2^2 + (1-\eta_0)d\sum_{i=1}^p \|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^n\|_2^2 \leq C_0\left(\|\mathbf{e}_j^0\|_2^2 + \mu_t\|\delta_t\mathbf{e}_j^0\|_2^2 + \sum_{i=1}^p \|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^1\|_2^2\right), \quad \forall n \in \mathcal{I}_{N-1},\ \forall j \in \mathcal{I}_q. \tag{1.58}$$

Proof. Let η_0 satisfy $\frac{5}{2}p\tau^2 g_h^{(\alpha)} < \eta_0 < 1$. Note that the following discrete problem is satisfied:

$$\mu_t\delta_t^{(2)}\overline{\mathbf{e}}_k^n + \gamma\delta_t\overline{\mathbf{e}}_k^n - d\mu_t\delta_x^{(\alpha)}\overline{\mathbf{e}}_k^n + \delta_{t,u}G_k^n(\mathbf{u}) - \delta_{t,v}G_k^n(\mathbf{v}) = 0, \qquad \forall(k,n)\in\mathcal{K}\times\mathcal{I}_{N-2},$$
$$\text{subject to}\begin{cases} \overline{\mathbf{e}}_k^0 = \phi_u(x_k) - \phi_v(x_k), & \forall k\in\mathcal{K}, \\ \overline{\mathbf{e}}_k^1 = \psi_u(x_k) - \psi_v(x_k), & \forall k\in\mathcal{K}, \\ \overline{\mathbf{e}}_k^2 = \chi_u(x_k) - \chi_v(x_k), & \forall k\in\mathcal{K}, \\ \overline{\mathbf{e}}_k^n = 0, & \forall(k,n)\in\partial\mathcal{K}\times\overline{\mathcal{I}}_N. \end{cases} \tag{1.59}$$

Following the nomenclature introduced at the beginning of this section, we identify the left-hand side of the difference equation of (1.59) as $L_u(\overline{\mathbf{u}}_k^n) - L_v(\overline{\mathbf{v}}_k^n)$. Moreover, for the sake of convenience, we let $\widetilde{G}_k^n = \delta_{t,u}G_k^n(\mathbf{u}) - \delta_{t,v}G_k^n(\mathbf{v})$, for each $(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_{N-1}$. For each $j\in\mathcal{I}_q$ and $n\in\mathcal{I}_{N-2}$, the following identities are satisfied:

$$\langle\mu_t\delta_t^{(2)}\mathbf{e}_j^n, \delta_t\mathbf{e}_j^n\rangle = \frac{1}{2}\delta_t\mu_t\|\delta_t\mathbf{e}_j^{n-1}\|_2^2 - \frac{\tau^2}{4}\delta_t\|\delta_t^{(2)}\mathbf{e}_j^n\|_2^2, \tag{1.60}$$
$$\langle-\mu_t\delta_{x_i}^{(\alpha)}\mathbf{e}_j^n, \delta_t\mathbf{e}_j^n\rangle = \frac{1}{2}\delta_t\|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^n\|_2^2, \qquad \forall i\in\mathcal{I}_p. \tag{1.61}$$

Next, we calculate the inner product of $\delta_t\mathbf{e}_j^n$ with both sides of the difference equation $L_u(\mathbf{u}_j^n) - L_v(\overline{\mathbf{v}}_k^n) = 0$, substitute then the identities above, and we obtain next the sum of the resulting identity for all $n\in\mathcal{I}_m$. Multiply then by 2τ on both sides, apply Lemma 1.17 with $\mathbf{R}^n = 0$ and simplify algebraically to obtain that, for each $m\in\mathcal{I}_{N-1}$,

$$\begin{aligned}
\frac{1}{2}\|\delta_t\mathbf{e}_j^{m+1}\|_2^2 + d\sum_{i=1}^p\|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^{m+1}\|_2^2 &\le \mu_t\|\delta_t\mathbf{e}_j^m\|_2^2 + d\sum_{i=1}^p\|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^{m+1}\|_2^2 \\
&\le \mu_t\|\delta_t\mathbf{e}_j^0\|_2^2 + d\sum_{i=1}^p\|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^1\|_2^2 + \frac{\tau^2}{2}\|\delta_t^{(2)}\mathbf{e}_j^{m+1}\|_2^2 + 2\tau\sum_{n=1}^m\left|\langle\widetilde{G}_j^n, \delta_t\mathbf{e}_j^n\rangle\right| \\
&\le \rho_j + \frac{5}{2}p\tau^2 g_h^{(\alpha)} d^2\sum_{i=1}^p\|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^{m+1}\|_2^2 + C_5\tau\sum_{n=0}^m\|\delta_t\mathbf{e}_j^n\|_2^2 \\
&\le \rho_j + d\eta_0\sum_{i=1}^p\|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^{m+1}\|_2^2 + C_5\tau\sum_{n=0}^m\omega_j^n.
\end{aligned} \tag{1.62}$$

Here, $C_4 = \max\{C_2 + 80C_1T^2, 11\}$, $C_5 = 2C_3 + 20(8C_1T^2 + \gamma^2)T$ and

$$\rho_j = C_4\left(\|\mathbf{e}_j^0\|_2^2 + \mu_t\|\delta_t\mathbf{e}_j^0\|_2^2 + d\sum_{i=1}^p\|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^1\|_2^2\right), \tag{1.63}$$
$$\omega_j^n = \frac{1}{2}\|\delta_t\mathbf{e}_j^n\|_2^2 + (1-\eta_0)d\sum_{i=1}^p\|\Lambda_{x_i}^{(\alpha)}\mathbf{e}_j^n\|_2^2, \quad \forall n\in\mathcal{I}_{N-1}. \tag{1.64}$$

Subtracting the second term on the right-hand side of (1.62), we note that the hypotheses of Lemma 1.19 are readily satisfied with $C = C_5$ and $\rho^k = \rho$ for each $k\in\mathcal{I}_{N-2}$. The conclusion of this theorem follows with $C_0 = C_4e^{C_5T}$. □

The following result is readily obtained using a proof similar to that of Theorem 1.20.

Corollary 1.21 (Boundedness). *Let $G:\mathbb{R}^q\to\mathbb{R}^q$ satisfy $G\in\mathcal{C}^2(\mathbb{R}^q)$ and $D^2G\in L^\infty(\mathbb{R})$. If* $\mathbf{u}$ *is the solution of* (1.43) *then there exists a constant $C\in\mathbb{R}^+$ such that $\|\overline{\mathbf{u}}_k^n\|_\infty \le C$, for each $(k,n)\in\overline{\mathcal{K}}\times\overline{\mathcal{I}}_N$.* □

Theorem 1.22 (Convergence). *Let $u\in\mathcal{C}_{x,t}^{5,4}(\overline{\Omega})$ be a solution of* (1.19), *and suppose that $G\in\mathcal{C}^2(\mathbb{R})$ and $D^2G\in L^\infty(\mathbb{R})$. If $\frac{5}{2}p\tau^2 g_h^{(\alpha)} d < 1$ and* (1.43) *has exact initial data then the numerical solution converges to that of the continuous problem with order $\mathcal{O}(\tau^2 + \|h\|_2^2)$ in the Euclidean norm.*

Proof. The proof is similar to that of Theorem 1.20. Fix η_0 as in that theorem, let $\mathbf{R}_k^n$ be the local truncation error at (x_k, t_n), for each $(k,n) \in \overline{\mathcal{K}} \times \overline{\mathcal{I}}_N$, and define $\epsilon_j^n = \mathbf{u}_j^n - u_j^n$. Then the following system is satisfied:

$$\begin{aligned} \mu_t \delta_t^{(2)} \overline{\epsilon}_k^n + \gamma \delta_t \overline{\epsilon}_k^n - d\mu_t \delta_x^{(\alpha)} \overline{\epsilon}_k^n + \delta_{t,u} G_k^n(\mathbf{u}) - \delta_{t,u} G_k^n(u) = \overline{\mathbf{R}}_k^n, \qquad \forall (k,n) \in \mathcal{K} \times \mathcal{I}_{N-2}, \\ \text{subject to} \begin{cases} \overline{\epsilon}_k^0 = \overline{\epsilon}_k^1 = \overline{\epsilon}_k^2 = 0, & \forall k \in \mathcal{K}, \\ \overline{\epsilon}_k^n = 0, & \forall (k,n) \in \partial\mathcal{K} \times \overline{\mathcal{I}}_N. \end{cases} \end{aligned} \tag{1.65}$$

Mimicking the poof of Theorem 1.20, we let $\widetilde{G}_k^n = \delta_{t,u} G_k^n(\mathbf{u}) - \delta_{t,u} G_k^n(u)$, for each $(k,n) \in \overline{\mathcal{K}} \times \overline{\mathcal{I}}_{N-2}$. Proceeding as in that proof of the previous result, we reach the inequality

$$\frac{1}{2}\|\delta_t \epsilon_j^{m+1}\|_2^2 + d\sum_{i=1}^p \|\Lambda_{x_i}^{(\alpha)} \epsilon_j^{m+1}\|_2^2 \leq \rho^{m+1} + \frac{5}{2} p\tau^2 g_h^{(\alpha)} d \sum_{i=1}^p \|\Lambda_{x_i}^{(\alpha)} \epsilon_j^{m+1}\|_2^2 + C_5 \tau \sum_{n=0}^m \omega^n, \quad \forall k \in \mathcal{I}_{N-1}, \tag{1.66}$$

where $C_4 = \max\{C_2 + 80C_0T^2, 11, 20T + 2\}$, the constant C_5 is as in the proof of Theorem 1.20 and

$$\rho_j^m = C_4 \left(\|\epsilon_j^0\|_2^2 + \mu_t \|\delta_t \epsilon_j^0\|_2^2 + d\sum_{i=1}^p \|\Lambda_{x_i}^{(\alpha)} \epsilon_j^1\|_2^2 + \tau \sum_{n=0}^{m-1} \|\mathbf{R}_j^n\|_2^2 \right), \qquad \forall m \in \mathcal{I}_{N-1}, \tag{1.67}$$

$$\omega_j^m = \frac{1}{2}\|\delta_t \epsilon_j^m\|_2^2 + (1-\eta_0) d \sum_{i=1}^p \|\Lambda_{x_i}^{(\alpha)} \epsilon_j^m\|_2^2, \qquad \forall m \in \mathcal{I}_{N-1}. \tag{1.68}$$

Move the second term on the right-hand side of (1.66) to the left-hand side and apply Lemma 1.19. Use then the initial data of (1.65), and use Theorem 1.16 to see that there exists a constant A_1 independent of τ and h, such that

$$\frac{1}{2}\|\delta \epsilon_j^n\|_2^2 \leq \frac{1}{2}\|\delta_t \epsilon_j^n\|_2^2 + (1-\eta_0) d \sum_{i=1}^p \|\Lambda_{x_i}^{(\alpha)} \epsilon_j^n\|_2^2 \leq A_1 \tau \sum_{n=0}^{m-1} \|\mathbf{R}_j^n\|_2^2 \leq A_2(\tau^2 + \|h\|_2^2)^2, \tag{1.69}$$

for each $n \in \mathcal{I}_{N-1}$ and $j \in \mathcal{I}_q$. Here, $A_2 = A_1 CT$, and C is the constant provided by Theorem 1.16. Multiply both ends of (1.69) by 2 and use the reversed triangle inequality to obtain that $\|\epsilon_j^{n+1}\|_2 - \|\epsilon_j^n\|_2 \leq \sqrt{2A_2}\tau(\tau^2 + \|h\|_2^2)$. Next, take the sum on both sides of this inequality for n between 0 and $m-1$, the formula for telescoping sums and the data at $n = 0$ to obtain

$$\|\epsilon_j^m\|_2 \leq \sqrt{2A_2}\tau \sum_{n=0}^{m-1} (\tau^2 + \|h\|_2^2) \leq A_0(\tau^2 + \|h\|_2^2), \qquad \forall m \in \mathcal{I}_N, \ \forall j \in \mathcal{I}_q, \tag{1.70}$$

where $A_0 = \sqrt{2A_2}T$. The conclusion readily follows now. □

Remarks 1.23. Before closing this section, it is important to point out that the stability and the convergence of the finite-difference scheme has been established using a discrete and fractional form of the well-known energy method. This approach has been followed in some previous papers in order to analyze some Hamiltonian systems consisting of a single hyperbolic partial differential equation with fractional diffusion [33, 40, 55]. However, most of the hyperbolic models in which complex patterns appear, are non-Hamiltonian regimes consisting of two or more partial differential equations. In that sense, previous efforts useless in the investigation of such systems. We have proved rigorously that the present methodology is numerically efficient, and it has the advantage of being applicable to a wide class of systems consisting of various hyperbolic fractional partial differential equations.

2. Computational models

In Section 2.1, we describe an efficient computational implementation of the finite-difference method reported in the previous chapter. We show that the scheme can be implemented using algorithms based on matrix algebra, and describe the implementation in the two- and the three-dimensional cases. For convenience, Section 2.2 provides a simplified numerical scheme and its efficient computational implementation. For convenience, both computational techniques are described in the form of algorithms. Section 2.3 shows some illustrative simulations in the two- and the three-dimensional scenarios. Concretely, we exhibit the formation of Turing patterns in those systems.

2.1 Computational model

In this section, we will describe an efficient computational implementation of the finite-difference method (1.43). To that end, we will focus our attention on the explicit equations (1.44), and we will set $p = 3$ and $q = 2$. In order to simplify the notation, we will convey that $x = x_1$, $y = x_2$ and $z = x_3$, and that $u = u_1$ and $v = u_2$. Using these conventions and the hypothesis **(H)**, the mathematical model (1.7) can be rewritten as

$$\begin{aligned}
&\frac{\partial^2 u(x,t)}{\partial t^2} + \gamma \frac{\partial u(x,t)}{\partial t} = d\Delta^\alpha u(x,t) - \frac{\partial G_1(u(x,t),v(x,t))}{\partial u}, \qquad \forall (x,t) \in \Omega,\\
&\frac{\partial^2 v(x,t)}{\partial t^2} + \gamma \frac{\partial v(x,t)}{\partial t} = d\Delta^\alpha v(x,t) - \frac{\partial G_2(u(x,t),v(x,t))}{\partial v}, \qquad \forall (x,t) \in \Omega,\\
&\text{such that} \begin{cases} u(x,0) = \phi_1(x) \quad v(x,0) = \phi_2(x), & \forall x \in B,\\ \dfrac{\partial u(x,0)}{\partial t} = \psi_1(x), \quad \dfrac{\partial v(x,0)}{\partial t} = \psi_2(x), & \forall x \in B,\\ u(x,t) = v(x,t) = 0, & \forall (x,t) \in \partial B \times [0,T]. \end{cases}
\end{aligned} \tag{2.1}$$

Let also $k = k_1$, $l = k_2$ and $m = k_3$, let $j = (k,l,m)$ and agree that $x_k = x_{1,k}$, $y_l = x_{2,l}$ and $z_m = x_{3,m}$. Under the new conventions, if $(j,n) \in \mathcal{K} \times \mathcal{I}_{N-2}$ then the recursive formulas (1.44) can be rewritten equivalently as

$$\begin{cases}
\mathbf{u}_j^{n+2} = \mathbf{u}_j^{n+1} + \mathbf{u}_j^n - \mathbf{u}_j^{n-1} + r_1\left(\delta_x^{(\alpha)}\mathbf{u}_j^{n+1} + \delta_x^{(\alpha)}\mathbf{u}_j^n\right) - r_2 \dfrac{G_1(\mathbf{u}_j^{n+1},\mu_t\mathbf{v}_j^n) - G_1(\mathbf{u}_j^n,\mu_t\mathbf{v}_j^n)}{\mathbf{u}_j^{n+1} - \mathbf{u}_j^n} - r_3\left(\mathbf{u}_j^{n+1} - \mathbf{u}_j^n\right),\\
\mathbf{v}_j^{n+2} = \mathbf{v}_j^{n+1} + \mathbf{v}_j^n - \mathbf{v}_j^{n-1} + r_1\left(\delta_x^{(\alpha)}\mathbf{v}_j^{n+1} + \delta_x^{(\alpha)}\mathbf{v}_j^n\right) - r_2 \dfrac{G_2(\mu_t\mathbf{u}_j^n,\mathbf{v}_j^{n+1}) - G_2(\mu_t\mathbf{u}_j^n,\mathbf{v}_j^n)}{\mathbf{v}_j^{n+1} - \mathbf{v}_j^n} - r_3\left(\mathbf{v}_j^{n+1} - \mathbf{v}_j^n\right),
\end{cases} \tag{2.2}$$

where $r_1 = \tau^2 d$, $r_2 = 2\tau^2$ and $r_3 = 2\tau\gamma$. Meanwhile, the initial-boundary data will assume the form

$$\begin{cases} \overline{\mathbf{u}}_k^0 = \phi^u(x_k), & \overline{\mathbf{v}}_k^0 = \phi^v(x_k), \quad \forall k \in \mathcal{K}, \\ \overline{\mathbf{u}}_k^1 = \psi^u(x_k), & \overline{\mathbf{v}}_k^1 = \psi^v(x_k), \quad \forall k \in \mathcal{K}, \\ \overline{\mathbf{u}}_k^2 = \chi^u(x_k), & \overline{\mathbf{v}}_k^2 = \chi^v(x_k), \quad \forall k \in \mathcal{K}, \\ \overline{\mathbf{u}}_k^n = \overline{\mathbf{v}}_k^n = 0, & \qquad \forall (k,n) \in \partial\mathcal{K} \times \overline{\mathcal{I}}_N. \end{cases} \tag{2.3}$$

Before we propose the computational implementation of the system (2.2), it is important to point out that its extension to account for other values of q is straightforward. In what follows, for any $\alpha \in (1,2]$ and $n \in \mathbb{N}$, we consider the real matrices of sizes $(n+1)\times(n+1)$ defined by

$$H_n^\alpha = \begin{pmatrix} g_0^{(\alpha)} & g_{-1}^{(\alpha)} & g_{-2}^{(\alpha)} & g_{-3}^{(\alpha)} & \cdots & g_{-n}^{(\alpha)} \\ g_1^{(\alpha)} & g_0^{(\alpha)} & g_{-1}^{(\alpha)} & g_{-2}^{(\alpha)} & \cdots & g_{1-n}^{(\alpha)} \\ g_2^{(\alpha)} & g_1^{(\alpha)} & g_0^{(\alpha)} & g_{-1}^{(\alpha)} & \cdots & g_{2-n}^{(\alpha)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ g_n^{(\alpha)} & g_{n-1}^{(\alpha)} & g_{n-2}^{(\alpha)} & g_{n-3}^{(\alpha)} & \cdots & g_0^{(\alpha)} \end{pmatrix}. \tag{2.4}$$

We will enumerate the rows and columns of matrices beginning from 0, and not from 1. It is easy to see then that the component of H_n^α at the ith row and jth column is given by $[H_n^\alpha]_{i,j} = g_{i-j}^{(\alpha)}$, for all $(i,j) \in \overline{\mathcal{I}}_n \times \overline{\mathcal{I}}_n$. Moreover, the properties of the coefficients $(g_k^{(\alpha)})_{k=-\infty}^{\infty}$ summarized in Lemma 1.9 assure that the matrix H_n^α is Hermitian.

Let $\mathbf{w} \in \mathbb{R}^{M_1+1}\times\mathbb{R}^{M_2+1}\times\mathbb{R}^{M_3+1}$. If $k \in \overline{\mathcal{I}}_{M_1}$ then we agree that $\mathbf{w}_{k,\cdot,\cdot} \in \mathbb{R}^{M_2+1}\times\mathbb{R}^{M_3+1}$ represent the matrix

$$\mathbf{w}_{k,\cdot,\cdot} = \begin{pmatrix} \mathbf{w}_{k,0,0} & \mathbf{w}_{k,0,1} & \mathbf{w}_{k,0,2} & \cdots & \mathbf{w}_{k,0,M_3} \\ \mathbf{w}_{k,1,0} & \mathbf{w}_{k,1,1} & \mathbf{w}_{k,1,2} & \cdots & \mathbf{w}_{k,1,M_3} \\ \mathbf{w}_{k,2,0} & \mathbf{w}_{k,2,1} & \mathbf{w}_{k,2,2} & \cdots & \mathbf{w}_{k,2,M_3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_{k,M_2,0} & \mathbf{w}_{k,M_2,1} & \mathbf{w}_{k,M_2,2} & \cdots & \mathbf{w}_{k,M_2,M_3} \end{pmatrix}. \tag{2.5}$$

In other words, $\mathbf{w}_{k,\cdot,\cdot}$ is obtained from $\mathbf{w}$ by fixing the first component equal to k, and arranging naturally the entries as an $(M_2+1)\times(M_3+1)$ matrix. In an entirely similar fashion, if $l \in \overline{\mathcal{I}}_{M_2}$ and $m \in \overline{\mathcal{I}}_{M_3}$ then $\mathbf{w}_{\cdot,l,\cdot} \in \mathbb{R}^{M_1+1}\times\mathbb{R}^{M_3+1}$ and $\mathbf{w}_{\cdot,\cdot,m} \in \mathbb{R}^{M_1+1}\times\mathbb{R}^{M_2+1}$ are the matrices of sizes $(M_1+1)\times(M_3+1)$ and $(M_1+1)\times(M_2+1)$ given respectively by

$$\mathbf{w}_{\cdot,l,\cdot} = \begin{pmatrix} \mathbf{w}_{0,l,0} & \mathbf{w}_{0,l,1} & \mathbf{w}_{0,l,2} & \cdots & \mathbf{w}_{0,l,M_3} \\ \mathbf{w}_{1,l,0} & \mathbf{w}_{1,l,1} & \mathbf{w}_{1,l,2} & \cdots & \mathbf{w}_{1,l,M_3} \\ \mathbf{w}_{2,l,0} & \mathbf{w}_{2,l,1} & \mathbf{w}_{2,l,2} & \cdots & \mathbf{w}_{2,l,M_3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_{M_1,l,0} & \mathbf{w}_{M_1,l,1} & \mathbf{w}_{M_1,l,2} & \cdots & \mathbf{w}_{M_1,l,M_3} \end{pmatrix} \tag{2.6}$$

and

$$\mathbf{w}_{\cdot,\cdot,m} = \begin{pmatrix} \mathbf{w}_{0,0,m} & \mathbf{w}_{0,1,m} & \mathbf{w}_{0,2,m} & \cdots & \mathbf{w}_{0,M_2,m} \\ \mathbf{w}_{1,0,m} & \mathbf{w}_{1,1,m} & \mathbf{w}_{1,2,m} & \cdots & \mathbf{w}_{1,M_2,m} \\ \mathbf{w}_{2,0,m} & \mathbf{w}_{2,1,m} & \mathbf{w}_{2,2,m} & \cdots & \mathbf{w}_{2,M_2,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_{M_1,0,m} & \mathbf{w}_{M_1,1,m} & \mathbf{w}_{M_1,2,m} & \cdots & \mathbf{w}_{M_1,M_2,m} \end{pmatrix}. \tag{2.7}$$

We will describe next the implementation of the calculations involving the terms with fractional centered differences. To that end, we let $\mathbf{w}$ be any of $\mathbf{u}$ or $\mathbf{v}$ as before, and recall that (1.32) holds. The first term on the right-hand side of that expression can be equivalently rewritten as

$$\delta_x^{(\alpha)} \mathbf{w}_{k,l,m}^n = -\frac{1}{h_1^\alpha}\sum_{j=0}^{M_1} g_{k-j}^{(\alpha)} \mathbf{w}_{j,l,m}^n = -\frac{1}{h_1^\alpha}\sum_{j=0}^{M_1} [H_{M_1}^\alpha]_{kj} [\mathbf{w}_{\cdot,\cdot,m}^n]_{jl} = -h_1^{-\alpha}[H_{M_1}^\alpha \mathbf{w}_{\cdot,\cdot,m}]_{k,l}. \tag{2.8}$$

Obviously, the product inside of the parenthesis at the right-hand side of this expression is the usual product of matrices. Similarly, it is easy to check that $\delta_y^{(\alpha)}\mathbf{w}_{k,l,m}^n = -h_2^{-\alpha}[H_{M_2}^\alpha \mathbf{w}_{k,\cdot,\cdot}^n]_{l,m}$ and $\delta_z^{(\alpha)}\mathbf{w}_{k,l,m}^n = -h_3^{-\alpha}[\mathbf{w}_{k,\cdot,\cdot}^n H_{M_3}^\alpha]_{l,m}$. Combining these expressions and representing the discrete fractional Laplacian operator simply by $\delta^{(\alpha)}$, it follows that

$$\delta^{(\alpha)}\mathbf{w}_{k,l,m}^n = -h_1^{-\alpha}[H_{M_1}^\alpha \mathbf{w}_{\cdot,\cdot,m}^n]_{k,l} - h_2^{-\alpha}[H_{M_2}^\alpha \mathbf{w}_{k,\cdot,\cdot}^n]_{l,m} - h_3^{-\alpha}[\mathbf{w}_{k,\cdot,\cdot}^n H_{M_3}^\alpha]_{l,m}, \tag{2.9}$$

for each $(k,l,m) \in \overline{\mathcal{K}}$.

We will require the matrices $H_x = r_1 h_1^{-\alpha} H_{M_1}^\alpha$, $H_y = r_1 h_2^{-\alpha} H_{M_2}^\alpha$ and $H_z = r_1 h_3^{-\alpha} H_{M_3}^\alpha$. Using this nomenclature, the finite-difference system (2.2) may be rewritten equivalently as

$$\begin{cases} \mathbf{u}_{k,l,m}^{n+2} = \mathbf{u}_{k,l,m}^{n+1} + \mathbf{u}_{k,l,m}^{n} - \mathbf{u}_{k,l,m}^{n-1} - r_2 \dfrac{G_1(\mathbf{u}_{k,l,m}^{n+1}, \mu_t \mathbf{v}_{k,l,m}^n) - G_1(\mathbf{u}_{k,l,m}^{n}, \mu_t \mathbf{v}_{k,l,m}^n)}{\mathbf{u}_{k,l,m}^{n+1} - \mathbf{u}_{k,l,m}^{n}} - r_3 \left(\mathbf{u}_{k,l,m}^{n+1} - \mathbf{u}_{k,l,m}^{n}\right), \\ \qquad - \left([H_x \mathbf{u}_{\cdot,\cdot,m}^{n+1}]_{k,l} + [H_y \mathbf{u}_{k,\cdot,\cdot}^{n+1}]_{l,m} + [\mathbf{u}_{k,\cdot,\cdot}^{n+1} H_z]_{l,m}\right) \\ \qquad - \left([H_x \mathbf{u}_{\cdot,\cdot,m}^{n}]_{k,l} + [H_y \mathbf{u}_{k,\cdot,\cdot}^{n}]_{l,m} + [\mathbf{u}_{k,\cdot,\cdot}^{n} H_z]_{l,m}\right), \\ \mathbf{v}_{k,l,m}^{n+2} = \mathbf{v}_{k,l,m}^{n+1} + \mathbf{v}_{k,l,m}^{n} - \mathbf{v}_{k,l,m}^{n-1} - r_2 \dfrac{G_1(\mu_t \mathbf{u}_{k,l,m}^{n}, \mathbf{v}_{k,l,m}^{n+1}) - G_1(\mu_t \mathbf{u}_{k,l,m}^{n}, \mathbf{v}_{k,l,m}^n)}{\mathbf{v}_{k,l,m}^{n+1} - \mathbf{v}_{k,l,m}^{n}} - r_3 \left(\mathbf{v}_{k,l,m}^{n+1} - \mathbf{v}_{k,l,m}^{n}\right), \\ \qquad - \left([H_x \mathbf{v}_{\cdot,\cdot,m}^{n+1}]_{k,l} + [H_y \mathbf{v}_{k,\cdot,\cdot}^{n+1}]_{l,m} + [\mathbf{v}_{k,\cdot,\cdot}^{n+1} H_z]_{l,m}\right) \\ \qquad - \left([H_x \mathbf{v}_{\cdot,\cdot,m}^{n}]_{k,l} + [H_y \mathbf{v}_{k,\cdot,\cdot}^{n}]_{l,m} + [\mathbf{v}_{k,\cdot,\cdot}^{n} H_z]_{l,m}\right), \end{cases} \tag{2.10}$$

for each $(k,l,m,n) \in \overline{\mathcal{K}} \times \mathcal{I}_{N-2}$.

In what follows, we will describe the computer implementation when $M = M_1 = M_2 = M_3$, $a = a_1 = a_2 = a_3$ and $b = b_1 = b_2 = b_3$. In such case, we observe that $h = h_1 = h_2 = h_3$ and $H = H_x = H_y = H_z$. Let $n \in \mathcal{I}_{N-2}$ and $w = u, v$, and define the three-dimensional real arrays R^w, S^w and T^w of sizes $(M+1)\times(M+1)\times(M+1)$, by

$$R_{k,l,m}^w = [H\mathbf{w}_{\cdot,\cdot,m}^n]_{k,l}, \qquad S_{k,l,m}^w = [H\mathbf{w}_{k,\cdot,\cdot}^n]_{l,m}, \qquad T_{k,l,m}^w = [\mathbf{w}_{k,\cdot,\cdot}^n H]_{l,m}, \tag{2.11}$$

for all $(k,l,m) \in \overline{\mathcal{K}}$. For simplification purposes, we are obviating here the dependence of R^w, S^w and T^w on k. From these set of identities, it readily follows that for each $(k,l,m) \in \overline{\mathcal{K}}$,

$$R_{\cdot,\cdot,m}^w = H\mathbf{w}_{\cdot,\cdot,m}^n, \qquad S_{k,\cdot,\cdot}^w = H\mathbf{w}_{k,\cdot,\cdot}^n, \qquad T_{k,\cdot,\cdot}^w = \mathbf{w}_{k,\cdot,\cdot}^n H. \tag{2.12}$$

Under this circumstances, Algorithm 1 provides the description of our computational implementation of (2.10).

Remarks 2.1. Some important remarks on the computer implementation of Algorithm 1 must be mentioned.

1. Firstly, it is important to point out that our implementation was carried out having Fortran 95 in mind. However, the approach is also valid for other computer languages, like C++ or Matlab. We opted to use Fortran in view that if is a free computer language, and our experience tells us that it is usually faster than other languages.

2. Various code lines can be obviously executed in parallel. One example is the set of instructions between lines 15–20 of Algorithm 1. Also, lines 22 and 23 can be realized in parallel, as well as all the instruction in lines 24 and 25. All these facts have been exploited in our computer implementation of Algorithm 1.

3. The algorithm was coded in Gfortran on a Linux Mint 18.3 “Sylvia” MATE distribution, and the use of Fortran commands for matrix multiplication has been crucial to speed up our implementation. In turn, the parallelization was carried out naturally using the package `OpenMP`.

Algorithm 1: Iterative algorithm to solve the discrete problem (1.43).

```
1  Initialize parameters;
2  Define matrix $H$;
3  Set $\mathbf{u}^0_{k,l,m} \leftarrow \phi^u_{k,l,m}$; $\mathbf{v}^0_{k,l,m} \leftarrow \phi^v_{k,l,m}$;
4  Set $\mathbf{u}^1_{k,l,m} \leftarrow \psi^u_{k,l,m}$; $\mathbf{v}^1_{k,l,m} \leftarrow \psi^v_{k,l,m}$;
5  Set $\mathbf{u}^2_{k,l,m} \leftarrow \chi^u_{k,l,m}$; $\mathbf{v}^2_{k,l,m} \leftarrow \chi^v_{k,l,m}$;
6  for $j = 0 : M$ do
7      for $w = u, v$ do
8          Set $R^w_{\cdot,\cdot,j} = H\mathbf{w}^1_{\cdot,\cdot,j}$;
9          Set $S^w_{j,\cdot,\cdot} = H\mathbf{w}^1_{j,\cdot,\cdot}$;
10         Set $T^w_{j,\cdot,\cdot} = \mathbf{w}^1_{j,\cdot,\cdot}H$;
11     end
12 end
13 for $n = 1 : N-2$ do
14     Set $W_{k,l,m} \leftarrow \dfrac{G_1(\mathbf{u}^{n+1}_{k,l,m}, \mu_t\mathbf{v}^n_{k,l,m}) - G_1(\mathbf{u}^n_{k,l,m}, \mu_t\mathbf{v}^n_{k,l,m})}{\mathbf{u}^{n+1}_{k,l,m} - \mathbf{u}^n_{k,l,m}}$;
         $Z_{k,l,m} \leftarrow \dfrac{G_2(\mu_t\mathbf{u}^n_{k,l,m}, \mathbf{v}^{n+1}_{k,l,m}) - G_2(\mu_t\mathbf{u}^n_{k,l,m}, \mathbf{v}^n_{k,l,m})}{\mathbf{v}^{n+1}_{k,l,m} - \mathbf{v}^n_{k,l,m}}$;
15     for $j = 0 : M$ do
16         for $w = u, v$ do
17             Set $r^w_{\cdot,\cdot,j} = H\mathbf{w}^{n+1}_{\cdot,\cdot,j}$;
18             Set $s^w_{j,\cdot,\cdot} = H\mathbf{w}^{n+1}_{j,\cdot,\cdot}$;
19             Set $t^w_{j,\cdot,\cdot} = \mathbf{w}^{n+1}_{j,\cdot,\cdot}H$;
20         end
21     end
22     Calculate $\mathbf{u}^{n+2} = \mathbf{u}^{n+1} + \mathbf{u}^n - \mathbf{u}^{n-1} - r_3(\mathbf{u}^{n+1} - \mathbf{u}^n) - r_2W - r^u - s^u - t^u - R^u - S^u - T^u$;
23     Calculate $\mathbf{v}^{n+2} = \mathbf{v}^{n+1} + \mathbf{v}^n - \mathbf{v}^{n-1} - r_3(\mathbf{v}^{n+1} - \mathbf{v}^n) - r_2Z - r^v - s^v - t^v - R^v - S^v - T^v$;
24     Set $R^u \leftarrow r^u$; $S^u \leftarrow s^u$; $T^u \leftarrow t^u$;
25     Set $R^v \leftarrow r^v$; $S^v \leftarrow s^v$; $T^v \leftarrow t^v$;
26 end
```

4. In the calculation of the coefficients $(g^{(\alpha)}_k)^{\infty}_{k=-\infty}$, we employed the following recursive formula:

$$\begin{cases} g^{(\alpha)}_0 & = & \dfrac{\Gamma(\alpha+1)}{\Gamma(\alpha/2+1)^2}, \\ g^{(\alpha)}_{k+1} & = & \left(1 - \dfrac{\alpha+1}{\alpha/2+k+1}\right) g^{(\alpha)}_k, \qquad \forall k \in \mathbb{N} \cup \{0\}. \end{cases} \tag{2.13}$$

2.2 Simplified model

The purpose of this section is to provide a simpler numerical model to solve the problem under investigation. Using the discrete nomenclature, a simpler finite-difference method to approximate the solutions of (1.7) is described by the following discrete system of algebraic equations, for each $(j,k) \in \overline{\mathcal{J}} \times \overline{\mathcal{I}}_{K-1}$:

$$\begin{gathered} \tau_u \delta^{(2)}_t \mathbf{u}^k_j + \delta^{(1)}_t \mathbf{u}^k_j = F(\mathbf{u}^k_j, \mathbf{v}^k_j) + d_u \delta^{(\alpha_1)} \mathbf{u}^k_j, \\ \tau_v \delta^{(2)}_t \mathbf{v}^k_j + \delta^{(1)}_t \mathbf{v}^k_j = G(\mathbf{u}^k_j, \mathbf{v}^k_j) + d_v \delta^{(\alpha_2)} \mathbf{v}^k_j, \\ \text{such that } \begin{cases} \mathbf{u}^0_j = \phi^u_j, \quad \mathbf{v}^0_j = \phi^v_j, & \forall j \in \overline{\mathcal{J}}, \\ \delta^{(1)}_t \mathbf{u}^0_j = \psi^u_j, \quad \delta^{(1)}_t \mathbf{v}^0_j = \psi^v_j, & \forall j \in \overline{\mathcal{J}}. \end{cases} \end{gathered} \tag{2.14}$$

Remark 2.2. Various facts must be noted.

1. Firstly, we must mention that an extension of the method (1.43) can be readily provided to solve numerically the system (1.7). This task only requires of introducing suitable discrete nomenclature. We do not tackle this problem in this thesis, in view that such extension is an easy task.

2. Also, it is important to note that the general difference equations of (1.43) can be solved exactly for $\mathbf{u}_j^{k+1}$ and $\mathbf{v}_j^{k+1}$, for each $j \in \overline{\mathcal{J}}$ and $k \in \overline{\mathcal{I}}_{K-1}$. Indeed, after doing some algebraic calculations, it is easy to check that

$$\mathbf{u}_j^{k+1} = r_1^u \mathbf{u}_j^k - r_2^u \mathbf{u}_j^{k-1} + r_3^u \left[F(\mathbf{u}_j^k, \mathbf{v}_j^k) + \delta^{(\alpha_1)} \mathbf{u}_j^k \right], \tag{2.15}$$

$$\mathbf{v}_j^{k+1} = r_1^v \mathbf{v}_j^k - r_2^v \mathbf{v}_j^{k-1} + r_3^v \left[G(\mathbf{u}_j^k, \mathbf{v}_j^k) + \delta^{(\alpha_2)} \mathbf{v}_j^k \right], \tag{2.16}$$

for each $(j,k) \in \overline{\mathcal{J}} \times \overline{\mathcal{I}}_{K-1}$. Here, for each $w = u, v$,

$$r_1^w = \frac{4\tau_w}{2\tau_w + \tau}, \qquad r_2^w = \frac{2\tau_w - \tau}{2\tau_w + \tau}, \qquad r_3^w = \frac{2\tau^2}{2\tau_w + \tau}. \tag{2.17}$$

3. In light of the previous remark, the discrete scheme (1.43) is uniquely solvable for any set of initial conditions.

4. The consistency of the discrete operators employed in our discretization together with Lemma 1.10 show that the finite-difference scheme has a local truncation error of order $\mathcal{O}(\tau^2 + \|h\|_2^2)$ for functions $u, v \in \mathcal{C}_{x,t}^{5,4}(\overline{\Omega})$. Here, the symbol $\|\cdot\|_2$ represents the usual Euclidean norm.

5. Note that the method requires to know the approximations $\mathbf{u}^{-1}$ and $\mathbf{v}^{-1}$ at the time $t_{-1} = -\tau$. However, using the discrete initial velocities of (1.43), we readily obtain that $\mathbf{u}_j^{-1} = \mathbf{u}_j^1 - 2\tau\psi_j^u$ and $\mathbf{v}_j^{-1} = \mathbf{v}_j^1 - 2\tau\psi_j^v$. Letting $k = 0$ and substituting now into equations (2.15) and (2.16), respectively, we obtain

$$\mathbf{u}_j^1 = \frac{r_1^u \mathbf{u}_j^0 + 2r_2^u \tau \psi_j^u + r_3^u \left[F(\mathbf{u}_j^0, \mathbf{v}_j^0) + \delta^{(\alpha_1)} \mathbf{u}_j^0 \right]}{1 + r_2^u}, \tag{2.18}$$

$$\mathbf{v}_j^1 = \frac{r_1^v \mathbf{v}_j^0 + 2r_2^v \tau \psi_j^v + r_3^v \left[G(\mathbf{u}_j^0, \mathbf{v}_j^0) + \delta^{(\alpha_2)} \mathbf{v}_j^0 \right]}{1 + r_2^v}, \tag{2.19}$$

for all $j \in \overline{\mathcal{J}}$.

Some computational simplifications are readily at hand if we define the matrices

$$\left\{ \begin{array}{ll} H_x^u = r_3^u h_1^{-\alpha_1} H_M^{\alpha_1}, & H_x^v = r_3^v h_1^{-\alpha_2} H_M^{\alpha_2}, \\ H_y^u = r_3^u h_2^{-\alpha_1} H_N^{\alpha_1}, & H_y^v = r_3^v h_2^{-\alpha_2} H_N^{\alpha_2}, \\ H_z^u = r_3^u h_3^{-\alpha_1} H_P^{\alpha_1}, & H_z^v = r_3^v h_3^{-\alpha_2} H_P^{\alpha_2}. \end{array} \right. \tag{2.20}$$

Using this nomenclature, the finite-difference equations (2.15)-(2.16) way be rewritten equivalently as

$$\left\{ \begin{aligned} \mathbf{u}_{m,n,p}^{k+1} &= r_1^u \mathbf{u}_{m,n,p}^k - r_2^u \mathbf{u}_{m,n,p}^{k-1} + r_3^u F(\mathbf{u}_{m,n,p}^k, \mathbf{v}_{m,n,p}^k) \\ &\qquad - [H_x^u \mathbf{u}_{\cdot,\cdot,p}^k]_{m,n} - [H_y^u \mathbf{u}_{m,\cdot,\cdot}^k]_{n,p} - [\mathbf{u}_{m,\cdot,\cdot}^k H_z^u]_{n,p}, \\ \mathbf{v}_{m,n,p}^{k+1} &= r_1^v \mathbf{v}_{m,n,p}^k - r_2^v \mathbf{v}_{m,n,p}^{k-1} + r_3^v G(\mathbf{u}_{m,n,p}^k, \mathbf{v}_{m,n,p}^k) \\ &\qquad - [H_x^v \mathbf{v}_{\cdot,\cdot,p}^k]_{m,n} - [H_y^v \mathbf{v}_{m,\cdot,\cdot}^k]_{n,p} - [\mathbf{v}_{m,\cdot,\cdot}^k H_z^v]_{n,p}, \end{aligned} \right. \tag{2.21}$$

Algorithm 2: Iterative algorithm to solve problem (2.21)-(2.22).

```
Initialize parameters;
Define matrices $H^u$; $H^v$;
Set $\mathbf{u}^0_{m,n,p} \leftarrow \phi^u_{m,n,p}$; $\mathbf{v}^0_{m,n,p} \leftarrow \phi^0_{m,n,p}$;
Set $W \leftarrow F(\mathbf{u}^0, \mathbf{v}^0)$; $Z \leftarrow G(\mathbf{u}^0, \mathbf{v}^0)$;
for $l = 0 : M$ do
    for $w = u, v$ do
        Set $R^w_{\cdot,\cdot,l} = H^w \mathbf{w}^k_{\cdot,\cdot,l}$;
        Set $S^w_{l,\cdot,\cdot} = H^w \mathbf{w}^k_{l,\cdot,\cdot}$;
        Set $T^w_{l,\cdot,\cdot} = \mathbf{w}^k_{l,\cdot,\cdot} H^w$;
    end
end
Calculate $\mathbf{u}^1 = \dfrac{r^u_1 \mathbf{u}^0 + 2r^u_2 \tau \psi^u + r^u_3 W - R^u - S^u - T^u}{1 + r^u_2}$;
Calculate $\mathbf{v}^1 = \dfrac{r^v_1 \mathbf{v}^0 + 2r^v_2 \tau \psi^v + r^v_3 Z - R^v - S^v - T^v}{1 + r^v_2}$;
for $k = 1 : K - 1$ do
    Set $W \leftarrow F(\mathbf{u}^k, \mathbf{v}^k)$; $Z \leftarrow G(\mathbf{u}^k, \mathbf{v}^k)$;
    for $l = 0 : M$ do
        for $w = u, v$ do
            Set $R^w_{\cdot,\cdot,l} = H^w \mathbf{w}^k_{\cdot,\cdot,l}$;
            Set $S^w_{l,\cdot,\cdot} = H^w \mathbf{w}^k_{l,\cdot,\cdot}$;
            Set $T^w_{l,\cdot,\cdot} = \mathbf{w}^k_{l,\cdot,\cdot} H^w$;
        end
    end
    Calculate $\mathbf{u}^{k+1} = r^u_1 \mathbf{u}^k - r^u_2 \mathbf{u}^{k-1} + r^u_3 W - R^u - S^u - T^u$;
    Calculate $\mathbf{v}^{k+1} = r^v_1 \mathbf{v}^k - r^v_2 \mathbf{v}^{k-1} + r^v_3 W - R^v - S^v - T^v$;
end
```

for each $(m,n,p,k) \in \overline{\mathcal{J}} \times \mathcal{I}_{K-1}$. Meanwhile, the initial data satisfy the following identities, for each $(m,n,p) \in \overline{\mathcal{J}}$:

$$\begin{cases} \mathbf{u}^1_{m,n,p} = \dfrac{1}{1+r^u_2} \left(r^u_1 \mathbf{u}^0_{m,n,p} + 2r^u_2 \tau \psi^u_{m,n,p} + r^u_3 F(\mathbf{u}^0_{m,n,p}, \mathbf{v}^0_{m,n,p}) \right. \\ \qquad \left. -[H^u_x \mathbf{u}^0_{\cdot,\cdot,p}]_{m,n} - [H^u_y \mathbf{u}^0_{m,\cdot,\cdot}]_{n,p} - [\mathbf{u}^0_{m,\cdot,\cdot} H^u_z]_{n,p} \right), \\ \mathbf{v}^1_{m,n,p} = \dfrac{1}{1+r^v_2} \left(r^v_1 \mathbf{v}^0_{m,n,p} + 2r^v_2 \tau \psi^v_{m,n,p} + r^v_3 G(\mathbf{u}^0_{m,n,p}, \mathbf{v}^0_{m,n,p}) \right. \\ \qquad \left. -[H^v_x \mathbf{v}^0_{\cdot,\cdot,p}]_{m,n} - [H^v_y \mathbf{v}^0_{m,\cdot,\cdot}]_{n,p} - [\mathbf{v}^0_{m,\cdot,\cdot} H^v_z]_{n,p} \right), \\ \mathbf{u}^0_{m,n,p} = \phi^u_{m,n,p}, \\ \mathbf{v}^0_{m,n,p} = \phi^v_{m,n,p}. \end{cases} \tag{2.22}$$

In what follows, we will describe the computer implementation for the case when $M = N = P$, $a = a_1 = a_2 = a_3$ and $b = b_1 = b_2 = b_3$. In such case, we observe that $h = h_1 = h_2 = h_3$ and $H^w = H^w_x = H^w_y = H^w_z$, for each $w = u, v$. Let $k \in \mathcal{I}_{K-1}$ and $w = u, v$, and define the three-dimensional real arrays R^w, S^w and T^w of sizes $(M+1) \times (M+1) \times (M+1)$, by

$$R^w_{m,n,p} = [H^w \mathbf{w}^k_{\cdot,\cdot,p}]_{m,n}, \qquad S^w_{m,n,p} = [H^w \mathbf{w}^k_{m,\cdot,\cdot}]_{n,p}, \qquad T^w_{m,n,p} = [\mathbf{w}^k_{m,\cdot,\cdot} H^w]_{n,p}, \tag{2.23}$$

for all $(m,n,p) \in \overline{\mathcal{J}}$. For simplification purposes, we are obviating here the dependence of R^w, S^w and

T^w on k. From these set of identities, it readily follows that

$$R^w_{\cdot,\cdot,p} = H^w \mathbf{w}^k_{\cdot,\cdot,p}, \qquad S^w_{m,\cdot,\cdot} = H^w \mathbf{w}^k_{m,\cdot,\cdot}, \qquad T^w_{m,\cdot,\cdot} = \mathbf{w}^k_{m,\cdot,\cdot} H^w, \tag{2.24}$$

for each $(m, n, p) \in \overline{\mathcal{J}}$.

Moreover, in our computational implementation, we will employ the matrices W and Z of sizes $(M+1) \times (M+1) \times (M+1)$ defined component-wise by

$$W_{m,n,p} = F(\mathbf{u}^k_{m,n,p}, \mathbf{v}^k_{m,n,p}), \tag{2.25}$$

$$Z_{m,n,p} = G(\mathbf{u}^k_{m,n,p}, \mathbf{v}^k_{m,n,p}), \tag{2.26}$$

for each $(m, n, p) \in \overline{\mathcal{J}}$. Under this circumstances, Algorithm 2 provides the algorithmic description of the finite-difference method (2.21)-(2.22). It is important to point out that our implementation was carried out having Fortran 95 in mind. However, the approach is also valid for other computer languages, like C++ or Matlab. For convenience, Appendix A provides a rough implementation of this algorithm in Fortran 95.

2.3 Computer simulations

The present section is devoted to provide some illustrative computer simulations of the methods proposed in the previous sections. We will consider firstly the case of the non-variational scheme (1.43), and provide some three-dimensional simulations to the investigation of Turing patterns in chemical systems of inhibitor-activator substances. More concretely, consider the system

$$\begin{gathered}
\tau_u \frac{\partial^2 u(x,t)}{\partial t^2} + \frac{\partial u(x,t)}{\partial t} = u(x,t) - av(x,t) + bu(x,t)v(x,t) - [u(x,t)]^3 + d_u \nabla^{\alpha_1} u(x,t), \quad \forall (x,t) \in \Omega, \\
\tau_v \frac{\partial^2 v(x,t)}{\partial t^2} + \frac{\partial v(x,t)}{\partial t} = u(x,t) - cv(x,t) + d_v \nabla^{\alpha_2} v(x,t), \quad \forall (x,t) \in \Omega, \\
\text{such that} \left\{ \begin{array}{lll} u(x,0) = \phi^u(x), & v(x,0) = \phi^v(x), & \forall x \in B, \\ \dfrac{\partial u}{\partial t}(x,0) = \psi^u(x), & \dfrac{\partial v}{\partial t}(x,0) = \psi^v(x), & \forall x \in B. \end{array} \right.
\end{gathered} \tag{2.27}$$

It is clear that this is a particular form of (1.7), using $q = 2$, $u = u_1$, $v = u_2$, $\tau_u = \tau_1$, $\tau_v = \tau_2$, $\gamma_1 = \gamma_2 = 1$, $\phi^u = \phi_1$, $\phi^v = \phi_2$, $\psi^u = \psi_1$ and $\psi^v = \psi_2$. Meanwhile, the functions F_1 and F_2 are provided by (1.13) and (1.14), respectively. Clearly, the functions $G_1, G_2 : \mathbb{R}^2 \to \mathbb{R}$ defined below satisfy the hypothesis **(H)**:

$$G_1(u,v) = \tfrac{1}{2}u^2 - auv + \tfrac{1}{2}bu^2 v - \tfrac{1}{4}u^4, \quad \forall u, v \in \mathbb{R}, \tag{2.28}$$

$$G_2(u,v) = uv - \tfrac{1}{2}cv^2, \quad \forall u, v \in \mathbb{R}. \tag{2.29}$$

In the following examples, we concentrate our attention on the solutions of the system (2.27). Our examples are motivated by a set of results reported in [44] for the two-dimensional version of the mathematical model considered here. Briefly, the authors of that work found out that, for any value of $\alpha_1, \alpha_2 \in (1, 2]$, Turing patterns appear in the system when the following parameter conditions are satisfied:

$$1 < c < a < a_T, \quad d > 1, \quad b < \sqrt{c(a-c)}. \tag{2.30}$$

Here, $a_T = \frac{1}{4}(d+c)^2 d^{-1}$. Motivated by this fact, we will fix the parameter values recorded in Table 2.1. We will only let b take on various values in $\mathbb{R}^+$.

Example 2.3. Consider the system (2.27) with the parameter values of Table 2.1 and $b = 0.5$. As initial profiles, we chose samples of a uniformly distributed random variable on the interval $[-0.03, 0.03]$, and used zero initial velocities. Figure 2.1 shows snapshots of the approximate solutions of the variable u of (1.7) versus x and y for $\alpha = 1.5$. The times (a) $t = 0$, (b) $t = 100$, (c) $t = 400$, (d) $t = 750$, (e) $t = 1500$ and (f) $t = 2500$ were used in this example, and the graphs have been normalized with

Parameter	p	a	c	d_u	d_v	τ_u	τ_v	α_1	α_2	Ω	T	h	τ
Example 2.3	2	7.45	5	1	20	1	1	1.5	1.5	$[0,200]^3$	2500	1	0.01
Example 2.4	3									$[0,100]^3$	2000		

Table 2.1: Set of fixed model and computational parameters employed in the simulations of Examples 2.3 and 2.4.

Parameter	a	c	d_u	d_v	τ_u	τ_v	α_1	α_2	Ω	T	h	τ
Value	7.45	5	1	20	1	1	1.5	1.5	$[0,100]^3$	2000	1	0.01

Table 2.2: Set of fixed model and computational parameters employed in the simulations of Example 2.6.

respect to the absolute maximum of the solution at each time. It is clear that he results are in good qualitative agreement with those obtained in [44]. Figures 2.2 and 2.3 provide similar results for the cases when $b = 1.5$ and $b = 2.5$, respectively. Again, a good agreement with the theory and simulations obtained in [44] is found in these graphs. $\square$

Our next example considers the three dimensional version of (2.27).

Example 2.4. Consider the problem (2.27) in three spatial dimensions, together with the set of model and computational parameters in Table 2.1. Additionally, we let $b = 0.5$. Figure 2.4 shows snapshots of the approximate solution obtained using the scheme (1.43) at various times, namely, (a) $t = 50$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The results depict the presence of complex patterns in this system. The results were obtained using a parallel implementation of our code in Fortran 95, and the visualizations were obtained using standard routines in Matlab. In particular, we employed the standard command `isosurface` with different values of the `isovalue` parameter to that end. More precisely, we used values of the parameter `isovalue` equal to 0.01. Figure 2.5 shows some x-, y- and z-cross sections of the solutions at each of those times. The solutions exhibit some complex patterns in this case. Figures 2.6 and 2.7 are similar to Figures 2.4 and 2.5 using now $b = 1.5$. Finally, Figures 2.8 and 2.9 show the same results for $b = 2.5$. In all cases, we note the presence of complex patters in the three-dimensional medium. $\square$

Remarks 2.5. As we mentioned at the end of Section 1.2, Examples 1.3–1.5 satisfy **(H)**. This fact was observed at the beginning of the present section for the model of Example 1.5, and we will show next that the other two examples satisfy this analytical condition.

- In the case of Example 1.3, a function satisfying the hypothesis is $G_1(u) = 1 - \cos(u)$, for each $u \in \mathbb{R}$.
- For Example 1.4 with the scaled reaction functions (1.11)-(1.12) the functions

$$G_1(u_1,u_2) = \frac{1}{S}u_2^2 \ln(u_2 + Su_1) - u_1u_2 + \frac{R}{2}u_1^2 - \frac{R}{2S}u_1^3, \quad \forall u_1, u_2 \in \mathbb{R}, \tag{2.31}$$

$$G_2(u_1,u_2) = Su_1u_2 - S^2u_1^2 \ln(u_2 + Su_1) - \frac{Q}{2}u_2^2, \quad \forall u_1, u_2 \in \mathbb{R}, \tag{2.32}$$

  satisfy the hypothesis **(H)**.

Also, we must mention that the simulations of Example 2.4 show the appearance of complex patterns. However, it is important to point out that the analytical prediction of patterns in the three-dimensional model (2.27) is still an open task of research. In that sense, the present computational model can be a useful tool in the resolution of such problem.

We provide next some illustrative applications of the finite-difference method (2.14). In the following examples, we concentrate our attention on the solutions of the system (1.7) with reaction functions

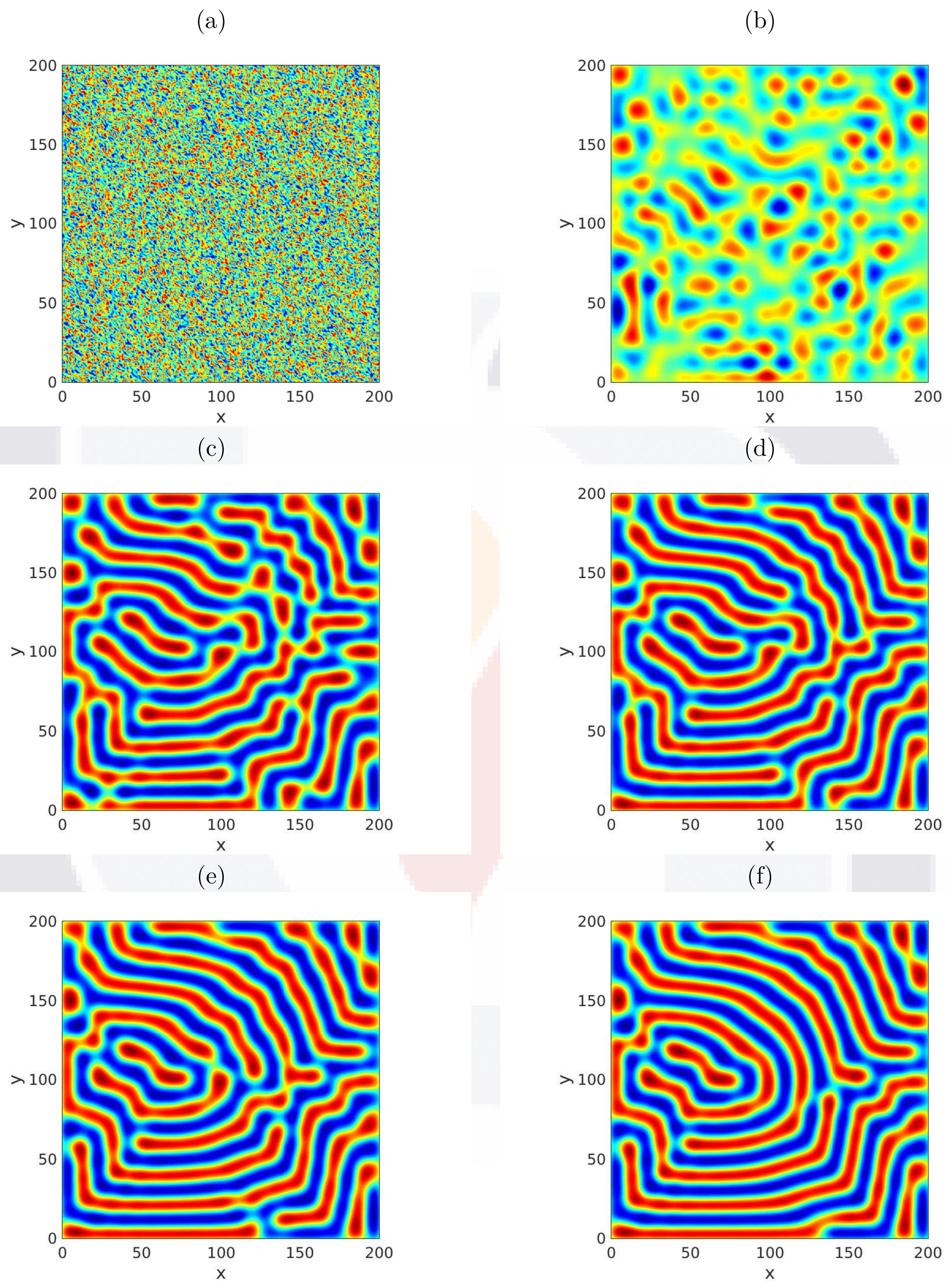


Figure 2.1: Snapshots of the approximate solutions of the variable u of (2.27) versus (x, y), using the parameters of Table 2.1 for Example 2.3, and $b = 0.5$. The times (a) $t = 100$, (b) $t = 400$, (c) $t = 1000$ and (d) $t = 2500$ were used in these simulations. The graphs were normalized with respect to the absolute maximum of the solution at each time.

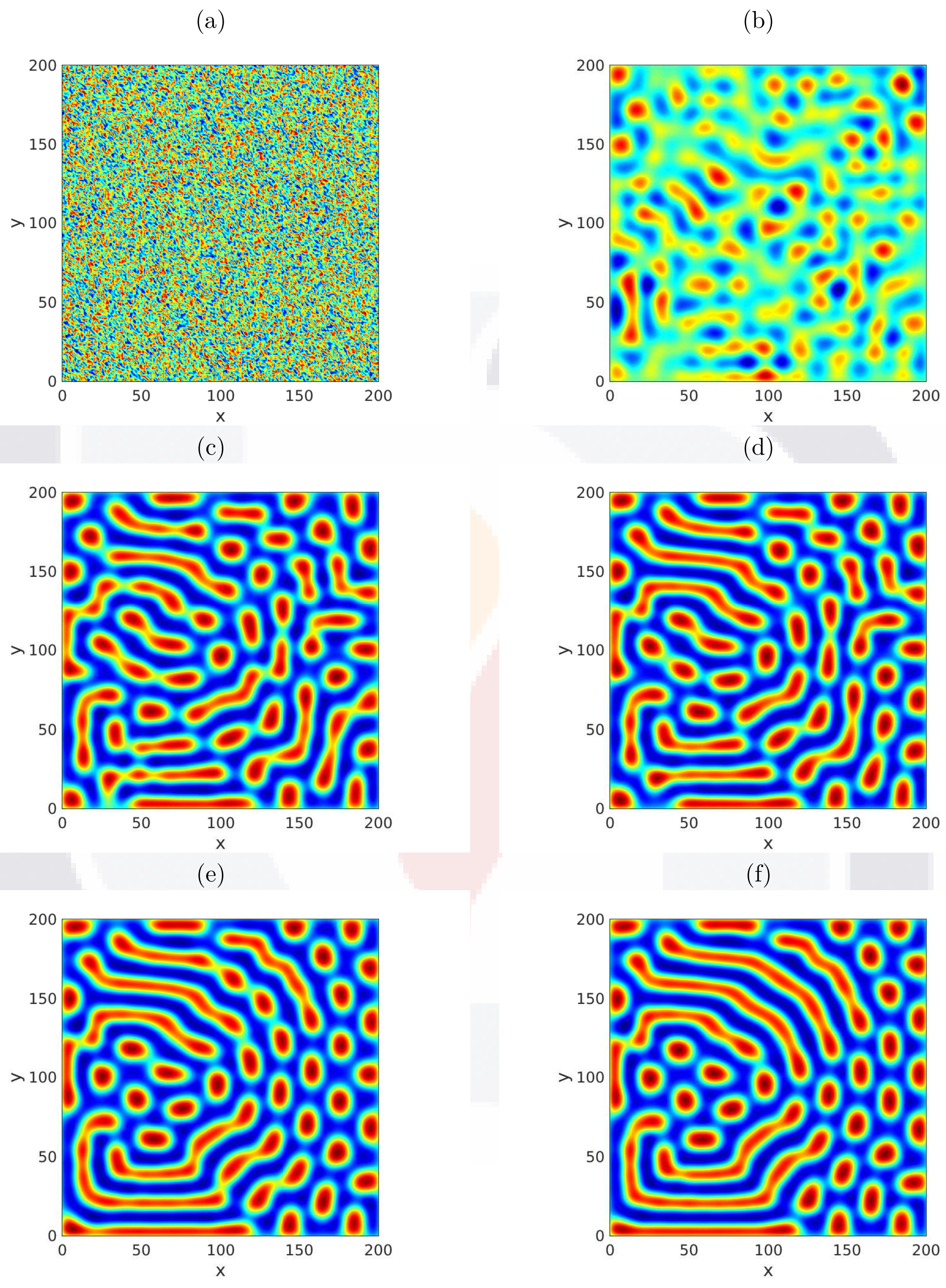


Figure 2.2: Snapshots of the approximate solutions of the variable u of (2.27) versus (x, y), using the parameters of Table 2.1 for Example 2.3, and $b = 1.5$. The times (a) $t = 100$, (b) $t = 400$, (c) $t = 1000$ and (d) $t = 2500$ were used in these simulations. The graphs were normalized with respect to the absolute maximum of the solution at each time.

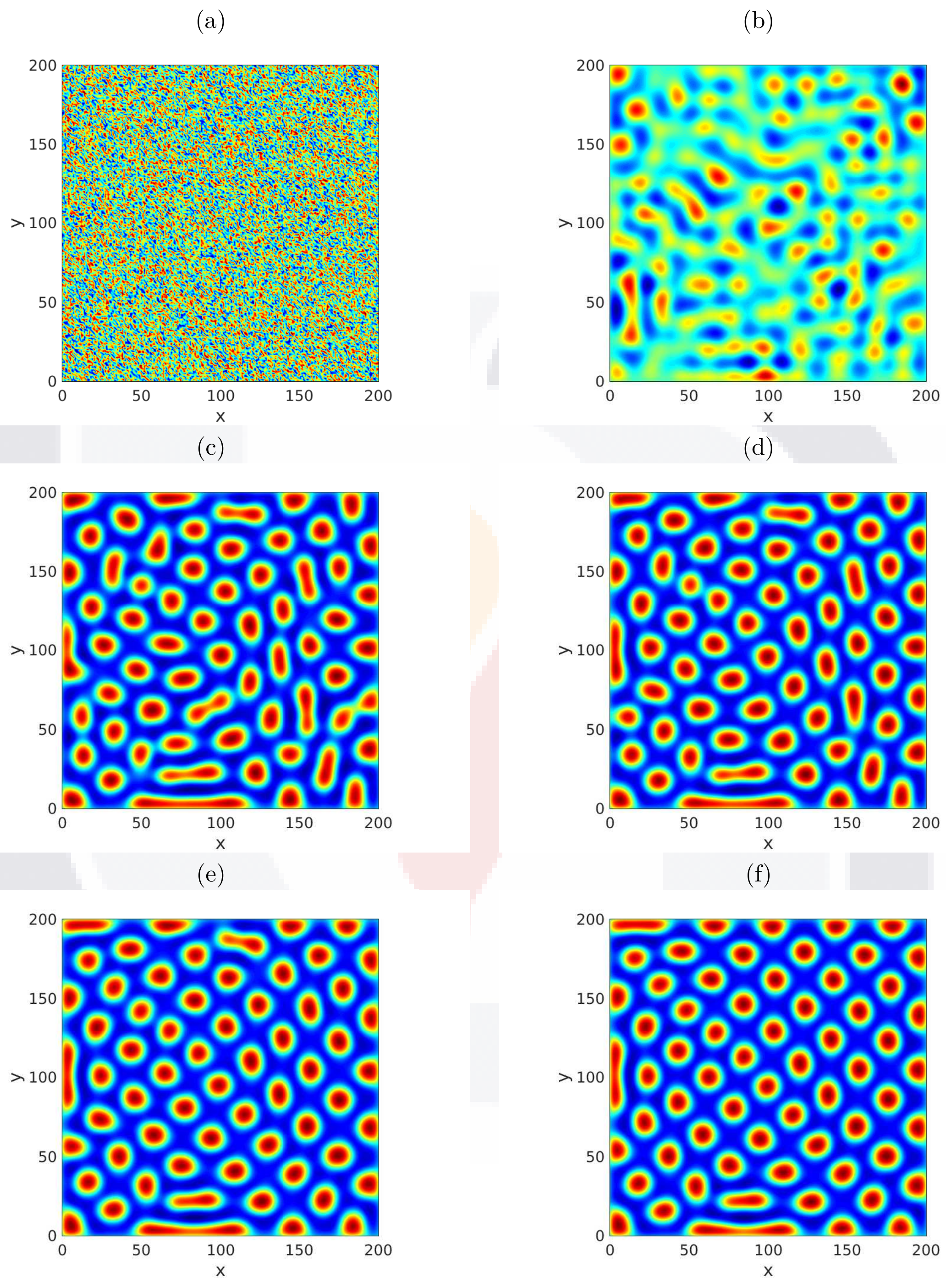


Figure 2.3: Snapshots of the approximate solutions of the variable u of (2.27) versus (x, y), using the parameters of Table 2.1 for Example 2.3, and $b = 2.5$. The times (a) $t = 100$, (b) $t = 400$, (c) $t = 1000$ and (d) $t = 2500$ were used in these simulations. The graphs were normalized with respect to the absolute maximum of the solution at each time.

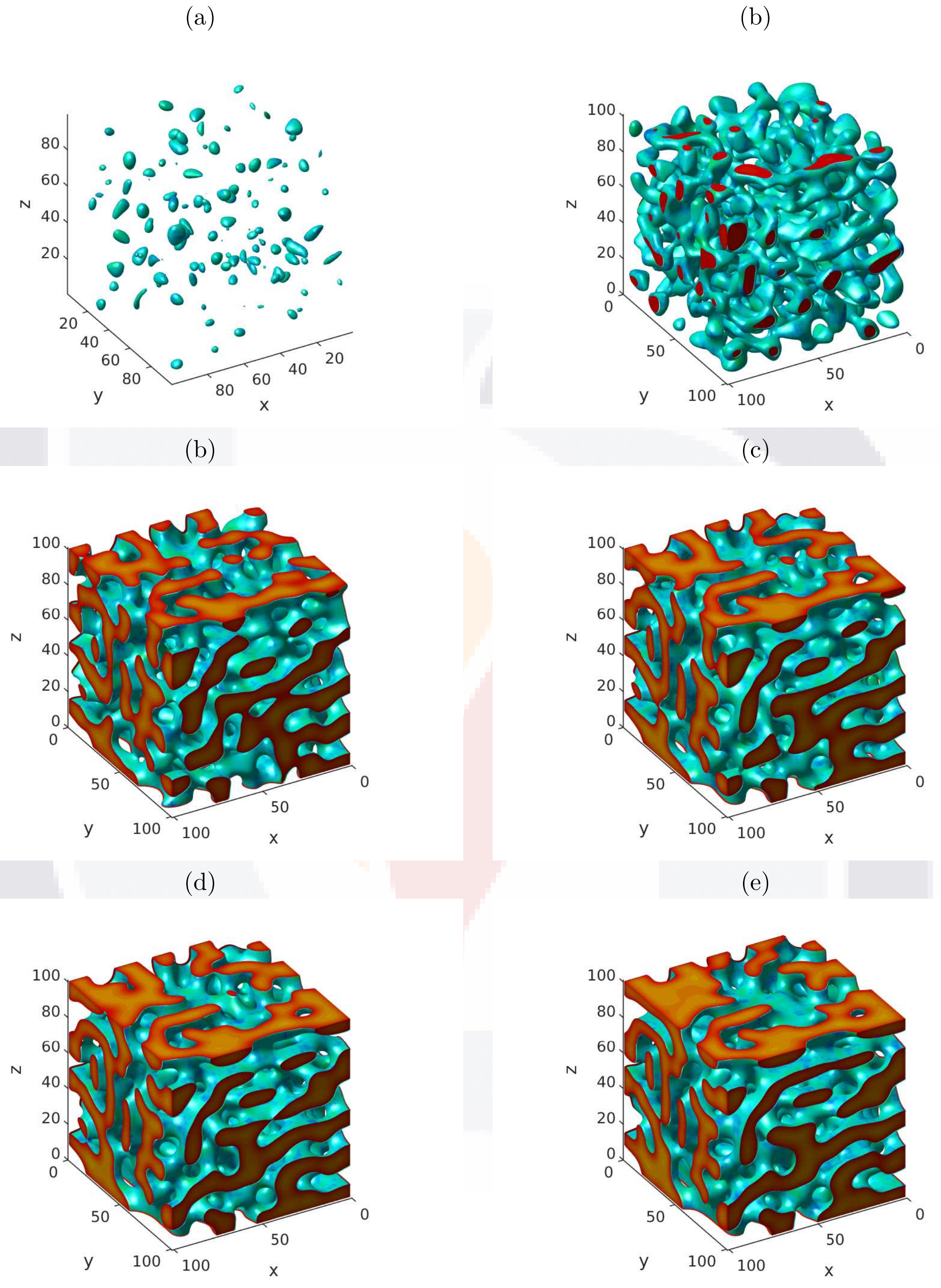
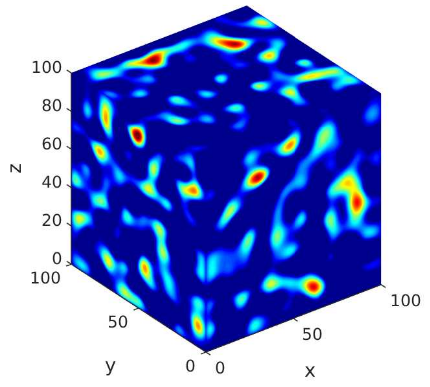
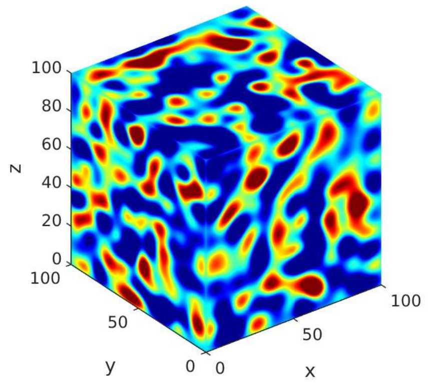


Figure 2.4: Approximation to the solutions of (2.27), using the finite-difference scheme (1.43) with the model and computational parameters in Table 2.1 for Example 2.4, and $b = 0.5$. Various times were considered, namely, (a) $t = 50$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The graphing subroutines were based on the function `isosurface` with values of the `isovalue` parameter equal to 0.01.

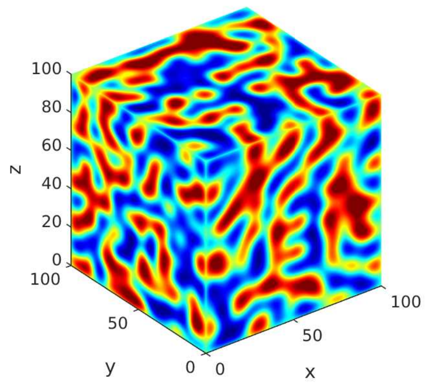
(a)



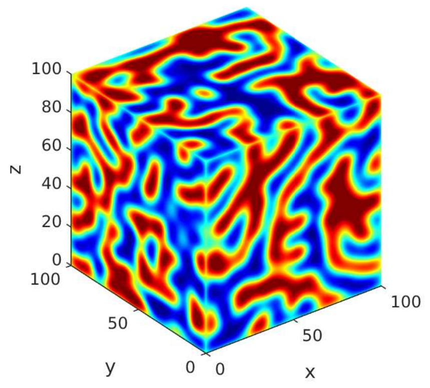
(b)



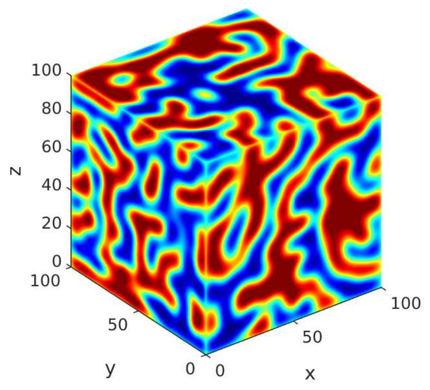
(b)



(c)



(d)



(e)

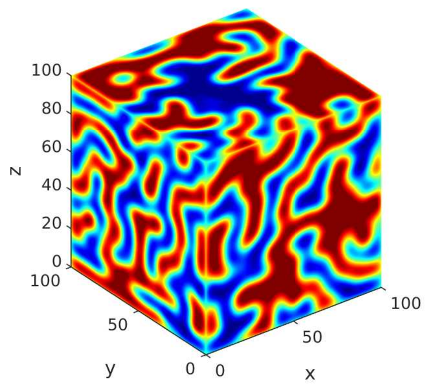
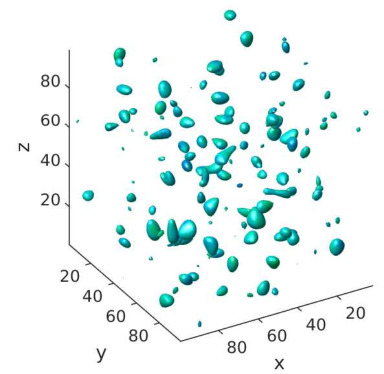
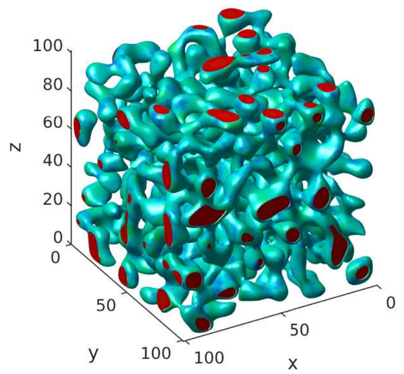


Figure 2.5: Approximation to some x-, y- and z-cross sections of the solutions of (2.27), using the finite-difference scheme (1.43) with the model and computational parameters in Table 2.1 for Example 2.4, and $b = 0.5$. Various times were considered, namely, (a) $t = 50$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$.

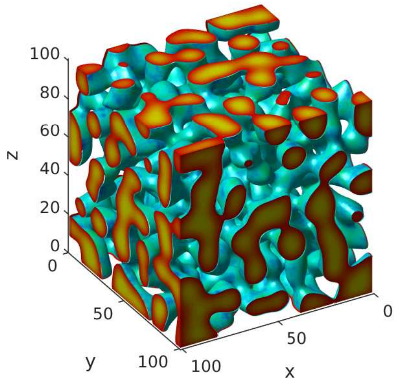
(a)



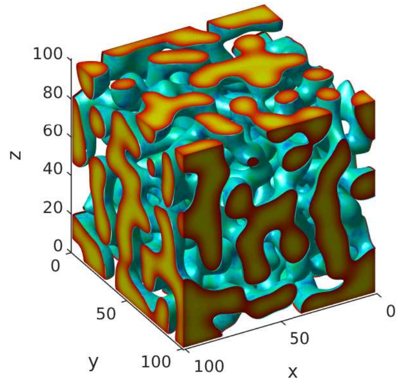
(b)



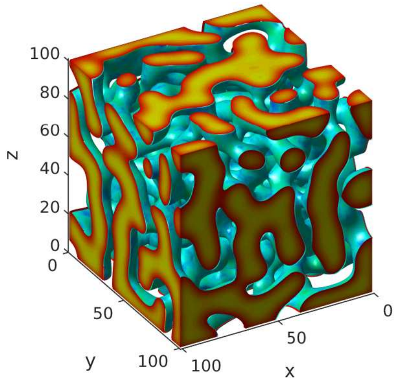
(b)



(c)



(d)



(e)

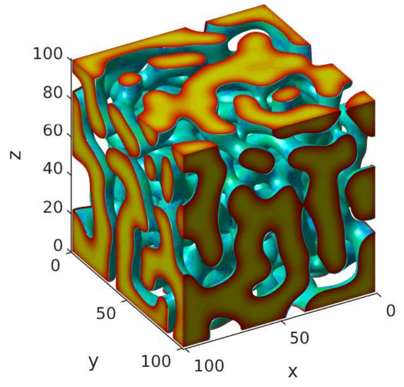


Figure 2.6: Approximation to the solutions of (2.27), using the finite-difference scheme (1.43) with the model and computational parameters in Table 2.1 for Example 2.4, and $b = 1.5$. Various times were considered, namely, (a) $t = 50$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The graphing subroutines were based on the function `isosurface` with values of the `isovalue` parameter equal to 0.01.

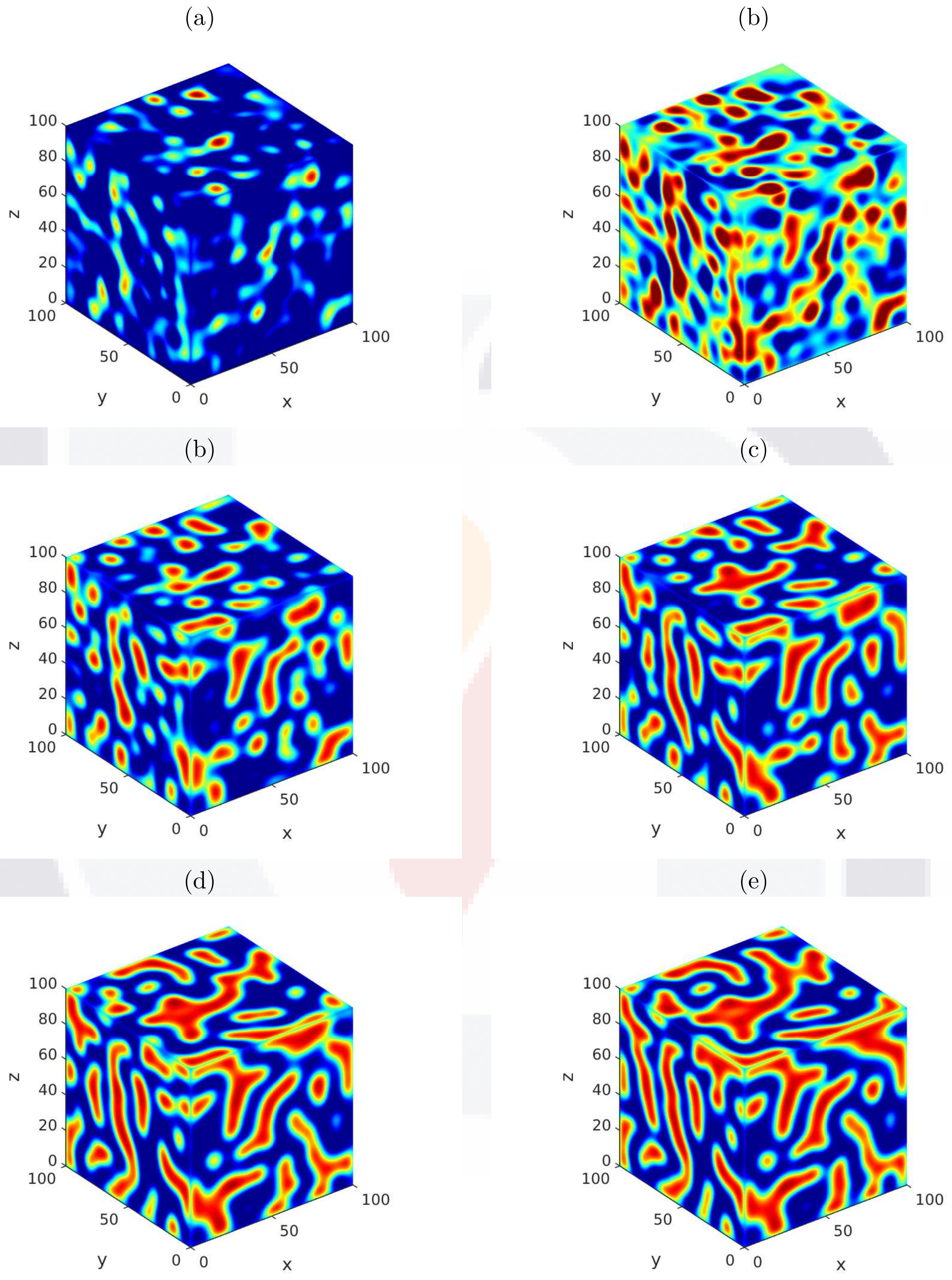


Figure 2.7: Approximation to some x-, y- and z-cross sections of the solutions of (2.27), using the finite-difference scheme (1.43) with the model and computational parameters in Table 2.1 for Example 2.4, and $b = 1.5$. Various times were considered, namely, (a) $t = 50$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$.

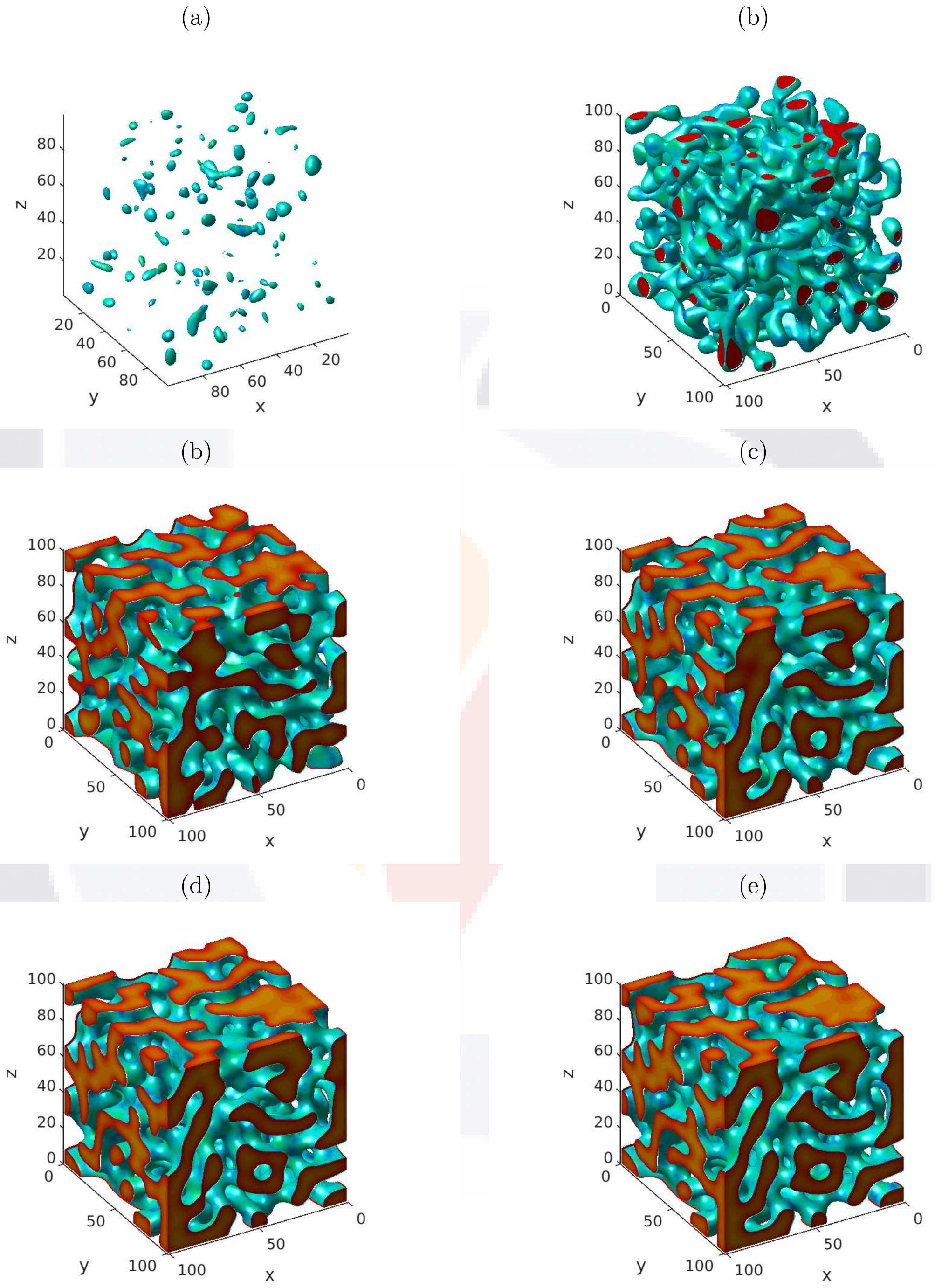


Figure 2.8: Approximation to the solutions of (2.27), using the finite-difference scheme (1.43) with the model and computational parameters in Table 2.1 for Example 2.4, and $b = 2.5$. Various times were considered, namely, (a) $t = 50$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The graphing subroutines were based on the function `isosurface` with values of the `isovalue` parameter equal to 0.01.

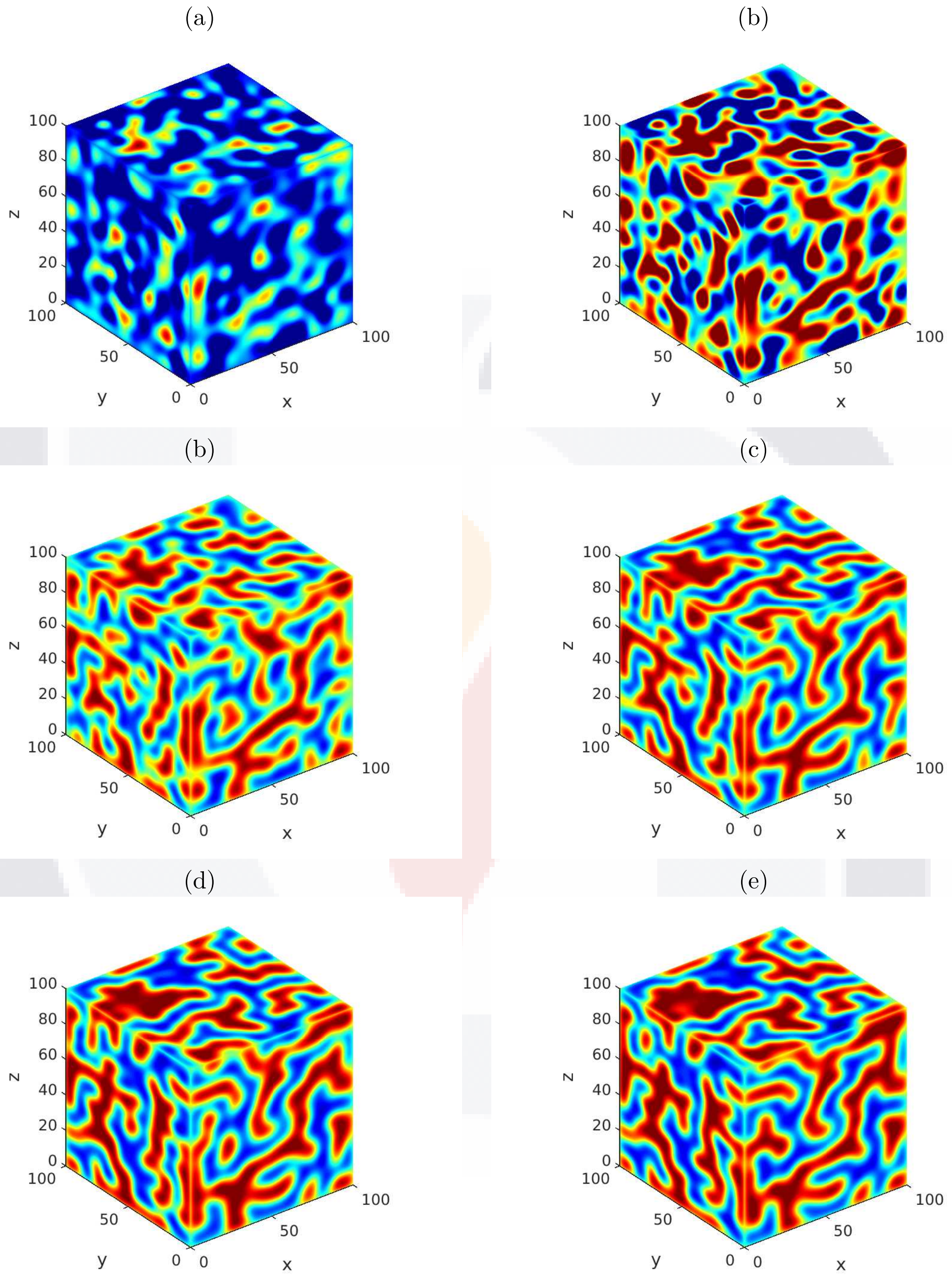


Figure 2.9: Approximation to some x-, y- and z-cross sections of the solutions of (2.27), using the finite-difference scheme (1.43) with the model and computational parameters in Table 2.1 for Example 2.4, and $b = 2.5$. Various times were considered, namely, (a) $t = 50$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$.

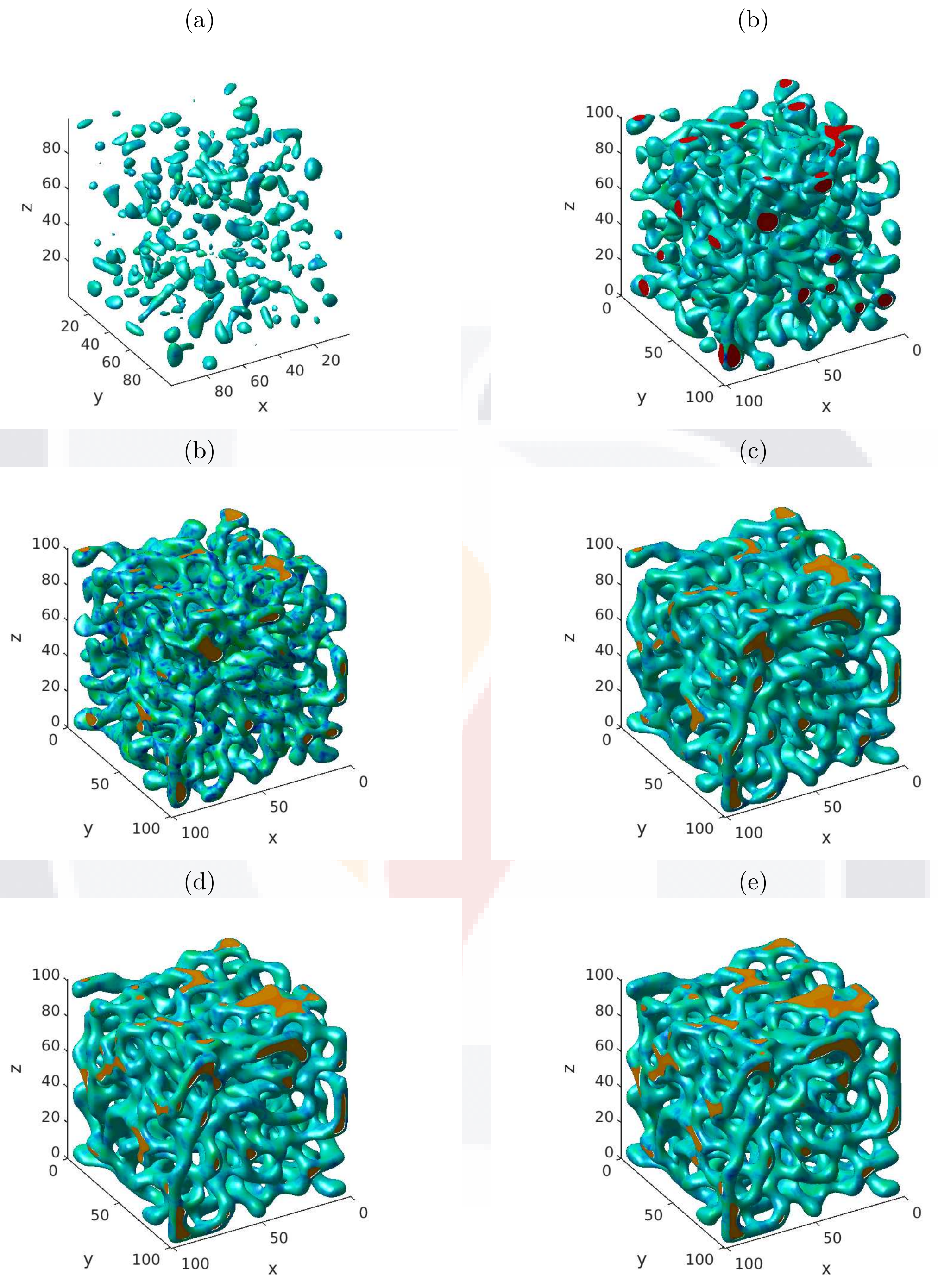
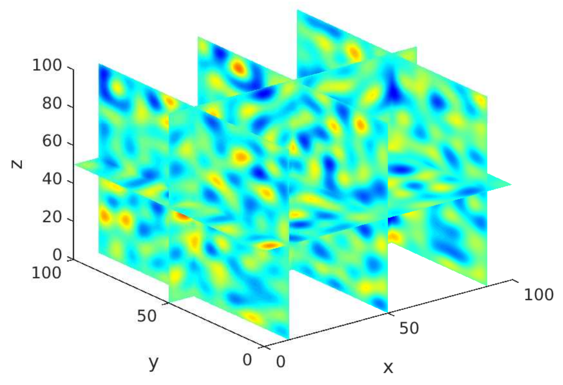
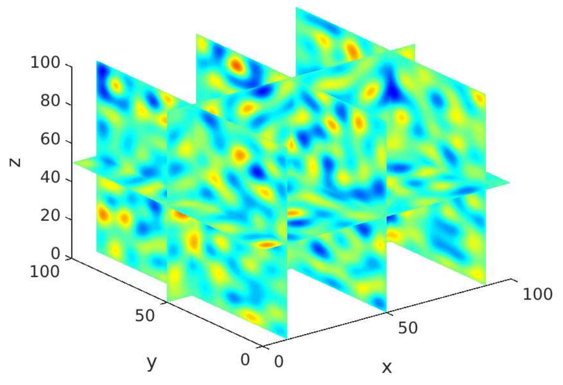


Figure 2.10: Approximation to the solutions of (1.7) with reaction functions (2.33)-(2.34), using the finite-difference scheme (2.14) with the model and computational parameters in Table 2.1, and $b = 0.5$. Various times were considered, namely, (a) $t = 0$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The results were obtained using a parallel Fortran 95 implementation of the Algorithm 1. The graphing subroutines were based on the function `isosurface` with values of the `isovalue` parameter equal to (a) 0.006, (b) 0.01 and (c)–(d) 0.1.

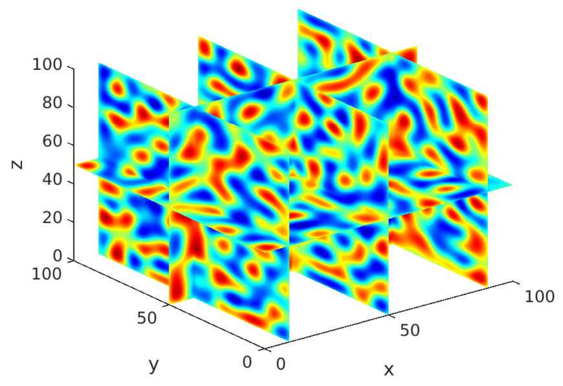
(a)



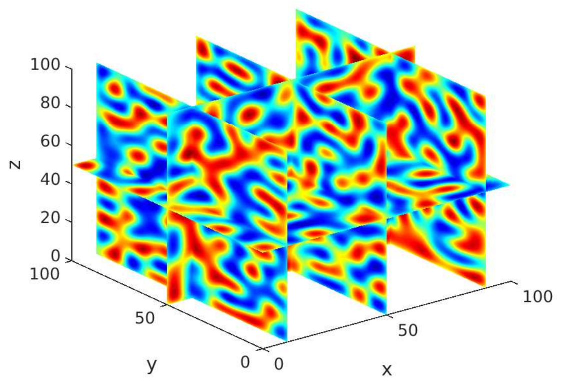
(b)



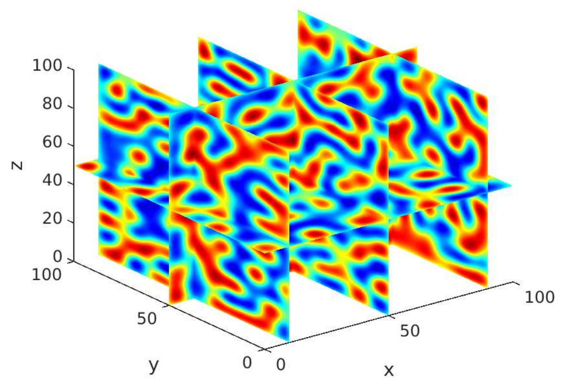
(b)



(c)



(d)



(e)

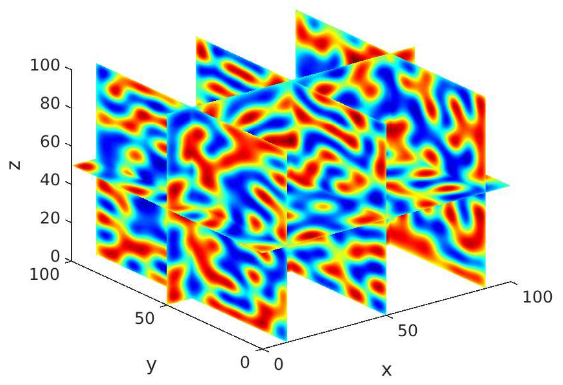


Figure 2.11: Approximation to some x-, y- and z-cross sections of the solutions of (1.7) with reaction functions (2.33)-(2.34), using the finite-difference scheme (2.14) with the model and computational parameters in Table 2.1, and $b = 0.5$. Various times were considered, namely, (a) $t = 0$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The results were obtained using a parallel Fortran 95 implementation of the Algorithm 1.

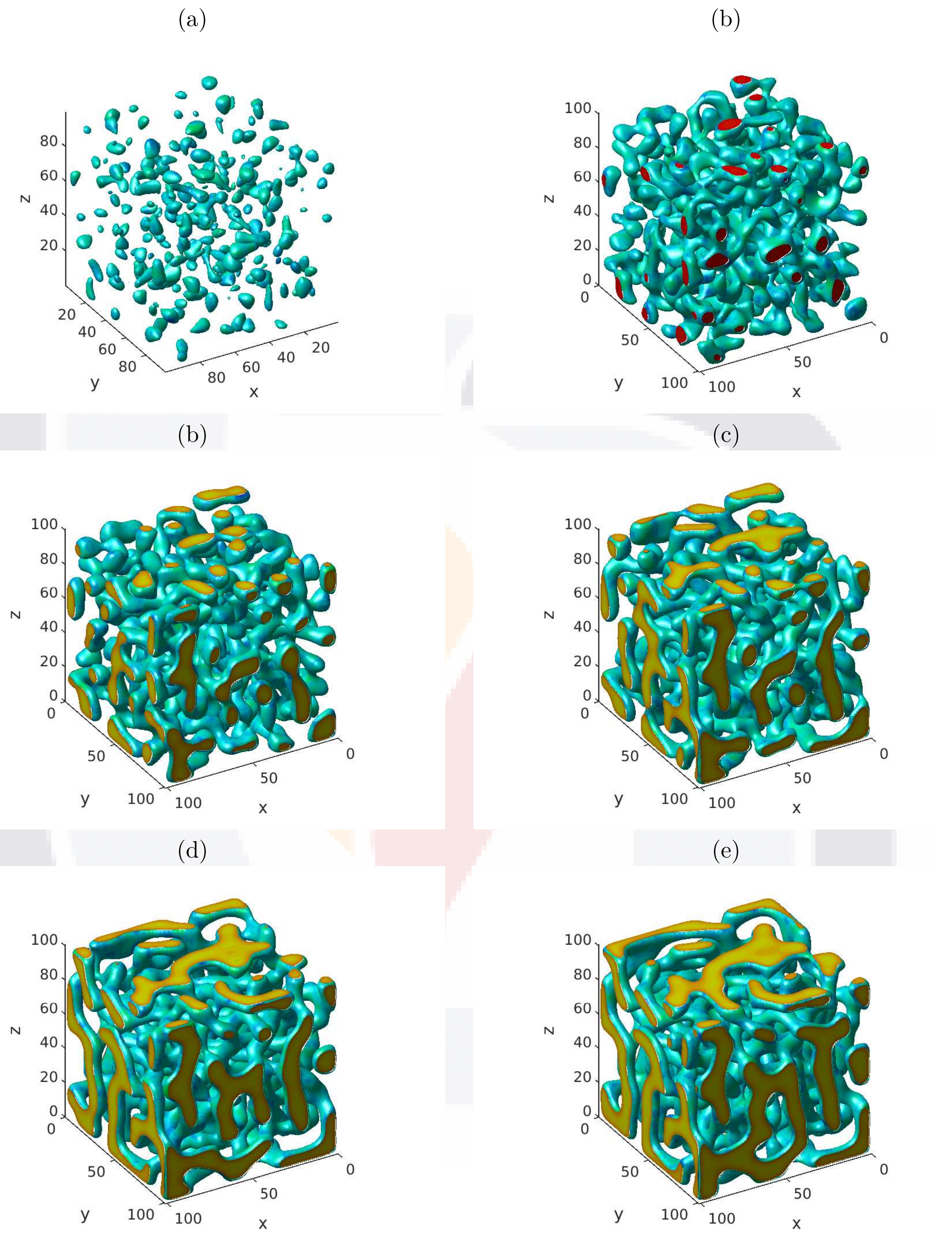
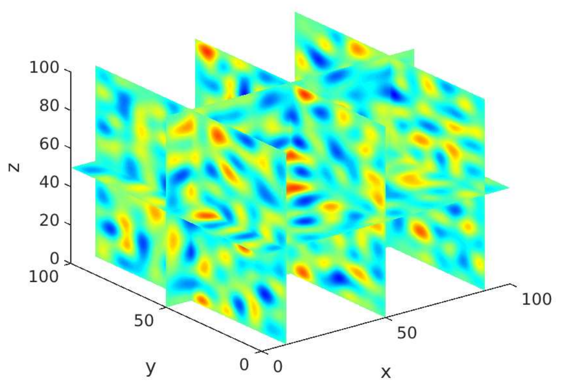
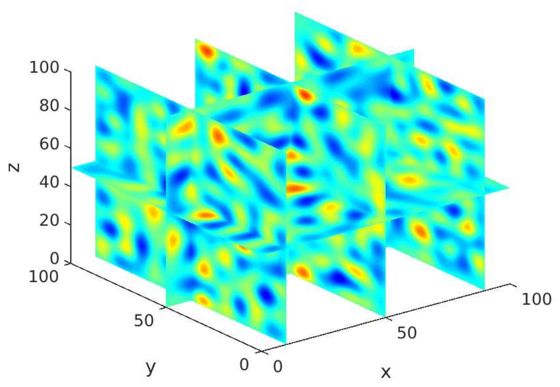


Figure 2.12: Approximation to the solutions of (1.7) with reaction functions (2.33)-(2.34), using the finite-difference scheme (2.14) with the model and computational parameters in Table 2.1, and $b = 1.5$. Various times were considered, namely, (a) $t = 0$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The results were obtained using a parallel Fortran 95 implementation of the Algorithm 1. The graphing subroutines were based on the function `isosurface` with values of the `isovalue` parameter equal to (a) 0.006, (b) 0.01 and (c)–(d) 0.1.

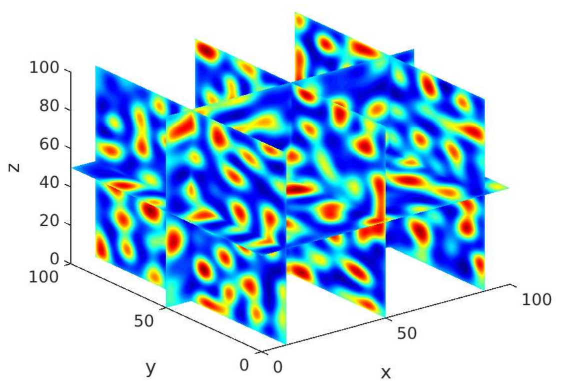
(a)



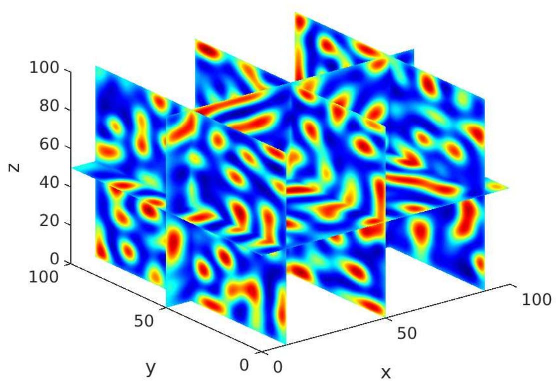
(b)



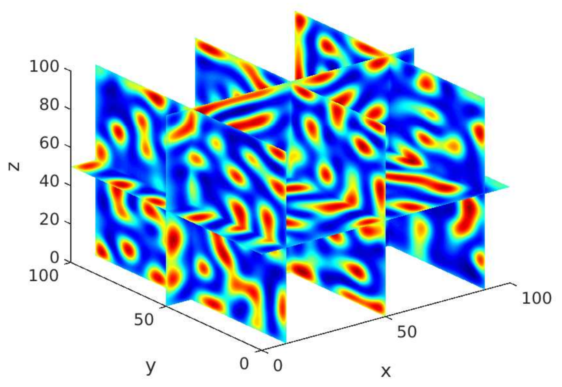
(b)



(c)



(d)



(e)

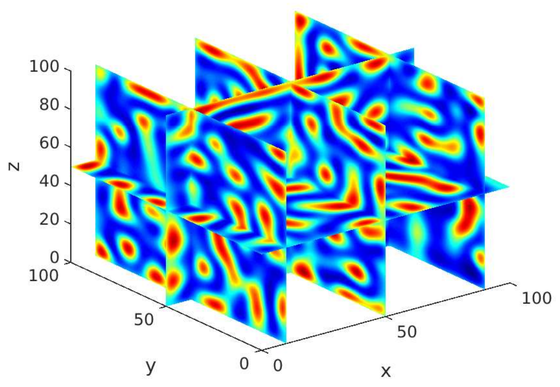
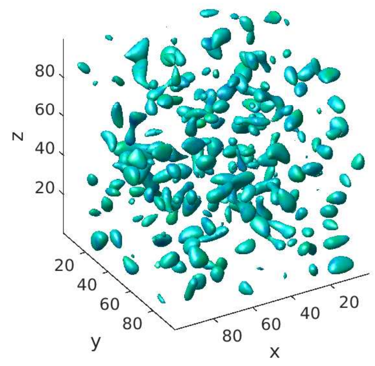
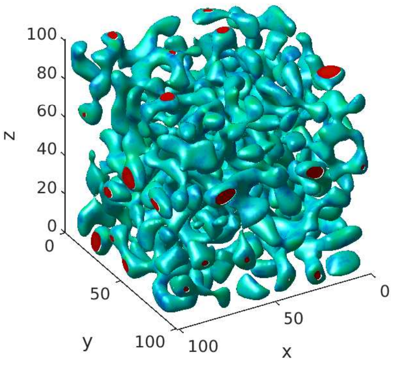


Figure 2.13: Approximation to some x-, y- and z-cross sections of the solutions of (1.7) with reaction functions (2.33)-(2.34), using the finite-difference scheme (2.14) with the model and computational parameters in Table 2.1, and $b = 1.5$. Various times were considered, namely, (a) $t = 0$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The results were obtained using a parallel Fortran 95 implementation of the Algorithm 1.

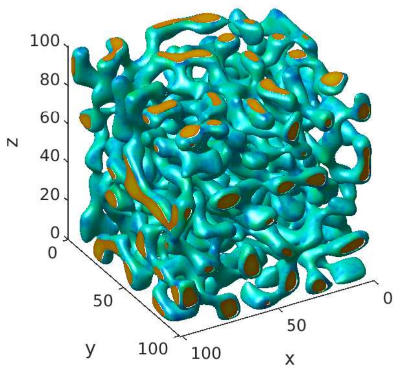
(a)



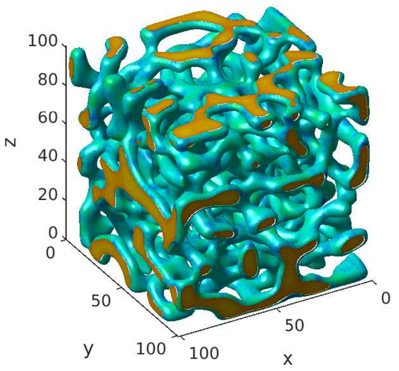
(b)



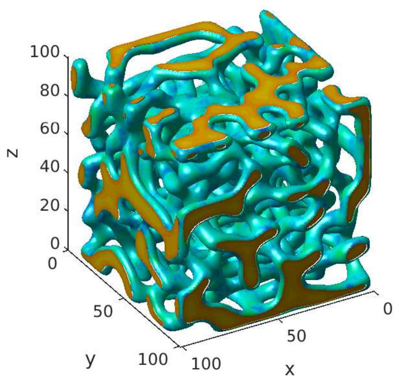
(b)



(c)



(d)



(e)

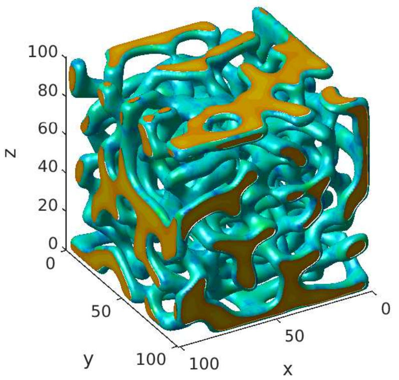


Figure 2.14: Approximation to the solutions of (1.7) with reaction functions (2.33)-(2.34), using the finite-difference scheme (2.14) with the model and computational parameters in Table 2.1, and $b = 2.5$. Various times were considered, namely, (a) $t = 0$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The results were obtained using a parallel Fortran 95 implementation of the Algorithm 1. The graphing subroutines were based on the function `isosurface` with values of the `isovalue` parameter equal to (a) 0.006, (b) 0.01 and (c)–(d) 0.1.

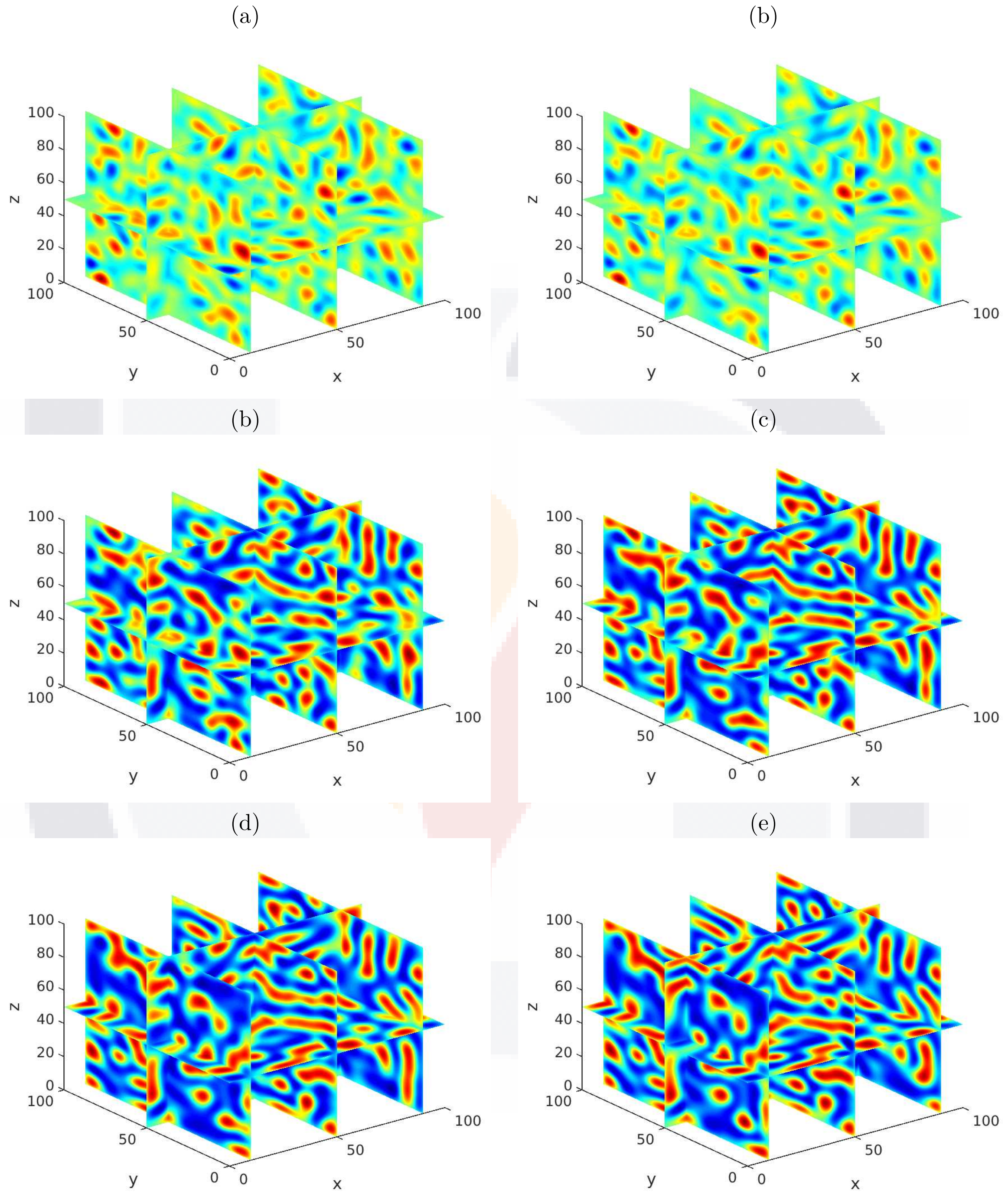


Figure 2.15: Approximation to some x-, y- and z-cross sections of the solutions of (1.7) with reaction functions (2.33)-(2.34), using the finite-difference scheme (2.14) with the model and computational parameters in Table 2.1, and $b = 2.5$. Various times were considered, namely, (a) $t = 0$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The results were obtained using a parallel Fortran 95 implementation of the Algorithm 1.

given by

$$F(u,v) = u - av + buv - u^3, \qquad \forall u, v \in \mathbb{R}, \tag{2.33}$$
$$G(u,v) = u - cv, \qquad \forall u, v \in \mathbb{R}. \tag{2.34}$$

Our examples are motivated by a set of results reported in [44] for the two-dimensional version of the mathematical model considered here. Briefly, the authors of that work found out that, for any value of $\alpha_1, \alpha_2 \in (1,2]$, Turing patterns appear in the system when (2.30) are satisfied. Here, $a_T = \frac{1}{4}(d+c)^2 d^{-1}$. Motivated by this fact, we will fix the parameter values recorded in Table 2.1. We will only let b take on various values in $\mathbb{R}^+$.

Example 2.6. Consider the problem (1.7) with the reaction functions (2.33)-(2.34), together with the set of model and computational parameters in Table 2.2. Additionally, we let $b = 0.5$. Figure 2.10 shows snapshots of the approximate solution obtained using the scheme (2.14) at various times, namely, (a) $t = 50$, (b) $t = 100$, (c) $t = 250$, (d) $t = 500$, (e) $t = 1000$ and (f) $t = 2000$. The results depict the presence of complex patterns in this system. The results were obtained using a parallel implementation of our code in Fortran 95, and the visualizations were obtained using standard routines in Matlab, In particular, we employed the standard command `isosurface` with different values of the `isovalue` parameter to that end. More precisely, we used values of the parameter `isovalue` equal to (a) 0.006, (b) 0.01 and (c)–(d) 0.1. Figure 2.11 shows some x-, y- and z-cross sections of the solutions at each of those times. The solutions exhibit some complex patterns in this case. Figures 2.12 and 2.13 are similar to Figures 2.10 and 2.11 using now $b = 1.5$. Finally, Figures 2.14 and 2.15 show the same results for $b = 2.5$. In all cases, we note the presence of complex patters in the three-dimensional medium. □

Remark 2.7. Before closing this stage, we would like to mention that the simulations were carried out using the Fortran 95 program in Appendix A. The advantages of the code are various:

1. On the one hand, the computer program is rather simple, and it is a straightforward implementation of the scheme (2.21)-(2.22). In that sense, the computational implementation of the numerical method is an easy task for any scientist with modest knowledge on scientific programming.

2. The computer program makes use of the Fortran 95 routine `matmul`, which is an efficient implementation of the multiplication of matrices that produces fast results.

3. The algorithm can be easily parallelized. Indeed, in our computer implementation we used the package OpenMP to that end. Needless to mention that the performance of the program is computationally faster using that package.

3. Computational method

Nowadays, because the available technology, a wide variety of problems can be solved. The solutions for these problems are intended to help humans to perform their everyday activities. These activities could be simple as turning on a TV set or complex such as forecasting stock exchange. We could see the impact of technology by analyzing how activities were perform in the past compared to today. Computers are an integral part of such solutions because they are capable of performing very complex calculations at an extremely short period of time. At doing so, allows humans to perform incredible feats such as traveling to outer space, observe microorganisms, or performing tasks in highly contaminated environments, among others. In the field of health care, computers and Information and Communication Technologies are being used in order to develop new drugs, new treatments, new devices for performing remote surgeries or brain surgeries. These technologies allow humans to execute more precise tasks, and, by doing so, minimizing risks. In the exact sciences field, technology is being used for solving very complex problems. Technology such as computers, robots, Internet of Things, among others. While using computers, software is an integral part of the implemented solutions. Software can be used to perform calculations, and to solve statistical and mathematical problems, instead of doing it manually. Thus, researchers could save a lot of time, reduce the costs by using technology as a tool in their projects. Technological advances are possible by the computer. It is necessary to have certain tools such as programs or software, which serve as intermediaries between computers and people who use it. Each program is designed or developed for a specific purpose, in order to facilitate the daily activity for each person. This section lists and describes existing software categories.

3.1 Types of software

Nowadays, we can find different types of software that is being used in all kind of people's activities for performing their daily activities. Such software may run in a computer or be imbedded into a device such as smart TV, smart phone, a washing machine, among others. Pressman [64] classifies that into 7 different groups, depending on the activity that is performed or their based on its purpose. However, these seven broad categories of computer software present continuing challenges for software engineers. These categories are as follow:

1. **System software:** a collection of programs written to service other pro-grams. Some software systems (e.g., compilers, editors, and file management utilities) process complex (but specific) information structures. Other systems applications (e.g., operating system components, drivers, networking software, telecommunications processors) process largely indeterminate data. In either case, the systems software area is characterized by heavy interaction with computer hardware; heavy usage by multiple users; concurrent operation that requires scheduling, resource

sharing, and sophisticated process management; complex data structures; and multiple external interfaces.

2. **Application software:** stand-alone programs that solve a specific business need. Applications in this area process business or technical data in a way that facilitates business operations or management/technical decision making. In addition to conventional data processing applications, application software is used to control business functions in real time (e.g., point-of-sale transaction processing, real-time manufacturing process control).

3. **Engineering/scientific software:** Has been characterized by "number crunching" algorithms. Applications range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing. However, modern applications within the engineering/scientific area are moving away from conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take on real-time and even system software characteristics.

4. **Embedded software:** resides within a product or system and is used to implement and control features and functions for the end user and for the system itself. Embedded software can perform limited and esoteric functions (e.g., key pad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems).

5. **Product-line software:** designed to provide a specific capability for use by many different customers. Product-line software can focus on a limited and esoteric marketplace (e.g., inventory control products) or address mass consumer markets (e.g., word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, and personal and business financial applications).

6. **Web applications:** called "WebApps," this network-centric software cate-gory spans a wide array of applications. In their simplest form, WebApps can be little more than a set of linked hypertext files that present information using text and limited graphics. However, as Web 2.0 emerges, WebApps are evolving into sophisticated computing environments that not only provide stand-alone features, computing functions, and content to the end user, but also are integrated with corporate databases and business applications.

7. **Artificial intelligence software:** makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straight for-ward analysis. Applications within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing.

Therefore, it is important to understand each category because they outline the main attributes for a specific solution. For the present thesis, a scientific software solution for was presented in section 3.2.

3.2 Software solutions

In previous sections, we presented a mathematical solution for a specific problem of. In this section, we present an algorithm that is implemented using software for such solution. Our approach is codded in Fortran programming language. Fortran was chosen because its simplicity and execution power for implementing mathematical problems [65]. In addition, Fortran 95 has the capability of exploiting modern processors because it can execute source code in sequential or parallel fashions.

Explanation

Once the algorithm, an initial solution was codded, which was tested by using hand-made tests. Such tests showed that execution, grammar and semantic issues were not existing within the code. Some hand-made runs were performed in order to prove that the algorithm was properly functioning.

Once the algorithm was obtained, it was moved to the coding stage. Coded was performed using Fortran f95 because this the best language to implement the found solution. In order to test the code, some sample runs were performed so that no syntax and/or grammar errors were existing within the coded solution. As expected, the coded solution delivers the required results. Thus, we can conclude that the implementation is an adequate solution or implementing the algorithm. The resulting algorithm is shown in Figure 1.

By testing the coded solution by using the above algorithm, we noticed some important aspects that we should take into consideration. There are two parameters that directly affect the response time of the algorithm, these two parameters are: sample size and the number of iterations. The best applications are when the algorithm uses large sample size and number of iterations are. However, the response time used by the implemented solution in order to solve those applications was very poor. The initial version was developed using a sequential approach. For now on, the coded solution will be referred as TurPatts.

For example, TurPatts has a response tome approximately 48 hours to solve a problem under the following conditions shown in Table 3.1

Table 3.1: Parameters that influence response time

Parameter	**Value (units)**
Number of iterations	2500
Sample size	200

We can notice that the above parameters for obtaining the best results of the real ideal case, TurPatts requires about two days to obtain the corresponding solution. Thus, it is required to develop and enhanced solution for TurPatts that solves the problem using a minimal response time. In order to prove what we argue in this paragraph, let us to calculate the corresponding complexity analysis of the initial approach. We might notice that the degree of complexity of the algorithm is high, as can be seen in the code snippet shown in Figure 3.2. Because it has four nested cycles, and since it is the main part of code, we can estimate that the code has an order of complexity equivalent to $O(n^4)$.

```
do n=3,NumIt
  do i=1,M
    do j=1,M
      do k=1:M
        .
        .
        .
      end do
    end do
  end do
end do
```

Figure 3.1: Sample Code

In conclusion, we need to develop enhanced solutions that deliver optimal outcomes. The following sections described two additional approaches that were developed. These two deliver outcomes that we believe are close to optimal.

Nowadays, there are several programming schemes or paradigms. Thus, depending on its purpose, each software development must follow a special paradigm that suits better the expected outcomes. For the present study, two different approaches were used trying to develop the best solution. Those approaches are: sequential and parallel programming.

The classic view of sequential programming refers to the execution control. This type of programming is one of the easiest to use, and the basic rule and to follow is that the instructions will be executed in an orderly and successive fashion: one by one. This programming paradigm when it is executed, an instruction is performed one at a time, and the following instruction cannot begin executed until the current one is fully finished; for example, loops within the code. When a sequential program reaches a loop-like instruction, the execution will remain in this statement indefinitely until all units in the loop is completed.

One of the features of this type of programming is that it will only use a single processor core, no matter the number of processors/cores the computer has. It is important to mention that current computers possess processors that have more than one core. Thus, we can argue that this type of programming does not make optimal of modern processors.

Figure 3.2 shows a graphical representation of how sequential programming works.

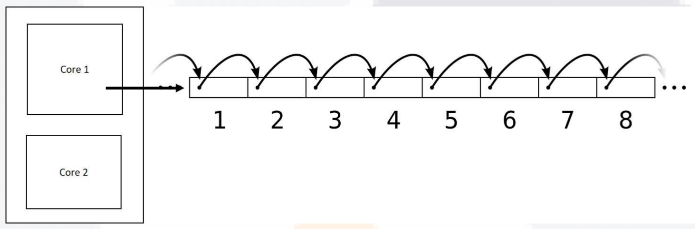


Figure 3.2: Graphical representation of sequential programming

Most programmers do not make use of parallel programming. This could be because the use of a single processor core is seen as more than enough to satisfy the requirements by the software programs they code. However, there are specific programing solutions in which the use of a single processor core causes the program to run slowly and this is often because the capacity of the processor core has been exceeded [66]

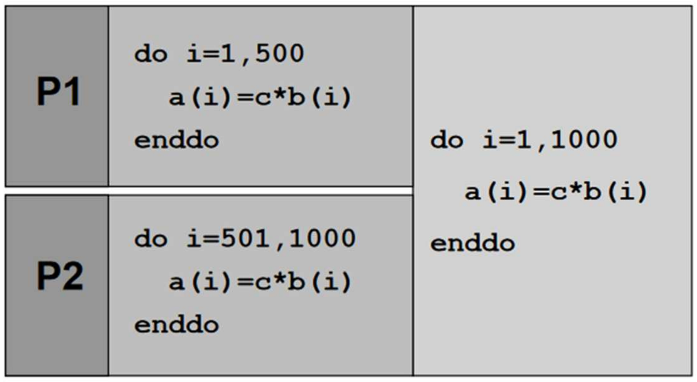
Writing a parallel software program must always start by identifying the parallelism inherent in the algorithm at hand. Different variants of parallelism induce different methods of parallelization.

Many problems in scientific computing involve processing of large quantities of data stored on a computer. If this manipulation can be performed in parallel, (i.e., by multiple processors working on different parts of the data), we are referring to data parallelism [66]. Since we identified what we need to perform, in order to improve the response time of our coded solution, we believe that developing codes based on parallelism would enhance significantly the performance of TurPatts. Section 3.3 describes briefly each approach used to implement TurPatts.

3.3 Final solution

In all, 3 different solutions were developed. Results from the first solution we found that it was necessary to improve the response time by our implementation. The response time was about 48 hours. Based on what we described in Section 3.2, we found that that it was possible to use a parallelization scheme, and therefore, try to update our software program in order to attempt optimizing the response time required for each solution.

The second approach parallelizes the source code as much as possible. In order to, we identified the parts of the source code that this was possible (see Figure 3.2). For this approach, we applied the parallelization on the cycles, so that all the processor's cores were used; and by doing so this, decreased the response time. Figure 3.3 shows a general outline of how cycle parallelization is achieved, using [66]. Also, we can clearly see how the same cycle can be divided into two parts, this will be limited depending on the number of cores available on the processor.



```
do i=1,500
  a(i)=c*b(i)
enddo
```

```
do i=501,1000
  a(i)=c*b(i)
enddo
```

```
do i=1,1000
  a(i)=c*b(i)
enddo
```

Figure 3.3: Division of cycles by parallelism.

Next, we compare the coded alternatives solutions. First, the original solution was programmed under a sequential programming paradigm. This alternative was discarded for the analysis because the time required to obtain each instance was extremely high. For example, for a data set shown in Table 3.2 the solution demanded about 48 hours.

For the comparison of the other two solutions obtained, 2 tests were performed to measure the performance of the algorithms. First, some executions were performed in order to confirm that the results obtained were the same or, at least, very similar (with a very small margin of error, equivalent to or less than $1{*}10^{-20}$). Once it was found that both solutions are correctly developed then the second test was carried out. In this proof it was necessary to perform 30 iterations with a set of parameters in order to observe the behavior of the algorithms. Table 3.2 shows the mean of the 30 iterations that were running for each instance of algorithm. We must remember that exists two important parameters that directly affect the execution time of the algorithm. For each iteration, the parameter L was used as a constant with a value of 200. The parameter T was changing in each iteration. It should be noted on Table 3.2 that the parameter T is in hundreds, so actually in row one the number of iterations were 5000 instead of 50.

Something important that we expected was that algorithms have a linear trend, that is, by increasing the number of iterations, the execution time is directly proportional.

Table 3.2: Times

Parameter (hundreds)	Parallel cycles algorithm (s)	Parallel matrix algorithm (s)
T = 50	28.66244	5.3940862
T = 100	57.26734	10.549308
T = 250	142.6873	26.067167
T = 500	285.5651	51.756834
T = 1000	572.2379	103.85001

If Table 3.2 is observed, the matrix algorithm was almost 6 times faster than other one. So, this algorithm drastically reduces the time required to get a response. As example, in the last row of Table 3.2, results of 100000 iterations were calculated. Instead of waiting approximately 10 minutes using cycles algorithm, now matrix algorithm can reach the same result in 2 minutes.

Finally, everything that has been mentioned so far can be seen more easily in Figure 3.3.

With the results obtained now, we can say that, of the 3 algorithms obtained, the best is the last one, since in the previous section we check the response times, and this turns out to be the fastest and therefore will be our final algorithm. The next step is that we will implement our algorithm with

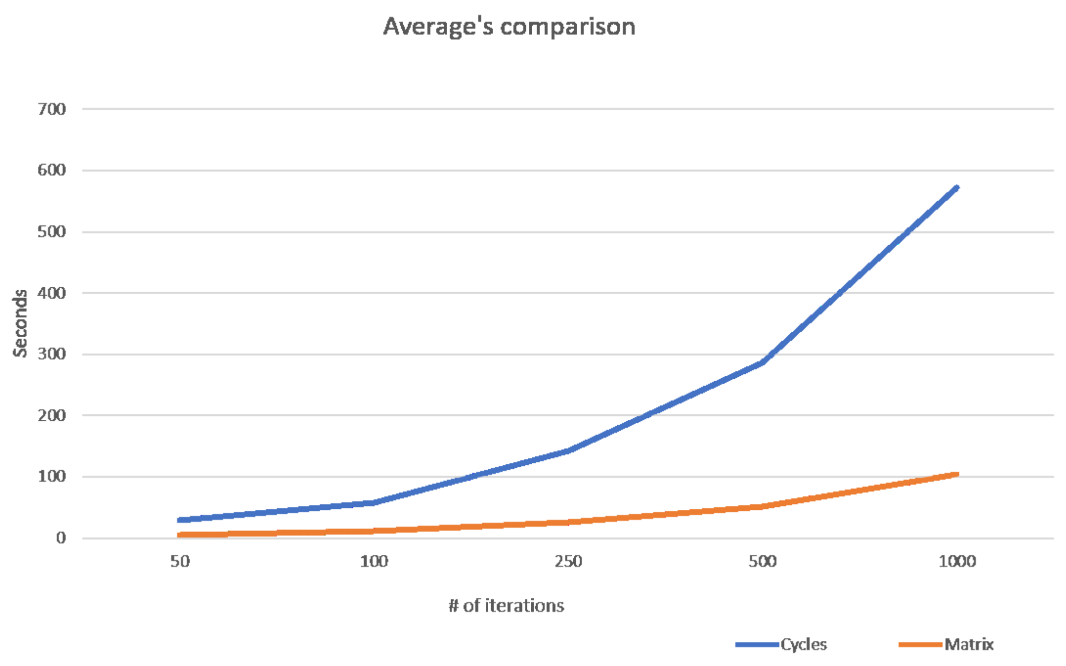


Figure 3.4: Comparation of execution time. The number of iterations is in hundreds.

a graphical environment, this with the aim of making the program more complete and easier for the user to use it.

4. Application development

Nowadays, software is part important of almost all business operation, so new software must be developed quickly to take advantage of new opportunities and to respond to competitive pressure. This new chapter focuses on application development. This software allows the end-user to use TurPatts algorithm using the main software principles: easy to use, easy to understand, and easy to learn. The agile methodology known as Rapid Application Development (RAD) was used for the final implementation. This methodology is described in detail in following sections. Previous chapters describe that TurPatts requires to input the necessary parameters to perform and generate a special scenario. Thus, it is necessary to change the parameters directly in the source code, recompile and execute for each time it is used, which deems to be very complex for people that is not savvy in developing software. Hence, it was decided to develop an application with GUIs so that the end-user can interact easily with the algorithm without having to change the parameters from the source code. In fact, the GUIs will play a very important role so that any end-user will be able to use the algorithm. GUI is a concept with various definitions accepted by several experts in the area. In general, the end-user interface can be understood as the component through which the end-user interacts and perceives the tasks executed by the software application; that is, they serve as an intermediary between the end-user and the hardware. Consequently, the GUI has the main objective of making computer applications understandable, simple and usable for end-users. The fact that interfaces are needed implies that there is a ”something different” between system and user. ”Something” that needs to be solved for the end-user's interaction with the system to be simple, fast and smooth. Therefore, it is necessary to know what an interface is, because its quality, usefulness, usability and acceptance depends on the success of a system [67]. As mentioned, previously, there is a wide repertoire on definitions of GUIs. One of the most accepted is the definition by Landauer [68]: ”a graphical interface is the set of controls by which end-users can make a system work”.

4.1 Methodology Selection

Existing literature [69] describe different software methodologies. For the proposed solution, two compiting were analyzed. First, the cascade approach that demands that each stage should be fully completed before initialing a new one. This methodology demands an extensive developing time as well a very little interaction with end-user during the software coding stage. Because of these issues are not well prepared for the present project it was rejected. The second considered methology is an Agile approach. This approach demands an extensive and intense interaction with end-users and usually requires less time for the overall project development. Hence, this is the best methodology for the type of project developed. Agile methods universally rely on an incremental approach to software specification, development, and delivery. They are best suited to application development where the system requirements usually change rapidly during the development process. In the book of Sommerville wrote down a philosophical idea. In addition, he argues that the philosophy behind agile methods is reflected in the agile manifesto that was agreed on by many of the leading developers of these methods. Somerville manifesto states:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

1. *Individuals and interactions over processes and tools*
2. *Working software over comprehensive documentation*
3. *Customer collaboration over contract negotiation*
4. *Responding to change over following a plan*

That is, while there is value in the items on the right, we value the items on theleft more

Agile methodologies are widely used because of the high success rate in developing certain types of software systems. However, Sommerville [69] argues that there are two conditions of using successfully agile methodologies, which are:

1. Product development where a software company is developing a small ormedium-sized product
2. Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process andwhere there are not a lot of external rules and regulations that affect the software.

Agile methodologies have advantages and disadvantages. Sommerville list these two aspects as follows.

1. Customer involvement. Customers should be closely involved throughout the development process. Their role is providing and prioritize new system requirements and to evaluate the iterations of the system.
2. Incremental delivery. The software is developed in increments with the customer specifying the requirements to be included in each increment.
3. People not process. The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
4. Embrace change. Expect the system requirements to change and so design the system to accommodate these changes.
5. Maintain simplicity. Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

These advantages are most visible within system components. The iterative design allows the end-user's feedback to be at the forefront of the process. This is very important issue, because the end-user will be responsible for using the application. Consequently, by including the end-user in the planning and design stages, it helps deliver a better application and developing it faster. However, it is important to remember that, as any other software development approach, there are some disadvantages by using agile methodologies. Sommerville [69] lists 5 of the most important, which are:

1. Although the idea of customer involvement in the development process is anattractive one, its success depends on having a customer who is willing and ableto spend time with the development team and who can represent all systemstakeholders.

2. Individual team members may not have suitable personalities for the intenseinvolvement that is typical of agile methods, and therefore not interact well withother team members.

3. Prioritizing changes can be extremely difficult, especially in systems for whichthere are many stakeholders. Typically, each stakeholder gives different priorities to different changes.

4. Maintaining simplicity requires extra work. Under pressure from deliveryschedules, the team members may not have time to carry out desirable systemsimplifications.

Because to the especial conditions, characteristics and requirements of the required application for the present project, it was decided to use the agile methodology known as RAD (Rapid Application Development). RAD processes are designed to produce useful software quickly. The software is not developed as a single unit but as a series of increments, with each increment including new system functionality. Figure 4.1 shows the steps for developing software development using this methodology, adapted from [70].

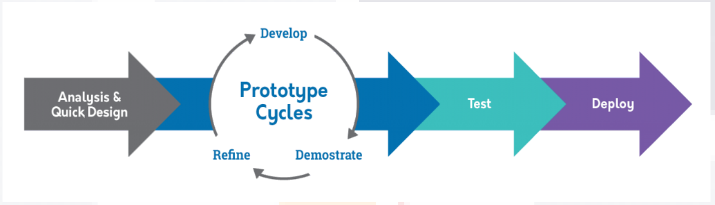


Figure 4.1: Phases of RAD.

4.2 Interface design

The methodology to be used was chosen in the previous section 4.1 and now this section will continue with the planned development of the system. Firstly, we take into consideration some very important points, for the construction of the application. According to Pressman he considers four aspects important: ”system response time, user help tools, error handling and legends or user support messages” [64]. Below are some of the aspects that were taken into account for the development of the application.

1. It must be a minimalist interface, that is, it does not consume many resources because the algorithm requires the increased use of the processor. So, if a very complex and detaining interface would develop, it will use a lot of computing resources. It will directly affect the response time of our algorithm. This would have a significant negative impact, which is a point to avoid.

2. The main algorithm was developed in the Fortran programming language. This allows the algorithm to be cross-platform, that is, it can be executed on Windows and Linux operating systems. For this reason, the application must also allow it to be run in any of these environments. Therefore, you must opt for some programming language that meets these requirements.

Given the above considerations, it was chosen to use several programming languages due to the features presented by the application. Languages and their functions are listed below:

1. Qt was the programming language used for the development of graphical interfaces. Because its structure fits perfectly for the development of prototypes. Also, Qt generates an executable for Windows and Linux.

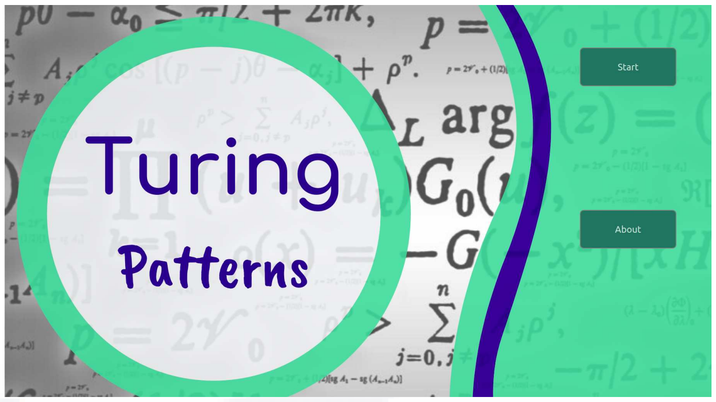
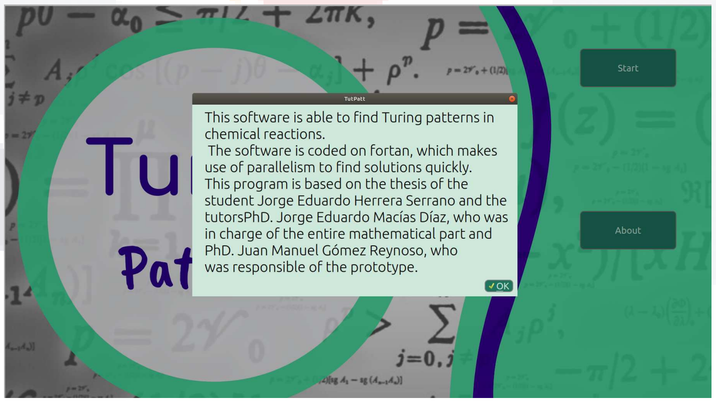


Figure 4.2: Programming model

2. The Python programming language was responsible for making the graphs that were previously drawled under the suite known as Matlab. This language was chosen because of its great performance for making graphs, in addition to the high compatibility it has with Matlab and the set of libraries they share.

3. Fortran continued its role of being the main algorithm. Its main feature was the parallelization support as shown in the previous chapter.

Once the languages to be used are defined, the system will be a completely development environment, that is, now the application will perform the simplest tasks due to the addition of graphical interfaces. In addition, the extra software used as Matlab will completely replace for the code developed in Python. Therefore, the same application will perform all the necessary functions so that the end user can use the developed algorithm. Figure 4.2 shows a block diagram that summarizes all this idea.



This software is able to find Turing patterns in chemical reactions.
The software is coded on fortan, which makes use of parallelism to find solutions quickly.
This program is based on the thesis of the student Jorge Eduardo Herrera Serrano and the tutorsPhD. Jorge Eduardo Macías Díaz, who was in charge of the entire mathematical part and PhD. Juan Manuel Gómez Reynoso, who was responsible of the prototype.

Figure 4.3: Main screen

The methodology chosen for the development of the application allows to carry out the outline presented in Figure 4.2. This, because one of the characteristics of the RAD methodology is that it allows to develop the system incrementally. Which is basically what we wanted to do.

One of the problems of working with different programming languages for tfhe same system is to achieve the connection between them. That is, each of the algorithms must perform its function and that the end user only perceives a unique scheme. As shown in Figure 4.2, each programming language depends on the data sent.

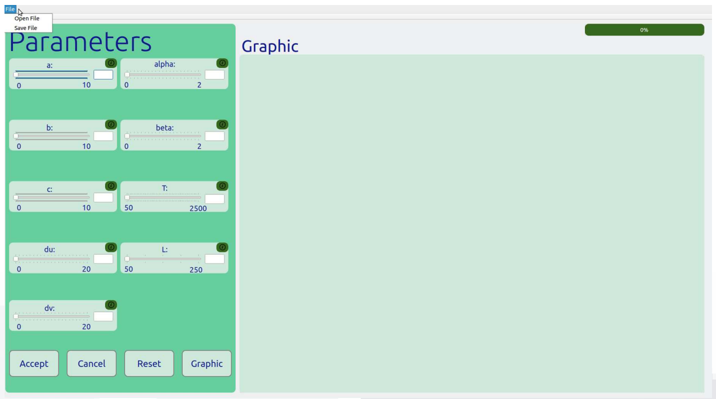


Figure 4.4: About button

The solution proposal to this problem was simple, it relied on the same operating system so that programming languages communicate through processes, that is, once the graphical interface captured the data entered by the end user, it will make a call to our main algorithm under a created process. Once the process of our TurPat is finished, it will now run a new process by calling the program developed in Python to perform the graph. Finally, the result will be sent back to our graphical interface and displayed to the end user.

4.3 Prototype

This section will show the final prototype that was developed using the RAD methodology. The next chapter will measure the performance and approval that the application obtained.

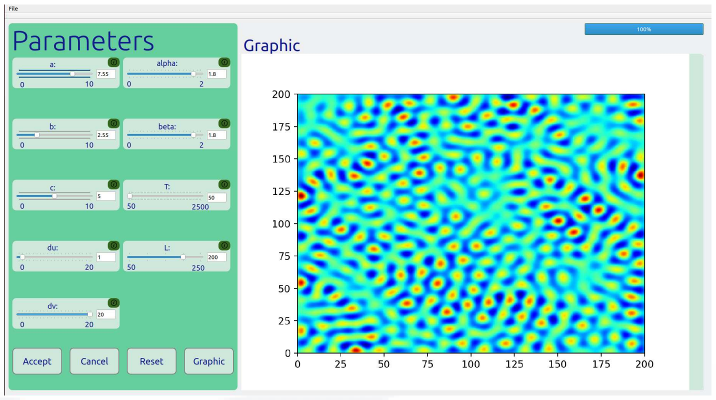
Figure 4.2 shows the main screen of the system. On this main screen the title of the application and to two buttons were shown. Starts button redirects to the work screen. About button shows a very brief overview of the application. This action is shown in Figure 4.2.

Figure 4.2 shows an informative message about what the application can do, in addition to some other details such as the names of the researchers. This message appears on the main screen and the OK button return to the main screen. Once back on the main screen, the start button redirects to the work screen which is shown in Figure 4.3.

Figure 4.3 contains all the functions necessary for our system to work. This particular screen is divided into three parts. The first part is the taskbar, in which you can open or save the values in the parameters area. The second part is the parameter area in which the desired values are entered so that the TurPat algorithm can work. For this it is necessary to use the horizontal bar. In addition, the parameter area contains buttons that allow to perform some actions which are listed below:

1. *Ok:* This button checks that no parameters are empty. Once this task is complete, it sends these parameters to the TurPat algorithm and starts the progress bar at the top right.
2. *Cancel:* Ends with the currently running.
3. *Reset:* Returns all parameters to their initial state along with the progress bar and chart area.
4. *Graph:* Returns the last previously obtained graph to the screen.

Figure 4.3 shows the action that allows the end user to open a file with the previously saved values and these are placed in their respective position in the parameter area. It also allows you to save the values placed in the parameters in any location.



a: 7.55 (0–10)
alpha: 1.8 (0–2)
b: 2.55 (0–10)
beta: 1.8 (0–2)
c: 5 (0–10)
T: 50 (50–2500)
du: 1 (0–20)
L: 200 (50–250)
dv: 20 (0–20)

Figure 4.5: Complete Process

Once the parameters have been captured, and the accept button is clicked, the main process begins. In turn, the progress bar on the top begins, indicating approximately the current progress of the algorithm. When finished, the corresponding graph will be displayed in the graph area and this also completes the whole process. Figure 4.3 shows this process.

5. Analysis

For the present research, ISO / IEC 25010 quality model was analyzed in order to identify the attributes that are relevant to potential end-users. This standard was selected because it is widely accepted as relevant for software quality measurement. There are other factors in other models, but they are evaluable from the developer's point of view, which is beyond the present research.

5.1 Instrument Development

Based on the review of the literature and base theories on software quality standards analyzed (ISO / IEC 25010), the quality elements of a software product that can be perceived by end-users were identified. Thus, an initial questionnaire was were created as well as the scale of possible answers, and it was applied in a pilot study to 14 students of the subject of Software Engineering in the 9th semester of Computer Systems Engineering. For this purpose, the students assessed the created instrument, the quality of a software that was delivered to them. Potential issues in the instrument were identified and corrected, resulting in the final instrument of quality evaluation of software products that was composed by a total of 25 questions. All questions have a scale of 1 (Excellent Quality) to 7 (Extremely Poor Quality).

This procedure delivered the final evaluation document that was used in the present research for the evaluation of TurPath software.

5.2 Instrument Validity

Data Collection In order to perform the analyses, the questionnaire was applied to 56 students in the Autonomous University of Aguascalientes registered in the Computer Systems bachelor degree program. Participants could drop out of the process; however, none of them opted out. They used the system following the instructions given by the lead researcher for about 30 minutes. After that, they answered the provided questionnaire using an online application so that damaged data could be avoided.

The following section presents a detailed data analysis.

Data Analysis For the data analysis, the present research uses a 90% acceptance value, which can be calculated using the Mean as the principal component.

Table 5.1 shows the results of dispersion analysis. It can be observed that in the *Understandability* factor the end-user acceptance is about 91.57%. Thus, this factor meets the requirement, and, therefore, the system is *Understandability* enough so that end-users can understand how to use the system as well as to understand all the interaction required to operate it. The calculated *Learnability* factor is

88.39%. This value is very close to the 90% required. However, still requires some upgrades. Further analyses and proposed upgrades are described in Section 5.2.

The third factor corresponding to *Usability compliance* is about 88.52%, which like the previous factor is very close to the minimum approval required but is not enough, so we will also go on to perform a more detailed analysis to find the possible cause and therefore the solution that can be proposed to improve. Further analyses and proposed upgrades are described in Section 5.2. Then, the Attractiveness factor is 90.10%, which is enough to the required value, so this factor meets the requirements so it will not be necessary to review it as previous factors. Finally, the Operability factor is the lowest rated with 87.60%. Although the factor is not far from the required value, the detailed analyses and proposed upgrades are described in Section 5.2.

Table 5.1: Factors

		Understand-ability	**Learn-ability**	**Usability compliance**	**Attrac-tiveness**	**Opera-bility**
N	56	56	56	56	56	56
	0	0	0	0	0	0
Mean		1.5060	1.6964	1.6888	1.5938	1.7440
Mean standard error		.07244	.08738	.11203	.09353	.10778
Median		1.3333	1.5000	1.4286	1.5000	1.6667
Mode		1.17	1.00	1.00	1.00	1.00
Standard deviation		.54213	.65393	.83837	.69994	.80652
Variance		.294	.428	.703	.490	.650
Skewness		1.627	1.106	1.894	2.310	1.579
Skewness standard error		.319	.319	.319	.319	.319
Kurtosis		2.428	1.032	3.567	8.924	3.657
Kurtosis standard error		.628	.628	.628	.628	.628
Range		2.17	2.80	3.86	4.00	4.00
Minimum		1.00	1.00	1.00	1.00	1.00
Maximum		3.17	3.80	4.86	5.00	5.00
Percentage		91.57	88.39	88.52	90.10	87.60

Some factors did not meet the requirement value, so in the following sections a more detailed analysis of these factors will be shown

Learnability Analysis Table 5.2 shows the estimated statistics for the *Learnability* factor, which is formed by a total of 5 variables. Easiness to remember has an 86.61% acceptance value, which means that, although it is close to the required value, it still needs to be enhanced. This variable refers to how *easy to remember* are the components on the interface as well as to their corresponding actions. Thus, we believe that changing the layout might solve this issue. The *Easiness to understand* variable has an 88.39%. Therefore, we believe that by adding tags that are more descriptive would solve this issue. *Easiness to read* variable has an 91.96%, which is above the acceptance value. We believe that the TurPath does not need to change nothing regarding this aspect.

The fourth question tells about the order in which the information is presented, the *Organized content*, it is 86.01% which the lowest percentage of the entire Learnability fact is, so more emphasis will be placed and in the next version of the system. Finally, the *Easiness to recognize* has an 88.99% value. We believe that changing the layout and the color contrast would enhance the end-user perception.

All these changes should be performed for a second version of TurPatt.

Table 5.2: Learnability Factor

		Easiness to remember	**Easiness to understand**	**Easiness to read**	**Organized content**	**Easiness to recognize**
N	Valid	56	56	56	56	56
	Lost	0	0	0	0	0
Mean		1.80	1.70	1.48	1.84	1.66
Mean standard error		.145	.130	.095	.137	.109
Median		2.00	1.00	1.00	1.50	1.00
Mode		1	1	1	1	1
Standard deviation		1.086	.971	.713	1.023	.815
Variance		1.179	.943	.509	1.046	.665
Skewness		2.086	1.768	1.780	.968	1.127
Skewness standard error		.319	.319	.319	.319	.319
Kurtosis		4.988	3.404	3.793	-.256	.730
Kurtosis standard error		.628	.628	.628	.628	.628
Range		5	4	3	3	3
Minimum		1	1	1	1	1
Maximum		6	5	4	4	4
Percentage		86.61	88.39	91.96	86.01	88.99

Usability compliance Analysis Table 5.3 shows the *Usability compliance* factor statics. This factor is composed by a total of 7 variables. Each one is analyzed individually as follows:

Similarity variable refers to the tasks that are performed in the system. Its evaluation is 89.29%, so for the next release we must find a way to make activities simpler. We believe that adding a value entry the end-user interfaces would make them feel more comfortable and with that solve the issue. *Completeness* variable refers to the tasks that can that application executes. Its value is about 90.18% approval. We believe that in this variable we do not have do any modifications. *Error Handling* variable has about 84.52% acceptance value. Although the scroll bar was used in the software to avoid data-entering errors by end-users, maybe them would prefer a different data-entry option. Thus, in the next version a data-entering option could be given to the end-user. They can select whether capture data using sliders or by using a keyboard. In addition, for the keyboard data-entering option we should add validation in order to avoid potential errors.

The *Feedback* variable has an 88.10% quality value. We believe that end-user would require more messages about the task that the software is performing; thus, we will add more comprehensive dialog messages for each variable in the Turing equations. For instance, a short description for each one would help to address this issue. *Help* variable has an 83.93% approval ratio. We believe that adding context help, which will indicate how the software works, a search help feature, and additional information about each section would solve the issue. Finally, the last two variables (*Navigation* and *Interaction*) are greater than 90%, (*93.45%* and *90.18%* respectively); therefore, end-user believes that the software meet these requirements.

Table 5.3: Usability compliance Factor

		Similarity	**Completeness**	**Error Handling**	**Feedback**	**Help**	**Navigation**	**Interaction**
N	56	56	56	56	56	56	56	56
	0	0	0	0	0	0	0	0
Mean		1.64	1.59	1.93	1.71	1.96	1.39	1.59
Mean standard error		.126	.137	.139	.167	.184	.113	.134
Median		1.00	1.00	2.00	1.00	1.00	1.00	1.00
Mode		1	1	2	1	1	1	1
Standard deviation		.943	1.023	1.042	1.246	1.375	.846	1.005
Variance		.888	1.046	1.086	1.553	1.890	.716	1.010
Skewness		1.732	2.498	1.447	2.206	1.591	2.499	2.365
Skewness standard error		.319	.319	.319	.319	.319	.319	.319
Kurtosis		2.882	7.276	2.179	4.652	1.750	6.498	6.615
Kurtosis standard error		.628	.628	.628	.628	.628	.628	.628
Range		4	5	4	5	5	4	5
Minimum		1	1	1	1	1	1	1
Maximum		5	6	5	6	6	5	6
Percentage		89.29	90.18	84.52	88.10	83.93	93.45	90.18

Operability Analysis For the *Operability* factor the present study uses 3 different variables. Although all the variables for this factor are very close to the 90% that we are looking for, we still need to make some improvements for the next version. *Interaction* variable has an 88.39%. For the next version a choice of two different data-entry ways that the end-user could select from would make them more comfortable as well as on control of the application. Therefore, this solution would solve the issue. *Trust* variable is 87.20%, which continues to refer to the design. We believe that adding information, confirm and alert dialogs would resolve this situation. Finally, *Completeness* variable is about 87.20%. We believe that adding the corresponding dialogs to the actions that the system does would help to solve the problem.

Reliability analysis In Table 5.5 we can see the reliability analysis. It shows the calculated Cronbach Alpha of 0.922, which means that the instrument reliability is Excellent [71]. Therefore, the internal validity is strong. In order to test whether each factor us realizable enough, a reliability analysis for each one was performed. Table 5.6, shows such results. [71] suggests that the Cronbach Alpha for an exploratory analysis should be at least 0.60. However, al factors exceed such cut off value. Therefore, the internal validity is strong for each factor.

5.3 TurPatt Versions

In this last section, the modified software version will be displayed according to the results obtained in the previous sections, in addition to the comments of the users who tested the system. Table 5.7 shows how the starting interface was developed (see Figure 5.7.a) compared to the new developed version (see Figure 5.7.b). The new visual design that was presented can be highlighted in the first instance. This change was the only modification that had this first screen. The functions of the buttons were maintained.

Table 5.4: Operability Factor

		Interaction	**Trust**	**Completeness**
N	56	56	56	56
	0	0	0	0
Mean		1.70	1.77	1.77
Mean standard error		.125	.127	.130
Median		2.00	1.50	1.50
Mode		1^a	1	1
Standard deviation		.933	.953	.972
Variance		.870	.909	.945
Skewness		2.606	1.273	1.352
Skewness standard error		.319	.319	.319
Kurtosis		9.540	1.421	1.533
Kurtosis standard error		.628	.628	.628
Range		5	4	4
Minimum		1	1	1
Maximum		6	5	5
Percentage		88.39	87.20	87.20

The next screen that corresponds to the workspace where a change in the layout is displayed. Figure 5.8.a shows the previous working screen and the new interface developed corresponds to Figure 5.8.b. The new version contains several improvements, which correspond to the proposals in sections 5.2 - 5.2 , in addition to taking into account the comments of the users. The main changes are listed below:

1. The buttons in each parameter display a message with data that provides information about each parameter as shown in Figure 5.8.

Table 5.5: Reliability analysis

Cronbach's Alpha	**Elements**
.922	25

Table 5.6: Reliability statistics by factors

Factor	**Elements**	**Cronbach's Alpha**
Understandability	4	.714
Learnability	5	.742
Usability compliance	7	.889
Attractiveness	4	.887
Operability	3	.802

Table 5.7: Comparison of the starting interfaces.

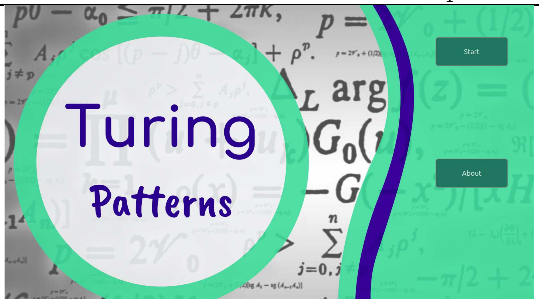
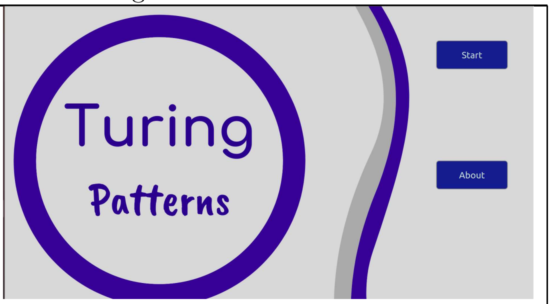
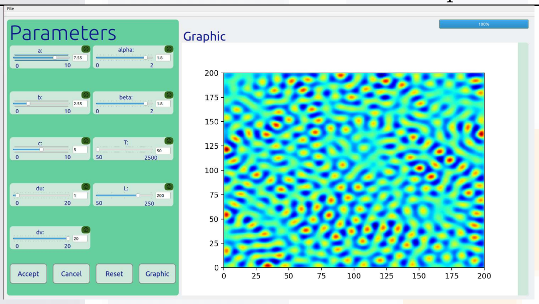
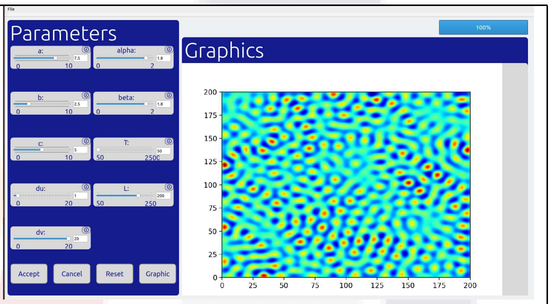
	
a. Previous starting interface	b. New starting interface

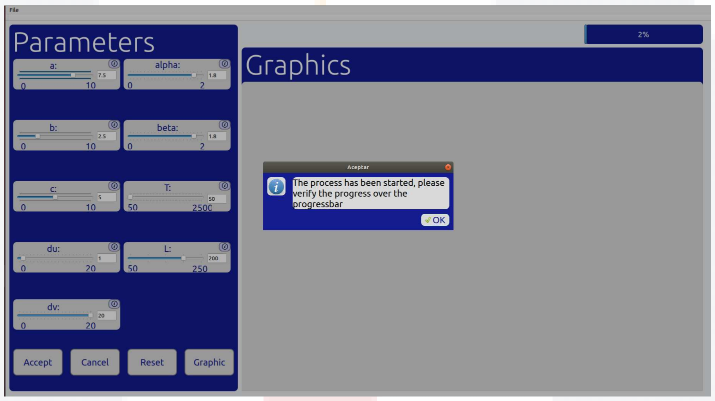
Table 5.8: Comparison of work screen interfaces.

 Parameters a: alpha: b: beta: c: T: du: L: dv: Accept Cancel Reset Graphic Graphic	 Parameters a: alpha: b: beta: c: T: du: L: dv: Accept Cancel Reset Graphic Graphics
a. Previous starting interface	b. New starting interface

2. The interface now allows to use two shapes for manipulating parameters. Added that the user can enter the data directly into the text box. This option is validated, so user will only be able to enter the values requested by the program, also the horizontal bar still could used. With this, the user can use the option which he prefers.

3. Finally, the OK, Cancel, Clean, and Graph buttons now display a message about the task that is performed by just over them. Figure 5.3 shows this new characteristic.

4. All these features were implemented according to what was proposed in the previous sections and taking into account the comments of the users.



a: 7.5 (0–10)
alpha: 1.8 (0–2)
b: 2.5 (0–10)
beta: 1.8 (0–2)
c: 5 (0–10)
T: 50 (50–2500)
du: 1 (0–20)
L: 200 (50–250)
dv: 20 (0–20)

The process has been started, please verify the progress over the progressbar

Figure 5.1: User help message

6. Conclusions

In Chapters 1 and 2, we investigated numerically a system of hyperbolic partial differential equations with fractional diffusion and coupled reaction terms. The mathematical model has various applications, depending on the form of the reaction functions. Motivated by these facts, we proposed an explicit finite-difference models to approximate the solutions of the continuous model. The discrete model is a non-variational scheme for which the properties of consistency, stability, boundedness and quadratic convergence are rigorously proved. To that end, an analytical constraint is imposed on the reaction terms of the mathematical model, and a discrete fractional form of the energy method is employed in order to establish rigorously the stability and the quadratic convergence of the scheme. The discretization is based on the use of fractional centered differences, and the computational implementation is carried out using parallel computing in Fortran 95. To that end, a convenient vector reformulation of the numerical method is proposed. Some illustrative applications of our methodology were presented in this work. Indeed, the finite-difference scheme was employed to solve some nonlinear systems which present Turing patterns in the two-dimensional scenario. Our simulations reflect this fact. Moreover, we used a three-dimensional implementation of our scheme to obtain Turing patterns in some three-dimensional systems. It is well known that the resolution of fractional systems in three-dimensional scenarios is computationally highly demanding. However, our computer implementation was able to carry out this task, and exhibited the presence of complex patterns in the media considered herein.

The 3 developed solutions in order to implement the algorithm previously obtained through the proposed numerical method are an attempt to bring computational solutions to users without them needing to learn a programming language. This is important since a greater impact on the academic and scientific community could be related to the use of Turing Patterns. This method represents a chemical model of activating and inhibiting substances, which is solved using partial fractional equations. The solutions were implemented under the Fortran programming language, each being significantly more efficient than its predecessor. To achieve these advances, it was necessary to thoroughly investigate the existing programming techniques that would help achieve the desired improvement. It should be remembered that the first solution obtained through sequential programming was the least efficient with respect to response time and with the multiprocessing technology present in today's computers, it was thought that it could be improved.

Once it was determined that the algorithm has the necessary characteristics to use parallelism, the second version was developed and thus achieve quite noticeable advances in time. Finally, it was observed that the algorithm could be further improved if instead of nested cycles, matrix algebra was used. With this, the third version was developed where, in fact, the response time to calculate the solution was further reduced. It is important to emphasize this point, since not all programming languages allow the implementation of algebra or parallelization, in addition to the use of nested cycles in the area of software systems. In addition to that programmers do not usually integrate this type of solutions because it is easier to implement solutions with cycles, since matrix algebra requires advanced knowledge in mathematics. It is also necessary to emphasize that not all problems can be parallelized. For this thesis, the work done generates very good results, since the comparisons in times are significantly better, so it was decided to use the third approach as the final solution. It is important to mention that although this solution is excellent it is likely that it can be further improved.

About the software development methodology used, the different approaches to the application were created. It was found that, for the development of the final solution, it is necessary to have a good knowledge in the management of the computer equipment, especially in programming. This situation led to find an alternative for the use of the algorithm, so it was decided to design an application where the end user would use the application using graphical user interfaces. The use of graphical user interfaces in any application potentially allows the user inexperienced in the use of the computer to more easily use the application. Each application is different, so it is very important to consider the requirements of the software to be developed since, based on these, you must choose the methodology that best suits. Some of the key points to consider when choosing the methodology are the size and complexity of the application, and all this can be taken when analyzing user requirements. Due to this, an analysis of some methodologies that allow the easy and fast development of applications was carried out, so that, of the candidates, it was decided to use the Rapid Application Development (RAD) methodology. It is very important to emphasize that every system must be developed under a quality standard, so our system should also be evaluated in this regard. Finally, for the design and development of the system we knew that it had to have the ability to evolve in some future work, that is, currently the system is only able to work with two-dimensional systems, but in the future, they could be n dimensions.

Finally, the evaluation was carried out in order to determine if the software application developed really meets the requirements of the users, as well as being easy to use, easy to understand and easy to learn. The instrument used is based on ISO and has been previously tested and used in previous investigations of other software systems, so it guarantees that the instrument is designed correctly. In turn, the questionnaire was applied electronically given its multiple benefits. The purpose of the evaluation being carried out in this way was that the users, when evaluating the system, made the least possible errors, in addition to all the questions that had to be answered and. especially. the data will be collected automatically, avoiding that each of the surveys had to be captured later, with the risk that, at the time of this step, mistakes were made. The evaluation of the system was carried out with students of the degree in Engineering in Computer Systems. The students of this degree were specifically chosen, given that they are the best qualified in the use of computer equipment and in turn in the use of systems, since within their curriculum they carry subjects about development methodologies, standards of quality, software engineering, among others; that is, they would be more demanding when evaluating the final version of the application. In this way, their evaluations and criticisms would better support the work of the thesis. Once the evaluation was done, it was necessary to interpret the data obtained to convert it into information. The information obtained was represented in tables in order to better observe the results and in turn the interpretation would be easier. The results obtained showed that of the 5 factors evaluated (Comprehension, Learning, Operation, Attractiveness and Acceptance). The study sought a percentage of at least 90% for software quality. The comprehension and attractiveness factor obtained an evaluation above the required value, while the remaining ones remained very close to this value, so a more detailed study of each of these factors was carried out to identify the possible causes and in turn grant them a solution. Each of the variables that comprised each factor and all those that were below 90% of the quality were analyzed, solutions were proposed that could improve the new version of the system. Once this part was finished, a new version of the system was developed, following the suggested improvements and user comments. Finally, comparisons of both versions of the systems were shown to highlight the improvements made.

The main objective of the present thesis was to develop an efficient solution for the proposed Turin Pattern recognition algorithm, which has the capability of entering different potential scenarios for representing a model of chemical substances. This objective was satisfactorily achieved by developing three different approaches, which were tested got the best solution based on performance. The approach that uses the multiprocessing technique was found as the best of them. This solution is significantly better than the first version obtained (as shown in Chapter 3). Additionally to the developed solution, a complete software application was developed that allows end-users to create desired scenarios easier in order to find the solution for such scenario. In addition, the final solution was proved to be easy-to use, easy-to- understand, and easy-to-learn, which was an additional objective of the present project. This objective was proved by testing the software application by end-users. Thus, both objectives were achieved. Finally, this application can be used in various types of operating systems such as Windows and Linux operating systems.

A. Fortran code

In this appendix, we provide a Fortran code to solve system (1.7). The code is one of the first versions employed to simulate the solutions of that system. However, it is easy to provide a more efficient implementation of this software, and adapt it for different or more general problems.

```
program fhs3d
  use omp_lib
  implicit none

  real, parameter :: a=7.45
  real, parameter :: b=0.5
  real, parameter :: c=5
  real, parameter :: du=1
  real, parameter :: dv=20
  real, parameter :: tu=1
  real, parameter :: tv=1
  real, parameter :: alpha=1.6
  real, parameter :: beta=1.6
  real, parameter :: T=1
  real, parameter :: Long=100

  real, parameter :: tau=0.02
  real, parameter :: h=1

  integer, parameter :: M=floor(Long/h)+1
  integer, parameter :: NumIt=floor(T/tau)+1

  real, parameter :: ru1=4*tu/(2*tu+tau)
  real, parameter :: ru2=(2*tu-tau)/(2*tu+tau)
  real, parameter :: ru3=2*tau*tau/(2*tu+tau)
  real, parameter :: rv1=4*tv/(2*tv+tau)
  real, parameter :: rv2=(2*tv-tau)/(2*tv+tau)
  real, parameter :: rv3=2*tau*tau/(2*tv+tau)

  real, dimension(:), allocatable :: ga,gb
  real, dimension(:,:), allocatable :: Ha,Hb
  real, dimension(:,:,:), allocatable :: U1,U2,U3,V1,V2,V3,W,Z
  real, dimension(:,:,:), allocatable :: frac1x,frac2x
  real, dimension(:,:,:), allocatable :: frac1y,frac2y
```

```
real, dimension(:,:,:), allocatable :: frac1z,frac2z

integer :: i,j,k,l,n

allocate(ga(M),gb(M))
allocate(Ha(M,M),Hb(M,M))
allocate(U1(M,M,M),U2(M,M,M),U3(M,M,M),W(M,M,M))
allocate(V1(M,M,M),V2(M,M,M),V3(M,M,M),Z(M,M,M))
allocate(frac1x(M,M,M),frac1y(M,M,M),frac2z(M,M,M))
allocate(frac2x(M,M,M),frac2y(M,M,M),frac1z(M,M,M))

do i=1,M
  do j=1,M
    do k=1,M
      U1(i,j,k)=0.06*rand()-0.03
      V2(i,j,k)=0.06*rand()-0.03
    end do
  end do
end do
U2=U1
V2=V1

ga(1)=ru3*du*gamma(alpha+1)/gamma(0.5*alpha+1)**2/h**alpha
gb(1)=rv3*dv*gamma(beta+1)/gamma(0.5*beta+1)**2/h**beta

do l=1,M-1
  ga(l+1)=(1-(alpha+1)/(0.5*alpha+l))*ga(l)
  gb(l+1)=(1-(beta+1)/(0.5*beta+l))*gb(l)
end do

do i=1,M
  do j=1,M
    Ha(i,j)=ga(abs(i-j)+1)
    Hb(i,j)=gb(abs(i-j)+1)
  end do
end do

do n=3,NumIt
  !$omp parallel
    !$omp do
      do k=1,M
        frac1x(:,:,k)=MATMUL(Ha,U2(:,:,k))
        frac2x(:,:,k)=MATMUL(Hb,V2(:,:,k))
        frac1y(k,:,:)=MATMUL(Ha,U2(k,:,:))
        frac2y(k,:,:)=MATMUL(Hb,V2(k,:,:))
        frac1z(k,:,:)=MATMUL(U2(k,:,:),Ha)
        frac2z(k,:,:)=MATMUL(V2(k,:,:),Hb)
      end do
    !$omp end do
  !$omp end parallel

  !$omp parallel
    !$omp sections
```

```
        !$omp section
          W=U2-a*V2+b*U2*V2-U2**3
          U3=ru1*U2-ru2*U1-frac1x-frac1y-frac1z+ru3*W
          U1=U2
          U2=U3
        !$omp section
          Z=U2-c*V2
          V3=rv1*V2-rv2*V1-frac2x-frac2y-frac2z+rv3*Z
          V1=V2
          V2=V3
      !$omp end sections
    !$omp end parallel
  end do

  deallocate(ga,gb,Ha,Hb)
  deallocate(U1,U2,U3,V1,V2,V3,W,Z)
  deallocate(frac1x,frac2x,frac1y,frac2y,frac1z,frac2z)
end program fhs3d
```

Bibliography

[1] V. Dufiet and J. Boissonade, “Dynamics of turing pattern monolayers close to onset,” *Physical Review E*, vol. 53, no. 5, p. 4883, 1996.

[2] A. De Wit, “Spatial patterns and spatiotemporal dynamics in chemical systems,” *Advances in Chemical Physics, Volume 109*, pp. 435–513, 2007.

[3] B. Rudovics, E. Barillot, P. Davies, E. Dulos, J. Boissonade, and P. De Kepper, “Experimental studies and quantitative modeling of turing patterns in the (chlorine dioxide, iodine, malonic acid) reaction,” *The Journal of Physical Chemistry A*, vol. 103, no. 12, pp. 1790–1800, 1999.

[4] B. Rudovics, E. Dulos, and P. De Kepper, “Standard and nonstandard turing patterns and waves in the cima reaction,” *Physica Scripta*, vol. 1996, no. T67, p. 43, 1996.

[5] L. Yang and I. R. Epstein, “Oscillatory turing patterns in reaction-diffusion systems with two coupled layers,” *Physical Review Letters*, vol. 90, no. 17, p. 178303, 2003.

[6] A. Coillet, I. Balakireva, R. Henriet, K. Saleh, L. Larger, J. M. Dudley, C. R. Menyuk, and Y. K. Chembo, “Azimuthal turing patterns, bright and dark cavity solitons in kerr combs generated with whispering-gallery-mode resonators,” *IEEE Photonics Journal*, vol. 5, no. 4, pp. 6100409–6100409, 2013.

[7] S. Kondo and T. Miura, “Reaction-diffusion model as a framework for understanding biological pattern formation,” *Science*, vol. 329, no. 5999, pp. 1616–1620, 2010.

[8] J. H. Cartwright, “Labyrinthine turing pattern formation in the cerebral cortex,” *Journal of Theoretical Biology*, vol. 217, no. 1, pp. 97–103, 2002.

[9] M. D. Morales-Hernández, I. E. Medina-Ramírez, F. Avelar-González, and J. E. Macías-Díaz, “An efficient recursive algorithm in the computational simulation of the bounded growth of biological films,” *International Journal of Computational Methods*, vol. 9, no. 04, p. 1250050, 2012.

[10] B. Pena and C. Perez-Garcia, “Stability of turing patterns in the brusselator model,” *Physical Review E*, vol. 64, no. 5, p. 056213, 2001.

[11] T. Biancalani, D. Fanelli, and F. Di Patti, “Stochastic turing patterns in the brusselator model,” *Physical Review E*, vol. 81, no. 4, p. 046215, 2010.

[12] X. Tang and Y. Song, “Bifurcation analysis and turing instability in a diffusive predator-prey model with herd behavior and hyperbolic mortality,” *Chaos, Solitons & Fractals*, vol. 81, pp. 303–314, 2015.

[13] T. Zhang, Y. Xing, H. Zang, and M. Han, “Spatio-temporal dynamics of a reaction-diffusion system for a predator–prey model with hyperbolic mortality,” *Nonlinear Dynamics*, vol. 78, no. 1, pp. 265–277, 2014.

[14] F. Lutscher, A. Stevens, *et al.*, “Emerging patterns in a hyperbolic model for locally interacting cell systems,” *Journal of Nonlinear Science*, vol. 12, no. 6, pp. 619–640, 2002.

[15] O. B. Isaeva, A. S. Kuznetsov, and S. P. Kuznetsov, “Hyperbolic chaos of standing wave patterns generated parametrically by a modulated pump source,” *Physical Review E*, vol. 87, no. 4, p. 040901, 2013.

[16] E. Barbera, G. Consolo, and G. Valenti, “Spread of infectious diseases in a hyperbolic reaction-diffusion susceptible-infected-removed model,” *Physical Review E*, vol. 88, no. 5, p. 052719, 2013.

[17] U.-I. Cho and B. C. Eu, “Hyperbolic reaction-diffusion equations and chemical oscillations in the brusselator,” *Physica D: Nonlinear Phenomena*, vol. 68, no. 3-4, pp. 351–363, 1993.

[18] M. Al-Ghoul and B. C. Eu, “Hyperbolic reaction- diffusion equations, patterns, and phase speeds for the Brusselator,” *The Journal of Physical Chemistry*, vol. 100, no. 49, pp. 18900–18910, 1996.

[19] R. Eftimie, “Hyperbolic and kinetic models for self-organized biological aggregations and movement: a brief review,” *Journal of Mathematical Biology*, vol. 65, no. 1, pp. 35–75, 2012.

[20] M. Wolfrum, “The turing bifurcation in network systems: Collective patterns and single differentiated nodes,” *Physica D: Nonlinear Phenomena*, vol. 241, no. 16, pp. 1351–1357, 2012.

[21] J. Xu, G. Yang, H. Xi, and J. Su, “Pattern dynamics of a predator–prey reaction–diffusion model with spatiotemporal delay,” *Nonlinear Dynamics*, vol. 81, no. 4, pp. 2155–2163, 2015.

[22] G. Consolo, C. Currò, and G. Valenti, “Pattern formation and modulation in a hyperbolic vegetation model for semiarid environments,” *Applied Mathematical Modelling*, vol. 43, pp. 372–392, 2017.

[23] V. E. Tarasov, “Continuous limit of discrete systems with long-range interaction,” *Journal of Physics A: Mathematical and General*, vol. 39, no. 48, p. 14895, 2006.

[24] V. E. Tarasov and G. M. Zaslavsky, “Conservation laws and hamilton’s equations for systems with long-range interaction and memory,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 13, no. 9, pp. 1860–1878, 2008.

[25] R. Koeller, “Applications of fractional calculus to the theory of viscoelasticity,” *ASME, Transactions, Journal of Applied Mechanics(ISSN 0021-8936)*, vol. 51, pp. 299–307, 1984.

[26] Y. Povstenko, “Theory of thermoelasticity based on the space-time-fractional heat conduction equation,” *Physica Scripta*, vol. 2009, no. T136, p. 014017, 2009.

[27] E. Scalas, R. Gorenflo, and F. Mainardi, “Fractional calculus and continuous-time finance,” *Physica A: Statistical Mechanics and its Applications*, vol. 284, no. 1, pp. 376–384, 2000.

[28] W. G. Glöckle and T. F. Nonnenmacher, “A fractional calculus approach to self-similar protein dynamics,” *Biophysical Journal*, vol. 68, no. 1, pp. 46–53, 1995.

[29] V. Namias, “The fractional order fourier transform and its application to quantum mechanics,” *IMA Journal of Applied Mathematics*, vol. 25, no. 3, pp. 241–265, 1980.

[30] N. Su, P. N. Nelson, and S. Connor, “The distributed-order fractional diffusion-wave equation of groundwater flow: Theory and application to pumping and slug tests,” *Journal of Hydrology*, vol. 529, Part 3, pp. 1262–1273, 2015.

[31] V. G. Pimenov, A. S. Hendy, and R. H. De Staelen, "On a class of non-linear delay distributed order fractional diffusion equations," *Journal of Computational and Applied Mathematics*, vol. 318, pp. 433–443, 2017.

[32] J. E. Macías-Díaz, "Sufficient conditions for the preservation of the boundedness in a numerical method for a physical model with transport memory and nonlinear damping," *Computer Physics Communications*, vol. 182, no. 12, pp. 2471–2478, 2011.

[33] J. E. Macías-Díaz, "An explicit dissipation-preserving method for riesz space-fractional nonlinear wave equations in multiple dimensions," *Communications in Nonlinear Science and Numerical Simulation*, vol. 59, pp. 67–87, 2018.

[34] J. E. Macías-Díaz, "On the solution of a riesz space-fractional nonlinear wave equation through an efficient and energy-invariant scheme," *International Journal of Computer Mathematics*, vol. accepted for publication, pp. 1–25, 2018.

[35] A. A. Alikhanov, "A new difference scheme for the time fractional diffusion equation," *Journal of Computational Physics*, vol. 280, pp. 424–438, 2015.

[36] A. H. Bhrawy and M. A. Abdelkawy, "A fully spectral collocation approximation for multi-dimensional fractional schrödinger equations," *Journal of Computational Physics*, vol. 294, pp. 462–483, 2015.

[37] A. El-Ajou, O. A. Arqub, and S. Momani, "Approximate analytical solution of the nonlinear fractional kdv–burgers equation: a new iterative algorithm," *Journal of Computational Physics*, vol. 293, pp. 81–95, 2015.

[38] F. Liu, P. Zhuang, I. Turner, V. Anh, and K. Burrage, "A semi-alternating direction method for a 2-d fractional fitzhugh–nagumo monodomain model on an approximate irregular domain," *Journal of Computational Physics*, vol. 293, pp. 252–263, 2015.

[39] H. Ye, F. Liu, and V. Anh, "Compact difference scheme for distributed-order time-fractional diffusion-wave equation on bounded domains," *Journal of Computational Physics*, vol. 298, pp. 652–660, 2015.

[40] J. E. Macías-Díaz, "A structure-preserving method for a class of nonlinear dissipative wave equations with riesz space-fractional derivatives," *Journal of Computational Physics*, vol. 351, pp. 40–58, 2017.

[41] T. Langlands, B. Henry, and S. Wearne, "Turing pattern formation with fractional diffusion and fractional reactions," *Journal of Physics: Condensed Matter*, vol. 19, no. 6, p. 065115, 2007.

[42] V. Gafiychuk and B. Datsko, "Spatiotemporal pattern formation in fractional reaction-diffusion systems with indices of different order," *Physical Review E*, vol. 77, no. 6, p. 066210, 2008.

[43] B. Datsko, Y. Luchko, and V. Gafiychuk, "Pattern formation in fractional reaction–diffusion systems with multiple homogeneous states," *International Journal of Bifurcation and Chaos*, vol. 22, no. 04, p. 1250087, 2012.

[44] A. Mvogo, J. E. Macías-Díaz, and T. C. Kofané, "Diffusive instabilities in a hyperbolic activator-inhibitor system with superdiffusion," *Physical Review E*, vol. 97, no. 3, p. 032129, 2018.

[45] B. I. Henry and S. L. Wearne, "Existence of turing instabilities in a two-species fractional reaction-diffusion system," *SIAM Journal on Applied Mathematics*, vol. 62, no. 3, pp. 870–887, 2002.

[46] Y. Nec and A. Nepomnyashchy, "Turing instability in sub-diffusive reaction–diffusion systems," *Journal of Physics A: Mathematical and Theoretical*, vol. 40, no. 49, p. 14687, 2007.

[47] D. Jeong, Y. Choi, and J. Kim, "Modeling and simulation of the hexagonal pattern formation of honeycombs by the immersed boundary method," *Communications in Nonlinear Science and Numerical Simulation*, vol. 62, pp. 61–77, 2018.

[48] D. Lacitignola, B. Bozzini, M. Frittelli, and I. Sgura, "Turing pattern formation on the sphere for a morphochemical reaction-diffusion model for electrodeposition," *Communications in Nonlinear Science and Numerical Simulation*, vol. 48, pp. 484–508, 2017.

[49] X. Wang, W. Wang, and G. Zhang, "Vegetation pattern formation of a water-biomass model," *Communications in Nonlinear Science and Numerical Simulation*, vol. 42, pp. 571–584, 2017.

[50] D. Prakasha, P. Veeresha, and H. M. Baskonus, "Two novel computational techniques for fractional gardner and cahn-hilliard equations," *Computational and Mathematical Methods*, vol. 1, no. 2, p. e1021, 2019.

[51] A. Q. M. Khaliq, X. Liang, and K. M. Furati, "A fourth-order implicit-explicit scheme for the space fractional nonlinear schrö dinger equations," *Numerical Algorithms*, vol. 75, no. 1, pp. 147–172, 2017.

[52] X. Liang, A. Q. M. Khaliq, H. Bhatt, and K. M. Furati, "The locally extrapolated exponential splitting scheme for multi-dimensional nonlinear space-fractional schrö dinger equations," *Numerical Algorithms*, vol. 76, no. 4, pp. 939–958, 2017.

[53] K. M. Furati, M. Yousuf, and A. Q. M. Khaliq, "Fourth-order methods for space fractional reaction–diffusion equations with non-smooth data," *International Journal of Computer Mathematics*, vol. 95, no. 6-7, pp. 1240–1256, 2018.

[54] Q.-J. Meng, D. Ding, and Q. Sheng, "Preconditioned iterative methods for fractional diffusion models in finance," *Numerical Methods for Partial Differential Equations*, vol. 31, no. 5, pp. 1382–1395, 2015.

[55] J. E. Macías-Díaz, A. S. Hendy, and R. H. De Staelen, "A pseudo energy-invariant method for relativistic wave equations with riesz space-fractional derivatives," *Computer Physics Communications*, vol. 224, pp. 98–107, 2018.

[56] J. E. Macías-Díaz and J. M. G. Reynoso, "Utilizando el modelo de calidad de mccall y el estándar iso-9126 para la evaluación de la calidad de sistemas de información por los usuarios," 2010.

[57] H. Ramos, "Development of a new runge-kutta method and its economical implementation," *Computational and Mathematical Methods*, vol. 1, no. 2, p. e1016, 2019.

[58] I. Podlubny, *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications*, vol. 198. Elsevier, 1998.

[59] W. Wang, Q.-X. Liu, and Z. Jin, "Spatiotemporal complexity of a ratio-dependent predator-prey system," *Physical Review E*, vol. 75, no. 5, p. 051913, 2007.

[60] M. D. Ortigueira, "Riesz potential operators and inverses via fractional centred derivatives," *International Journal of Mathematics and Mathematical Sciences*, vol. 2006, 2006.

[61] C. Çelik and M. Duman, "Crank–nicolson method for the fractional diffusion equation with the riesz fractional derivative," *Journal of Computational Physics*, vol. 231, no. 4, pp. 1743–1750, 2012.

[62] J. E. Macías-Díaz, "Numerical simulation of the nonlinear dynamics of harmonically driven riesz-fractional extensions of the fermi–pasta–ulam chains," *Communications in Nonlinear Science and Numerical Simulation*, vol. 55, pp. 248–264, 2018.

[63] K. Pen-Yu, “Numerical methods for incompressible viscous flow,” *Scientia Sinica*, vol. 20, pp. 287–304, 1977.

[64] R. S. Pressman, *Software engineering : a practitioner’s approach.* New York: McGraw-Hill Higher Education, 7th ed., 2010.

[65] J. E. Moreira, S. P. Midkiff, and M. Gupta, “A comparison of java, c/c++, and fortran for numerical computing,” *IEEE Antennas and Propagation Magazine*, vol. 40, no. 5, pp. 102–105, 1998.

[66] G. Hager and G. Wellein, *Introduction to high performance computing for scientists and engineers.* CRC Press, 2010.

[67] R. M. Baecker, *Readings in human-computer interaction : toward the year 2000.* San Francisco, Calif.: Morgan Kaufmann Publishers, 2nd ed., 1995.

[68] T. K. Landauer, *The trouble with computers : usefulness, usability, and productivity.* Cambridge, Mass.: MIT Press, 1995.

[69] I. Sommerville, *Software engineering.* Boston: Pearson, tenth edition. ed., 2016.

[70] J. Martin, *Rapid application development.* New York Toronto: Macmillan Pub. Co. ; Collier Macmillan Canada ; Maxwell Macmillan International, 1991.

[71] J. F. Hair, *Multivariate data analysis.* Upper Saddle River, NJ: Prentice Hall, 7th ed., 2010.