Fundamental Molecular Communication Modelling

Thesis by

Nadezhda Briantceva

In Partial Fulfillment of the Requirements

For the Degree of

Masters of Science

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

July, 2020
The thesis of Nadezhda Briantceva is approved by the examination committee.

Committee Chairperson: Prof. Mohamed-Slim Alouini
Committee Co-Chairperson: Prof. Matteo Parsani
Committee Member: Prof. David Keyes
ABSTRACT

Fundamental Molecular Communication Modelling
Nadezhda Briantceva

As traditional communication technology we use in our day-to-day life reaches its limitations, the international community searches for new methods to communicate information. One such novel approach is the so-called molecular communication system. During the last few decades, molecular communication systems become more and more popular. The main difference between traditional communication and molecular communication systems is that in the latter, information transfer occurs through chemical means, most often between microorganisms. This process already happens all around us naturally, for example, in the human body. Even though the molecular communication topic is attractive to researchers, and a lot of theoretical results are available - one cannot claim the same about the practical use of molecular communication. As for experimental results, a few studies have been done on the macroscale, but investigations at the micro- and nanoscale ranges are still lacking because they are a challenging task. In this work, a self-contained introduction of the underlying theory of molecular communication is provided, which includes knowledge from different areas such as biology, chemistry, communication theory, and applied mathematics. Two numerical methods are implemented for three well-studied partial differential equations of the MC field where advection, diffusion, and the reaction are taken into account. Numerical results for test cases in one and three dimensions are presented and discussed in detail. Conclusions and essential analytical and numerical future directions are then drawn.
ACKNOWLEDGEMENTS

I am grateful to my supervisor, Professor Mohamed-Slim Alouini who enabled me to work on such a promising topic, for his continuous assistance, and depth of his experience.

I am also grateful to my co-advisor, Professor Matteo Parsani, for his constant guidance, essential tips on science and life, valuable corrections, and his sense of humor.

I am also grateful to Professor David Keyes for taking the time to join my defense and be a committee member, for his precious help and comments that allowed me to go deeper into the thesis topic.

I am also thankful to Dr. Lisandro Dalcin for his assistance in solving many technical issues related to my work and his openness.

I also appreciate the constant supportive conversations with my friends who are located at KAUST and outside.

Finally, I am deeply grateful to my parents, Elena and Vladimir, and my husband Dmitrii for their patience, faith in me, and support.
# TABLE OF CONTENTS

**EXAMINATION COMMITTEE PAGE** 2

**COPYRIGHT** 3

**ABSTRACT** 4

**ACKNOWLEDGEMENTS** 5

**TABLE OF CONTENTS** 6

**LIST OF ABBREVIATIONS** 8

**LIST OF SYMBOLS** 9

**LIST OF FIGURES** 11

1 Introduction 13
   1.1 Motivation .................................................. 13
   1.2 Applications .................................................. 18

2 Molecular communication theory 22
   2.1 Introduction .................................................. 22
   2.2 Components of the MC system .................................. 22
      2.2.1 Transmitter ................................................. 23
      2.2.2 Channel ..................................................... 26
      2.2.3 Receiver .................................................... 29
   2.3 Fundamental laws for the channel modeling in MC field ....... 30
      2.3.1 Diffusion model ............................................. 32
      2.3.2 Advection-diffusion model ................................ 34
      2.3.3 Advection-diffusion-reaction model ....................... 35
   2.4 Conclusion ..................................................... 36
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AANSLab</td>
<td>Advanced Algorithm and Numerical Simulations Laboratory</td>
</tr>
<tr>
<td>ADR</td>
<td>Advection-Diffusion-Reaction</td>
</tr>
<tr>
<td>ATP</td>
<td>Adenosine Triphosphate</td>
</tr>
<tr>
<td>BC</td>
<td>Boundary Condition</td>
</tr>
<tr>
<td>CEA</td>
<td>Carcinoembryonic Asntigen</td>
</tr>
<tr>
<td>CFD</td>
<td>Computational Fluid Dynamics</td>
</tr>
<tr>
<td>CNSADR</td>
<td>Coupling of the Navier-Stokes with the Advection-Diffusion-Reaction</td>
</tr>
<tr>
<td>CTC</td>
<td>Circulating Tumor Cell</td>
</tr>
<tr>
<td>DG</td>
<td>Discontinuous Galerkin</td>
</tr>
<tr>
<td>DNA</td>
<td>Deoxyribonucleic Acid</td>
</tr>
<tr>
<td>FDM</td>
<td>Finite Difference Scheme</td>
</tr>
<tr>
<td>FTCS</td>
<td>Forward Time Centered Space</td>
</tr>
<tr>
<td>GCL</td>
<td>Geometric Conservation Law</td>
</tr>
<tr>
<td>IC</td>
<td>Initial Condition</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>IoNT</td>
<td>Internet-of-Nano-Things</td>
</tr>
<tr>
<td>ISI</td>
<td>Inter-Symbol Interference</td>
</tr>
<tr>
<td>LGL</td>
<td>Legendre-Gauss-Lobatto</td>
</tr>
<tr>
<td>MC</td>
<td>Molecular Communication</td>
</tr>
<tr>
<td>MEMS</td>
<td>Micro-Electro-Mechanical System</td>
</tr>
<tr>
<td>MRI</td>
<td>Magnetic Resonance Imaging</td>
</tr>
<tr>
<td>ODE</td>
<td>Ordinary Differential Equation</td>
</tr>
<tr>
<td>PDE</td>
<td>Partial Differential Equation</td>
</tr>
<tr>
<td>PETSc</td>
<td>Portable and Extensible Toolkit for Scientific computing</td>
</tr>
<tr>
<td>pH</td>
<td>power of Hydrogen</td>
</tr>
<tr>
<td>RNA</td>
<td>Ribonucleic Acid</td>
</tr>
<tr>
<td>Rx</td>
<td>Receiver</td>
</tr>
<tr>
<td>SAT</td>
<td>Simultaneous-Approximation-Terms</td>
</tr>
<tr>
<td>SBP</td>
<td>Summation-by-parts</td>
</tr>
<tr>
<td>SMIEET</td>
<td>Simultaneous Molecular Information and Energy Transfer</td>
</tr>
<tr>
<td>SSDC</td>
<td>Entropy Stable Discontinuous Collocation Spectral Element</td>
</tr>
<tr>
<td>Tx</td>
<td>Transmitter</td>
</tr>
</tbody>
</table>
LIST OF SYMBOLS

$\Delta t$  Step size in time
$D$  Diffusion coefficient
$\mathcal{N}$  Gaussian random variable
$\mu$  Mean of the Gaussian random variable
$\sigma^2$  Variance of the Gaussian random variable
$C$  Concentration of molecules
$\Delta$  Laplacian
$\frac{\partial^2}{\partial x_i^2}$  Second partial derivative w.r.t. the Cartesian coordinates $x_i$
$C^*$  Analytical solution for the concentration of molecules
$\delta(\cdot)$  Dirac delta function
$\infty$  Infinity
$\| \cdot \|_p$  $p$-norm
$t_0$  Information molecules release time
$N$  Number of released information molecules
$\pi$  Mathematical constant $\approx 3.1416$
$\dim$  Dimension of the space
$v$  Flow velocity vector
$D$  Diffusion vector
$\nabla$  Nambla operator
$r$  Identifier which shows whether the molecule is a product or a reactant of the reaction
$k_f$  Degradation rate constant
$\Delta x$  Step size in space in x direction
$\Delta y$  Step size in space in y direction
$\Delta z$  Step size in space in z direction
$\Delta t$  Time step
$\Omega$  Open connected set
$\partial \Omega$  Boundary of the domain $\Omega$
$h$  Boundary data
$g$  Initial data
$J_\kappa$  Determinant of the metric Jacobian
$N_j$  Number of LGL points per direction in a given element
$\mathds{1}_{N_j}$  $N_j \times N_j$ identity matrix
\(D^{(1D)}_{x'j}\) Matrix SBP operator
\(J_{\kappa}\) Diagonal matrix with the metric Jacobians
\(SAT_{k}\) SATs vector
\(1_{\kappa}\) Vector of ones of size \(N_{\kappa}\)
\(t_{\text{init}}\) Initial simulation time
\(T\) Final simulation time
\(U\) Numerical solution
\(u\) Analytical solution
\(E\) Error
\(x_{\text{min}}\) Minimum boundary of the simulation area
\(x_{\text{max}}\) Maximum boundary of the simulation area
\(x_{tx}\) Transmitter location
\(x_{rx}\) Receiver location
LIST OF FIGURES

1.1 The comparison of telecommunication and molecular communication techniques [18]. .................................................. 14
1.2 Major application areas of molecular communication. .......... 19

2.1 Simplified molecular communication system diagram. .......... 23
2.2 Two main types of a cell: eukaryote cell (left) and prokaryote cell (right). 25
2.3 Examples and characteristic scales of nature-made bio-nanomachines [20]. 26
2.4 Propagation of a molecules through gap junction channels. ....... 28
2.5 Propagation of molecules using a transport molecule. ............ 29
2.6 Propagation of molecules using a bacterial motor. ............... 29
2.7 Main processes affecting the propagation of signaling molecules where (a) corresponds to the first time slot and (b)-(e) to the second time slot. 32

4.1 Concentration of molecules $C^*(x,t)$ [molecules/m] versus time $t[\mu s]$ at different locations of the receiver $x_{rx}$ for the 1D diffusion simulation. 48
4.2 Global $L^\infty$-error versus time $t[\mu s]$ for the 1D diffusion simulation. .. 49
4.3 Concentration of molecules $C^*(x,t)$ [molecules/m] versus time $t[\mu s]$ with different flow velocities for the 1D advection-diffusion simulation. 50
4.4 Global $L^\infty$-error versus time $t[\mu s]$ for 1D advection-diffusion simulation under the different flow velocities and $x \in \{-1700,1700\}$ nm. ........ 51
4.5 Global $L^\infty$-error versus time $t[\mu s]$ for 1D advection-diffusion simulation under the different flow velocities and $x \in \{-3000,3000\}$ nm. ......... 52
4.6 Concentration of molecules $C^*(x,t)$ [molecules/m] versus time $t[\mu s]$ with different degradation rate constants for the 1D advection-reaction-diffusion simulation. ......................... 53
4.7 Global $L^\infty$-error versus time $t[\mu s]$ for 1D advection-reaction-diffusion simulation under the different rate constants and $x \in \{-1700,1700\}$ nm. 54
4.8 Global $L^\infty$-error versus time $t[\mu s]$ for 1D advection-reaction-diffusion simulation under the different rate constants and $x \in \{-3000,3000\}$ nm. 55
4.9 Concentration of molecules $C^*(x,t)$ [molecules/m$^3$] versus time $[\mu s]$ at different Rx locations $x_{rx}$ for the 3D diffusion simulation. ............ 60
4.10 Concentration of molecules $C^*(x, t)$ [molecules/m$^3$] versus time [$\mu$s]
with different flow velocities for the 3D advection-diffusion simulation. 62

4.11 Concentration of molecules $C^*(x, t)$ [molecules/m$^3$] versus time [$\mu$s]
with different degradation rate constants for the 3D advection-reaction-diffusion simulation. 63

5.1 Concentration of molecules $C^*(x, t)$ [molecules/m] versus time [$\mu$s] with
different degradation rate constants for the 1D advection-reaction-diffusion simulation. 67

5.2 Concentration of molecules $C^*(x, t)$ [molecules/m$^3$] versus time [$\mu$s]
with different degradation rate constants for the 3D advection-reaction-diffusion simulation. 68

5.3 The MC system with the presence of the Poiseuille flow. 70

5.4 Concentration of molecules $C^*(x, t)$ [molecules/m$^2$] versus time [minutes]
with different flow velocities profiles of the 2D advection-diffusion model. 71

5.5 Concentration of molecules $C^*(x, t)$ [molecules/mm$^2$] versus time [minutes]
with different flow velocities profiles of the 2D advection-reaction-diffusion model. 73
Chapter 1

Introduction

1.1 Motivation

Molecular communication (MC) is a new communication concept in which sharing the information among microorganisms whose size varies from 1 to 100 nanometers takes place. There is no doubt that the interaction of molecules in nature has been present for a very long time, but, in spite of this fact, active research of this area began only two decades ago, during which significant results were obtained [1], [2]. The increase of interest in MC was particularly affected by the mention of the main concepts in [3] in 2005. A considerable amount of work in this area confirms the relevance of this topic right now. Most of the works in this area are theoretical, but there are some experimental works; for example, [4], where the properties of macro-scale MC are experimented and studied. A question about why exactly we need this type of communication may arise, and the answer is quite simple: MC paradigm provides a promising technique based on the cooperation of natural biological nanomachines. Molecules are used everywhere, and due to their sizes, it can be possible to create weightless bio-nanonetworks, which constitute thousands and millions of nanomachines capable of performing simple computations.

Consideration of the table below with the main differences between traditional and molecular communications can be useful for understanding the strengths and weaknesses of both paradigms.
Figure 1.1: The comparison of telecommunication and molecular communication techniques [18].

There are such advantages of the MC system over traditional communication systems as:

1. Functional complexity: it was mentioned above that a bio-nanonetwork consists of many devices so we can divide a complex problem into small tasks and assign them to each unit of the network.

2. Feasibility: MC is easier to implement compared to other techniques (for example, telecommunication based on electrical or optical signals which use space or communication cables to transmit information).

3. Scalability: nano-to-micro bio-nanomachines can to communicate at the small ranges.

4. The simultaneous solution of a large number of tasks can become available since it can be possible to split tasks among all robots inside the bio-nanonetwork.

5. Bio-compatibility: it can be possible to integrate such systems with biological systems.
6. Self-replicating of bio-nanomachines is an attractive feature of this nanonetworks, which can improve the efficiency of a whole system.

7. Energy efficiency: biochemical reactions have high efficiencies and they occur during communication very often.

Other features can be found, which depend on the implementation of the exact MC system. Despite all the positive properties of these systems, we can also face challenges such as a) stochasticity: random propagation of molecules in the physical channel, environmental noise (for example, interfering noise molecules, noise by advection, diffusion or reaction), b) propagation delay: the movement of molecules throughout the channel may take a very long time so that some of these molecules can be considered lost; c) coverage area: one of the main challenges of MC systems is the molecule propagation over short distances; d) fragility: biological components can be very sensitive to environmental changes such as temperature, pH, viscosity, flow velocity, other reagents and etc. It is also worth noting the difference in such terms as path loss, noise, bandwidth, channel capacity and multi-path fading, that arise as a result of the propagation which presented in the above Fig. 1.1. Path loss in telecommunication systems can be decreased by changing whether the propagation distance or signal frequencies, whereas, in MC, the molecule absorption loss occurs due to the consumption of the energy by the additional molecules presented in the medium [5].

Talking about the bandwidth in MC, it can be noted that due to the dependence of the molecular absorption from the medium characteristics, we will get the medium sensitiveness of the bandwidth capacity. In should be mentioned that the capacity of the MC channel (b/s) is much slower compared to the traditional one (Kb/s - Tb/s). In [6], it is shown that the minimization of the signal pulses lengths maximizes the bandwidth under the specified ranges of the diffusion-based physical channel. As we know, the communication systems performance has limitations caused by the energy loss during the signal propagation. In [7], the authors demonstrate the nano-scale
communications with ultra-low energy consumption. The two “simultaneous molecular information and energy transfer (SMIET)” nano-relay systems were implemented, where it is possible to achieve extremely high efficiency (12-25%) in energy harvesting. Moreover, it was also demonstrated that using the MCs in the group of nanomachines can make it possible to improve the energy harvesting efficiency by 2-5dB. With this, it may be possible to create systems where the 0 external power/bit cost is needed for the molecule propagation. Despite the fact that the diffusion-based molecular communication systems do not require external energy for molecules to propagate, there is still a need to use energy-efficient schemes that minimize the energy cost of the synthesis of molecules. In [8], it was demonstrated that using the relays with the SMIET capability, a lower minimum bit error probability along with the lower synthesis cost ($1.0604 \times 10^{-16}$ KJ) can be achieved via the conversion of the molecules or their division compared to the traditional relay systems ($1.3121 \times 10^{-16}$ KJ). In presented two-hop MC system, after absorbing the A type of molecules SMIET relay can decode the information along with the conversion of arrived type A of molecules into a different molecule type B to be sent by the relay transmitter.

Using MC systems, we can carry out targeted drug delivery in the human body; for example, in chemotherapy, we can directly deliver drugs to malignant tumors within the body. To do that we need to think about how sequentially to perform such tasks as:

- provide the system with a thousand nanorobots;
- guarantee the cooperation between these devices;
- ensure that such a bio-nanonetwork which allows robots to interact freely in the body;
- construct such a system so that a group of devices operates in the body without disrupting healthy organs;
• build such a strategy that will provide the nanorobots with the opportunity to get as much additional energy as possible from the environment because the robot sizes allow them to store only a small amount of energy;

• find a method by which robots are guaranteed to achieve the final goal (tumor in our case);

• create a way to determine the approximate amount of drugs that will be needed to destroy the tumor, depending on how many robots will reach the final place;

• construct such a system that will guarantee that the immune system will not destroy devices before completing the mission, and they will be able to release drugs in tumor places in a body.

Let’s discuss one of the possible ways to implement the detection of tumors in the human body. In [19], the detection of the circulating tumor cells (CTCs) approach in the bloodstream was proposed. Most of the time, the immune system of a healthy person can detect and remove any minor concentrations of the CTCs or DNA released by them and propagated via microvesicles through the circulatory system in the early stages of the propagation from the central tumor spot. The problem will occur when the immune system cannot detect small CTC concentrations or, for example, when the tumor cells excessively produce the biomarkers, causing the inability of them to be destroyed by macrophages. Thus, suitable MCs must be implemented to prevent previously mentioned situations. Different biomarkers located on the CTCs surface, such as the CD47, the microRNA-101, the carcinoembryonic antigen (CEA), the CD164 protein, can be detected on the contact with the circulating mobile nanomachines embedded in the human body acting as sensors. Another way for CTCs to be detected is to use the circulating nanomachines, able to absorb the microvesicles that transport the DNA emitted by the CTCs. After the CTC detection, sensor nanomachines release the information molecules into the channel and further detecting them to the
destination nanomachine where they can be stored. The nanomachine in charge of collecting the information molecules then can deliver it to an implanted device, which acts as a smart probe, responsible for exporting the information. The channel models proposed in this work can be used in the design of such MC systems for the detection of diseases in the early stages of development, as well as multiple others.

1.2 Applications

Molecular communications have received a lot of attention from the scientific community due to a wide range of applications, especially in medicine. A wide number of MC applications are closely related to the presence of a variety of interactions between natural biological nanomachines both at the intracellular (chemotactic signaling, vesicular trafficking), intercellular (calcium signaling, quorum signaling), and inter-organ levels (endocrine signaling, neuronal signaling). As natural participants of the above communication processes can be used proteins, DNA/RNA molecules, lipid membranes, and vesicles as separately as well as their combinations. On the figure 1.2 areas of MC application that are currently interesting for research are demonstrated. Brief descriptions of each application area is presented at the end of this chapter.
Figure 1.2: Major application areas of molecular communication.

- Biomedical (medical and healthcare) applications:
  
  - Targeted drug delivery: delivery of medicine to a specific area by cooperative group of biological nanomachines;
  
  - Health monitoring: integration of bio-nanomachines into a human body in order to gather information from specific places in the human body to diagnose and prevent a diseases on early stages;
  
  - Regenerative medicine: integration of bio-nanomachines into a human body in order to reconstruct damaged tissues or artificially create a new organs;
– Genetic engineering: direct DNA manipulation in a human organism using bio-nanomachines;

– Brain-machine interfaces: an implantation of biological nanomachines into the brain and an interaction between them and artificial devices from the outside which allow to monitor a different parts of the brain, detect a potentially dangerous places before the disease develops, premature treatment of mental or physical damage of the brain;

– Intracellular therapy: critical cell-to-cell communication process that maintains the biological functions and microenvironmental hemostasis of cells, organs, and intact systems [9];

– Micro-Electro-Mechanical Systems (MEMS): performing a chemical analysis by combining laboratory processes on a single chip which is called lab-on-a-chip [10].

• Environmental applications:

– Bio-sensor networks: integration of bio-nanomachines into different scale devices which can be located in the environment to perform various functions such as monitoring, a gathering of data of interest and further removal detected pollutants from the environment [11], [12];

– Recycling unnecessary products without harming the environment.

• Industrial applications:

– Manufacturing:

  * Quality control: product defect detection by using groups of bio-nanomachines;

  * Precise component construction;
* Functionality extension of existing products and creating new products through the integration of nanotechnologies.

– Agriculture.

• Information and communication technology (ICT) applications:

  – Bio-chip integrated technology;

  – Internet-of-Nano-Things (IoNT): integration of the nanosensors with a variety of devices and objects, interconnection of nanodevices with existing communication networks and the Internet, leading to better performance of many real applications [13];

  – Unconventional computation: creating a new-based computation networks by integration group of bio-nanomachines into non-silicon-based systems.
Chapter 2

Molecular communication theory

2.1 Introduction

Molecular communication is an interdisciplinary area. Specialists from different areas such as math, engineering, or communications do not necessarily need an in-depth knowledge of biology or chemistry to be able to investigate the MC topic. However, everyone who starts to study this field should first concentrate on basic knowledge of MC. This chapter will provide essential information on the MC systems: the main components of the system, their description, functionality, and their interaction with each other.

2.2 Components of the MC system

A simple communication system consists of three main components which will be discussed next. The first component is a sender bio-nanomachines or in other words a transmitter (Tx) responsible for encoding input information into a message and its further sending. Second is a receiver (Rx) bio-nanomachines, which receives that message at the end of the molecule propagation and decodes it. The last component is a channel that is located between the Tx and Rx, where the molecules are moving. Figure 2.1 gives an overview of an MC system where biological nanomachines act as senders/receivers. First, the Tx responds to information that comes from the external environment (for example, fluid, gas, air) and encodes this information into a molecule. Then a message (an information-encoded molecule) passes through the
physical channel (on the picture we can see the diffusion of molecules, i.e., they are transported passively) to the Rx, which finally will decode and deliver a result to the destination.

Figure 2.1: Simplified molecular communication system diagram.

2.2.1 Transmitter

As was mentioned above, the main goal of the Tx is to send information which should be encoded onto molecules. First, a Tx should generate the information, then encode obtained information onto molecules and, finally, release information (signal) molecules into the channel. There are such processes that a Tx should perform as:

1. Encoding. The first stage during which source information is encoded into molecules that finally should be detected by the receiver. Information may be encoded into various forms within the information molecules, such as in the three-dimensional structure of the information molecule (e.g., a specific type of molecule), in the specific molecules that compose the information molecules (e.g., DNA is formed by the specific sequence of nucleotides), or in the concentration of information molecules (i.e., the number of information molecules per unit volume of solvent molecules) modulated over time. [14].

2. Modulation. The second stage is responsible for changing the properties of information molecules in such a way, so we get a message to send by a Tx. Nowadays, there are three main groups of modulation techniques that differ in property in
what the information was encoded: molecule concentration, molecule type, or the molecule release time/molecule release order [15]. Modulation techniques play a very important role in the whole process of transmitting information, and knowing which scheme is more suitable in a particular case can significantly improve the communication rate [16].

3. Sending. The final stage that should be performed by a Tx during which information-encoded molecules are emitted into the channel.

Bio-nanomachines are actively involved in these processes since they work as a Tx. There are some important characteristics of bio-nanomachines:

- The size of bio-nanomachine depends on the type of that machine, for example, we have:
  - biological cells’ size ranges from 1 $\mu m$ to 100 $\mu m$;
  - single atoms are around 1 $nm$ in diameter;
  - tiny molecules are about 1 $nm$;
  - proteins range between 5 and 10 $nm$;
  - macromolecules (i.e., large molecules) have a diameter ranging from about $10^{-5}$ to $10^{-3}$ $mm$.

- They can be made either from biological (i.e., proteins, nucleic acids such as DNA and RNA molecules, carbohydrates, lipid membranes, vesicles, liposomes, biological cells) or human-made materials; a bio-nanomachine may also be a hybrid of both of them.

- Capabilities of each bio-nanomachine are also restricted since their sizes, a small amount of energy that they can store.
We can divide all living organisms into two main groups depending on the basic structure of their cells: prokaryotes which appeared on Earth 2.7 billion years ago and eukaryotes which appeared 1.2 billion years later than prokaryotes (Fig. 2.2).

![Diagram of cell structures](image)

**Figure 2.2:** Two main types of a cell: eukaryote cell (left) and prokaryote cell (right).

We can find such examples of biological nanomachines in Fig. 2.3 including their sizes as:

- Eukaryotes are living organisms, which are composed of cells which have a cell nucleus and membrane organelles, for example, animals, plants, fungi (Fig. 2.2);

- Prokaryotes are living organisms, which cells unlike eukaryotic cells do not have a cell nucleus and membrane organelles, e.g., bacteria (Fig. 2.2);

- Virus is a non-cellular tiny infectious microorganism that reproduces inside living cells and stimulate infections diseases;

- Vesicles are quite small intracellular organelles in which nutrients are stored or transported;

- DNA/RNA proteins are macromolecules that play an important rule in coding, reading, regulation, and expression of genes.
2.2.2 Channel

After the emission of information/signaling molecules (like proteins, ions, DNA) by the Tx, they propagate in the environment (in the aqueous medium in MC). There are two main ways for molecules to travel, such as:

- Passive transport. The molecules propagate randomly without using chemical energy due to the mechanisms which occur naturally in the environment. That includes, for example, diffusion, gap junction channels, advection, or possible reactions, which will be discussed later in detail. In this regime, we cannot guarantee that a certain number of molecules will reach a final destination by a predetermined time. Here we may also encounter the fact that molecules can enter into production or degradation reactions, for example, and the channel noise will occur, over which we have no control.

- Active transport. In this regime, a system may include such additional types of molecules as:
  - transport/carrier molecules (like rail molecules, hormones) are used when we want to move different types of signal molecules along the channel;
  - guide molecules are needed to artificially direct our signaling molecules to have a higher probability of information molecules reaching their final place;
– interface molecules which are responsible for providing communication
techniques to transport molecules or protecting information molecules from
a channel noise;

– addressing molecules whose main goal is to deliver information molecules
to a specific location, i.e., this type of molecule can be attached to signal
molecules or interface molecules.

There are few passive and active ways for information molecules to propagate in
the channel such as through:

• Free diffusion-based propagation, which is the passive way for molecules to
propagate, is illustrated in Fig. 2.7 (b).

• Advection-diffusion based propagation, which is the passive way for molecules
to propagate, is illustrated in Fig. 2.7 (d).

• Advection-reaction-diffusion based propagation, which is the passive way for
molecules to propagate, is illustrated in Fig. 2.7 (e) and this type of propagation
will be discussed in detail in the next chapter along with the previous two passive
ways of the molecule propagation.

• Gap junction channels (passive way to propagate) are mediate passes shown in
Fig. 2.4 between two cells connected with each other. An important function of
these channels is that they allow signaling molecules to move from the cytoplasm
of one cell to the cytoplasm of a neighboring cell.
Molecular motors (proteins), which are the active way to propagate, can generate motion by consuming chemical energy from the environment to ensure regular movement. It can be provided by breaking down adenosine triphosphate (ATP) to produce mechanical work [17]. ATP is the main carrier of energy in the cell, which stores chemical energy in its high-energy phosphoanhydride bonds. In the figure below, we can see an information molecule that not only binds to the transport molecule but also is contained in a vesicle-based interface molecule. The vesicle is responsible for protecting information molecule from external noise (for example, it is not possible for an information molecule to be engaged in chemical reaction while locating inside a vesicle).
Figure 2.5: Propagation of molecules using a transport molecule.

- Bacterial motors are an active way to propagate, where the bacteria move a signaling molecules through the physical channel by consuming chemical substances along the way and rotating their flagellas at a speed of about micrometers per seconds (Fig. 2.6). Bacteria also exchange DNA through the process of conjugation [18], i.e., bacteria can transmit information encoded in DNA to each other.

Figure 2.6: Propagation of molecules using a bacterial motor.

2.2.3 Receiver

A Rx is the third important component of MC systems. Propagation of signaling molecules occurs in the channel until they will appear in the vicinity of the Rx which should perform such actions as:
1. Receiving. The type of the molecule will affect the way which will be used by the Rx to decode the message once the information molecules reach the surrounding of the Rx. The choice of one of the possible Rx types such as passive, absorbing, and reacting one [15] will affect the way the message will be decoded. There such possible Rx surface properties closely related to the type of the Rx as:

- Permeability of the surface. An example is the plasma membrane of a biological cell is assumed to be permeable to all or some types of signal molecules. Therefore the signal molecules propagating in the environment can enter the cell and directly bind to the receptors within the cell [20].

- Surface absorption. Here the surface of the Rx must be able to consume all information molecules touching the surface.

- Surface reactivity. Under this property, molecules can be bind to the receptors on the Rx surface, and thus a chemical reaction will take place between them. The location of a larger number of receptors on the Rx surface guarantees a higher absorption rate. Another possible option is the gap junction channels located on the surface which can regulate entering information molecules into a receiver using internal mechanisms.

2. Decoding is the process by which the Rx decodes the incoming received message using its various states. One of the common ways to decode the collected molecules is a chemical reaction that occurs on the Rx surface. Different chemical reactions may occur during which we can obtain a new type of molecules or other information molecules.

2.3 Fundamental laws for the channel modeling in MC field

There are three major processes: diffusion, advection, and reaction, which govern the propagation of molecules along the channel. Let us discuss how these processes will
affect the process of molecular propagation after the release of molecules by the Tx (Fig. 2.7 (a)):

- Free diffusion is the process of the moving of a substance from a high-concentrated region in space while interacting with another substance during which its concentrations are becoming equally distributed within the area. We can see in Fig. 2.7 (b) the result of the impact of the diffusion on the movement of signaling molecules. It can be noticed that molecules diffuse (randomly move in different directions) over time, thereby the molecules are evenly distributed near to the Rx.

- Advection is the process of the bulk movement of solute substances in the direction of flow causing that movement. The properties of the substance that was taken in the starting place (Tx) are preserved during transportation to the final place (Rx). In contrast to diffusion, we can see in Fig. 2.7 (c) that under the advection, the concentration of molecules near the Rx remains the same during the movement along the channel. According to Fig. 2.7 (d), having both processes simultaneously, i.e., advection and diffusion occur in the channel, will cause the more concentrated molecules near the Rx in comparison with the diffusion effect in Fig. 2.7 (b).

- A chemical reaction is a process by which one or more starting materials (reagents) are converted into other substances in which the atomic nuclei do not change. Meanwhile, the redistribution of electrons and nuclei occurs during this process, which is the cause of new chemicals. From Fig. 2.7 (e) the result of interplaying of all three processes together can be seen. In general, reactions can be of two types: degradation and production. As a result of the degradation reaction, we have more information molecules than it was at the beginning. In contrast, we will get more molecules as a result of the reaction. The second type of the reac-
tion demonstrated in the channel in Fig. 2.7 (e). The main difference between the two cases (d) and (e) is that we will have fewer molecules near the Rx than the advection-diffusion case due to some degraded molecules.

![Diagram](image)

Figure 2.7: Main processes affecting the propagation of signaling molecules where (a) corresponds to the first time slot and (b)-(e) to the second time slot.

### 2.3.1 Diffusion model

The movement on the molecules in fluid environment is a random process which is also called Brownian motion or random walk. We can represent a Brownian motion as:

\[ x_i(t + \Delta t) = x_i(t) + \Delta x_i, \]
where $\mathbf{x}_i(t)$ is a position vector of the $i$-th molecule in 1D, 2D or 3D environment at time $t$, $\Delta \mathbf{x}_i$ is the Gaussian random variable with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{2D\Delta t}$, $D \text{ [nm}^2/\mu\text{s}]$ is the diffusion coefficient which is constant in our work, but generally, it can depend on the concentration of molecules and location $\mathbf{x}$, $\Delta t$ is the time step.

Diffusion is a random process that affects the movement of molecules in such a way that their concentration $C(\mathbf{x}, t)$ changes chaotically in time and space. This phenomenon can be described by Fick’s second law of diffusion, which expression is presented as:

$$\frac{\partial C(\mathbf{x}, t)}{\partial t} = D \Delta C(\mathbf{x}, t), \quad (2.1)$$

where $\Delta$ is a Laplacian: $\Delta C = \sum_{i=1}^{n} \frac{\partial^2 C}{\partial x_i^2}$ in the Cartesian coordinates $x_i$. The diffusion equation 2.1 has an analytical solution for simple initial (IC) and boundary conditions (BC) which will be considered next. Let’s take an unbounded environment with an impulsive point release of information molecules by the Tx and denote the solution of the partial differential equation (PDE) as $C^*(\mathbf{x}, t)$. Under these assumptions we will have the following initial and boundary conditions:

$$\text{IC} : C(\mathbf{x}_0, t \to t_0) = N\delta(\mathbf{x} - \mathbf{x}_0), \quad (2.2)$$

$$\text{BC} : C(\|\mathbf{x}\| \to \infty, t) = 0 \quad (2.3)$$

where $N$ is the number of molecules that were released by the Tx at time $t = 0$ and $\delta(\cdot)$ is the Dirac delta function. In this work, the Gaussian function with a small mean $\mu$ was implemented as an approximation of a Dirac delta function in the finite difference scheme (FDM) framework.
The solution of (2.1) under (2.2)-(2.3) can be represented as:

$$C^*(x, t) = \frac{N}{(4\pi D (t - t_0))^{\text{dim}/2}} \exp \left(- \frac{|x - x_0|^2}{4D (t - t_0)} \right),$$  \hspace{1cm} (2.4)$$

where \text{dim} is the dimension of the simulation environment, i.e., \text{dim} = 1, 2 or 3 for one-dimensional, two-dimensional or three-dimensional problems, respectively.

### 2.3.2 Advection-diffusion model

In addition to diffusion, molecules can be subjected to advection in an aqueous medium. The velocity of the fluid $v(x, t)$ should be taken into account before describing the movement by the advection. Knowing the movement of the $i$-th particle at a position $x_i$ which undergoes advection, it is possible to express its position at the time $t + \Delta t$ which results in the expression below:

$$x_i(t + \Delta t) = x_i(t) + v(x_i(t), t) \Delta t$$

where $\Delta t$ should be small enough such that the velocity vector is constant between $x_i(t)$ and $x_i(t + \Delta t)$ [21].

Given both of these phenomena, the advection-diffusion equation can be represented as:

$$\frac{\partial C(x, t)}{\partial t} = D \Delta C(x, t) - \nabla (v(x, t)C(x, t))$$ \hspace{1cm} (2.5)$$

As for advection-diffusion case, it is not possible to find an analytical solution for any velocity $v(x, t)$ which depends on space and time in most realistic models. In this work, the constant velocity $v(x, t) = v$ will be considered due to the aforementioned issue. Therefore, taking this condition into account, an analytical solution of the
35

problem (2.5) with (2.2)-(2.3) can be represented as:

\[ C^*(x, t) = \frac{N}{(4\pi D(t - t_0)^{\text{dim}/2}} \exp \left( -\frac{|x - (t - t_0)v - x_0|^2}{4D(t - t_0)} \right). \tag{2.6} \]

2.3.3 Advection-diffusion-reaction model

In this section, the last phenomenon, known as reaction, will be considered. The final PDE that will be presented below describes a process involving a combination of three processes together. An equation which includes advection, diffusion, and reaction called advection-diffusion-reaction (ADR) equation and in general, can be represented as:

\[ \frac{\partial C(x, t)}{\partial t} = D\Delta C(x, t) - \nabla (v(x, t)C(x, t)) + rf(k_f, C(x, t)) \tag{2.7} \]

where \( r \) can take values 1, if the molecule formed during the reaction (i.e. a product) or -1, if the molecule is the cause of the chemical reaction (i.e. a reactant). Let’s assume an unbounded environment, constant velocity \( v(x, t) = v \), initial and boundary conditions (2.2)-(2.3) and the first-order degradation reaction, i.e. \( r = -1 \) and \( f(k_f, C(x, t)) = k_fC(x, t) \). Therefore, the final version of the advection-diffusion-reaction equation can be presented as:

\[ \frac{\partial C(x, t)}{\partial t} = D\Delta C(x, t) - \nabla (vC(x, t)) - k_fC(x, t). \tag{2.8} \]

Given all the conditions listed above, an analytical solution of the problem (2.8) can be obtained in the form presented below:

\[ C^*(x, t) = \frac{N}{(4\pi D(t - t_0)^{\text{dim}/2}} \exp \left( -k_f(t - t_0) - \frac{|x - (t - t_0)v - x_0|^2}{4D(t - t_0)} \right). \tag{2.9} \]
2.4 Conclusion

In this chapter, the basic MC theory was presented. We saw that for three main PDEs (diffusion, advection-diffusion, and advection-reaction-diffusion), which are commonly used in this field and well-studied, we could get a closed-form solution under the simple assumptions for the diffusion coefficient, flow velocity, reaction rate function, initial and boundary conditions.
Chapter 3

Numerical methods overview

3.1 Introduction

In this chapter, the MC problem will be formulated that is going to be solved by two different numerical methods. An overview of both these methods will be presented later in this chapter. The first method is a Finite Difference Scheme (FDM), which allows finding a solution for the one-dimensional, two-dimensional and three-dimensional case problems, but the results only for one-dimensional and three-dimensional cases will be presented. A python code which contains an implementation of this framework can be found in the Appendix. The description of the second method which is a discontinuous Galerkin formulation will be presented in the second part of this chapter.

3.2 Finite difference scheme

In real life, any phenomenon may be complicated because it may be closely related to other processes, whereas interest may be confined to a single process. To be able to study the mathematical model, which includes a large number of processes, huge computational resources are required. Therefore, an important task is to highlight the most important phenomena of the process and include them in the mathematical model. The construction of an inaccurate mathematical model can affect the increase in the error of the solution which can occur in such cases as:

1. an incorrectness of the constructed mathematical model;
2. inaccuracy of initial data;

3. rounding results during calculations;

4. inappropriate selection of approximate method.

Numerical methods are powerful tools for solving complex mathematical problems. They are effective in the MC field due to the arising of the complex mathematical models which describe the real processes of this field and do not have analytical solutions most of the time. In this work, the FDM scheme was chosen because it has several advantages over other methods of solving differential equations such as:

- simplicity of an implementation;
- ease of understanding (variety of books about numerical methods);
- providing a good accuracy of the solution;
- applicability to a wide range of the physical problems.

Creating a universal numerical method is a complicated problem which has not been resolved so far. For this reason, there are a lot of methods that are suitable for a limited range of tasks. This scheme, like others, has its drawbacks, some of which we will analyze further. Firstly, increasing the accuracy of the solution leads to an increase in computational load, and, thus, requires more computational resources, which cannot be obtained most of the time. Next, to guarantee the stability of the scheme, we need to satisfy the relevant conditions, i.e., it is necessary to find the appropriate time and space step parameters, which is sometimes a laborious task. The third issue that could be encountered is that a large amount of time is needed to calculate the accurate enough solution.

In this work we have three main equations (mathematical models) which describe the physical phenomena. To be able to solve it numerically we need to discretize it
(i.e replace all available partial derivatives with the corresponding differences). The following are finite difference formulas for the discretization of a three-dimensional problem, where \( \mathbf{x} = (x, y, z) \):

\[
\Delta C (\mathbf{x}, t) = \frac{\partial^2 C}{\partial x^2} (\mathbf{x}, t) + \frac{\partial^2 C}{\partial y^2} (\mathbf{x}, t) + \frac{\partial^2 C}{\partial z^2} (\mathbf{x}, t)
\]

\[
\frac{\partial C}{\partial t} (\mathbf{x}_{i,j,k}, t_l) \approx \frac{C(\mathbf{x}_{i,j,k}, t_{l+1}) - C(\mathbf{x}_{i,j,k}, t_l)}{\Delta t} \quad (3.1)
\]

\[
\frac{\partial^2 C}{\partial x^2} (\mathbf{x}_{i,j,k}, t_l) \approx \frac{C(x_{i-1}, y_j, z_k, t_l) - 2C(x_i, y_j, z_k, t_l) + C(x_{i+1}, y_j, z_k, t_l)}{\Delta x^2} \quad (3.2)
\]

\[
\frac{\partial^2 C}{\partial y^2} (\mathbf{x}_{i,j,k}, t_l) \approx \frac{C(x_i, y_{j-1}, z_k, t_l) - 2C(x_i, y_j, z_k, t_l) + C(x_i, y_{j+1}, z_k, t_l)}{\Delta y^2} \quad (3.3)
\]

\[
\frac{\partial^2 C}{\partial z^2} (\mathbf{x}_{i,j,k}, t_l) \approx \frac{C(x_i, y_j, z_{k-1}, t_l) - 2C(x_i, y_j, z_k, t_l) + C(x_i, y_j, z_{k+1}, t_l)}{\Delta z^2} \quad (3.4)
\]

\[
\frac{\partial C}{\partial x} (\mathbf{x}_{i,j,k}, t_l) \approx \frac{C(x_{i+1}, y_j, z_k, t_l) - C(x_{i-1}, y_j, z_k, t_l)}{2\Delta x} \quad (3.5)
\]

\[
\frac{\partial C}{\partial y} (\mathbf{x}_{i,j,k}, t_l) \approx \frac{C(x_i, y_{j+1}, z_k, t_l) - C(x_i, y_{j-1}, z_k, t_l)}{2\Delta y} \quad (3.6)
\]

\[
\frac{\partial C}{\partial z} (\mathbf{x}_{i,j,k}, t_l) \approx \frac{C(x_i, y_j, z_{k+1}, t_l) - C(x_i, y_j, z_{k-1}, t_l)}{2\Delta z} \quad (3.7)
\]

where \( x \in [x_{\min}, x_{\max}], y \in [y_{\min}, y_{\max}], z \in [z_{\min}, z_{\max}], t \in [0, T], \Delta x \) and \( \Delta t \) are step
sizes for time and space, respectively. The last two parameters must meet the stability condition [22] of the FDM scheme to get a numerically correct solution. Changing the \( x_i, y_j, z_k, t_l \) parameters is presented in the expressions below:

\[
\begin{align*}
  x_i &= x_{\min} + i\Delta x, \\
  y_j &= y_{\min} + j\Delta y, \\
  z_k &= z_{\min} + k\Delta z, \\
  t_l &= t_{\text{init}} + l\Delta t,
\end{align*}
\]

where \( i \in [0, N_x - 1], j \in [0, N_y - 1], k \in [0, N_z - 1], l \in [0, N_t - 1] \), and \( N_x = \frac{|x_{\min} - x_{\max}|}{\Delta x}, N_y = \frac{|y_{\min} - y_{\max}|}{\Delta y}, N_z = \frac{|z_{\min} - z_{\max}|}{\Delta z}, N_t = \frac{T - t_{\text{init}}}{\Delta t} \). It is worth noting that one-dimensional and two-dimensional simulations can be performed using the FDM framework by applying the above formulas in the form which corresponds to the environment’s dimension.

### 3.3 Discontinuous Galerkin method

Numerically efficient algorithms that exploit \( \mathcal{O}(10^9) \) floating point operations \( 10^9 \) times per second, or exaFLOP/s, are sought on next-generation data-centric hardware [23]. The compatibility and efficiency of the hardware are the key factors that should be taken into account to determine the viability of an algorithm at an exascale. Consideration of the algorithmic robustness is no less important, and achieving it is becoming more challenging with an increase in both the size of the problem and the underlying physical complexity. Therefore, each step of the decision chain must be performed with a high level of reliability/robustness for the sake of minimization of user intervention. In computational science, the accuracy of the high-order methods allows them to be considered as the candidates for next-generation hardware, as they offer a reduction in the ratio of the communication to local computations compared to
the low order methods under the commensurate accuracy levels (for example, [24,25]).

In this work, we use a high-order entropy stable discontinuous collocation Galerkin (DG) method developed in the Entropy Stable Discontinuous Collocation Spectral Element (SSDC) solver. The DG method is based on the summation-by-parts (SBP) formulation, which allows transferring of important properties from the continuous analysis to the semi-discrete one. The SSDC solver is developed in the Advanced Algorithm and Numerical Simulations Laboratory (AANSLab) at KAUST and is built on top of the PETSc toolkit. PETSc consists of data structures and routines that can be used as tools for the implementation of large-scale application codes on parallel (and serial) computers [26]. In [27], a numerical study of the performance of the fully-discrete hp-adaptive non-linearly (entropy) stable discontinuous collocation Galerkin methods of any order with the SBP property for the compressible Navier-Stokes equations is presented. Numerical results of the selected tests were obtained that demonstrates the competitiveness of these methods for a new generation of unstructured computational fluid dynamics (CFD) tools. The Navier-Stokes equation plays an important role in the MC field. For example, it describes the bloodstream velocity profile, which complies with the laws of fluid dynamics [28]. In CFD, despite the importance of hardware compatibility and efficiency, algorithmic robustness is no less important, which is becoming increasingly difficult to achieve due to the appearance of more physically complex problems. [23] contains numerical evidence of reliability and robustness of any order non-linearly stable discontinuous spatial discretizations based on SBP and simultaneous-approximation-terms (SAT) operators for flow with under-resolved physical features or discontinuities. An example of a stable discretization of the three-dimensional convection-diffusion equation can also be found in the previously mentioned article. The explicit class of Runge-Kutta schemes was used to integrate in time the system of ordinary equations (ODEs) arising from the entropy stable spatial discretizations. In particular, the 3(2) Runge-Kutta
pair proposed by [29], with an adaptive time-stepping based on the signal processing [30], [31], is used to integrate the numerical solution in time for all the numerical results provided in the paper.

In the next subsection we give an overview of the algorithm used in the SSDC solver to discretize and solve the linear advection-diffusion equation.

### 3.3.1 The linear advection-diffusion equation

In this section, we will present the discretization of the linear advection-diffusion equation which in an \((x_1 x_2 x_3)\)-Cartesian coordinate system can be presented as

\[
\frac{\partial C (x, t)}{\partial t} + \sum_{i=1}^{3} \frac{\partial (v_i C (x, t))}{\partial x_i} = \sum_{i=1}^{3} \frac{\partial^2 (d_i C (x, t))}{\partial x_i^2}, \quad \forall x \in \Omega, \quad t \geq 0, \tag{3.9}
\]

where \((v_i C (x, t))\) are the inviscid fluxes, \(v_i\) are the constant components of the velocity vector \(v\), \(\frac{\partial (d_i C (x, t))}{\partial x_i}\) are the viscous fluxes, and \(d_i\) are the components of the diffusion vector \(D\) which are assumed to be constant and positive. The boundary \(h(x, t)\) and initial \(g(x)\) conditions are assumed to be square integrable functions.

Solving the PDE (3.9) numerically, we are working with a computation space with the reference coordinates \((x'_1, x'_2, x'_3)\) that are mapped to the physical coordinates \((x_1, x_2, x_3)\) by the push-forward transformation \((x_1, x_2, x_3) = X(x'_1, x'_2, x'_3)\), which is usually a higher-order polynomial when the curved elements are present. Thus, we need to use differentiation operators defined on a computational space to approximate our derivatives. Under the notation \(C = C (x, t)\) which will be used elsewhere for simplicity, the transformation of the (3.9) from physical to computational coordinates
\((x'_1, x'_2, x'_3)\) can be presented in the following form

\[
J_{\kappa} \frac{\partial C}{\partial t} + \sum_{j,i=1}^{3} J_{\kappa} \left( \frac{\partial x'_j}{\partial x_i} \right) \left( \frac{\partial (v_i C)}{\partial x'_j} \right) = \sum_{j,l,i=1}^{3} J_{\kappa} \left( \frac{\partial x'_j}{\partial x_i} \right) \left( \frac{\partial x'_l}{\partial x'_j} \right) \left( \frac{\partial (d_i C)}{\partial x'_l} \right),
\]

(3.10)

where \(J_{\kappa}\) is the determinant of the metric Jacobian is used along with the chain rule

\[
\frac{\partial}{\partial x_i} = \sum_{j=1}^{3} \frac{\partial x'_j}{\partial x_i} \frac{\partial}{\partial x'_j}, \quad \frac{\partial^2}{\partial x_i^2} = \sum_{j,l=1}^{3} \frac{\partial x'_j}{\partial x_i} \frac{\partial}{\partial x'_j} \left( \frac{\partial x'_l}{\partial x'_j} \right).
\]

If we bring the metric terms \(\left[ J_{\kappa} \frac{\partial x'_j}{\partial x_i} \right]\) inside the derivative in the (3.10), and use the product rule, we get

\[
J_{\kappa} \frac{\partial C}{\partial t} + \sum_{j,i=1}^{3} \frac{\partial}{\partial x'_j} \left( J_{\kappa} \frac{\partial x'_j}{\partial x_i} v_i C \right) - \sum_{j,i=1}^{3} v_i C \frac{\partial}{\partial x'_j} \left( J_{\kappa} \frac{\partial x'_j}{\partial x_i} \right) = \sum_{j,l,i=1}^{3} \frac{\partial x'_j}{\partial x_i} \frac{\partial}{\partial x'_j} \left( J_{\kappa} \frac{\partial x'_l}{\partial x'_j} \right) \left( \frac{\partial (d_i C)}{\partial x'_l} \right) - \sum_{j,l,i=1}^{3} \frac{\partial x'_l}{\partial x'_j} \frac{\partial (d_i C)}{\partial x'_j} \frac{\partial}{\partial x'_j} \left( J_{\kappa} \frac{\partial x'_l}{\partial x_i} \right).
\]

(3.11)

Taking into account the geometric conservation law (GCL) relations, we will get zero terms on the left- and right-hand sides of (3.11)

\[
\sum_{j=1}^{3} \frac{\partial}{\partial x'_j} \left( J_{\kappa} \frac{\partial x'_j}{\partial x_i} \right) = 0, \quad i = 1, 2, 3,
\]

(3.12)

which results in the strong conservation form of the advection-diffusion equation in computational space

\[
J_{\kappa} \frac{\partial C}{\partial t} + \sum_{j,i=1}^{3} \frac{\partial}{\partial x'_j} \left( J_{\kappa} \frac{\partial x'_j}{\partial x_i} v_i C \right) = \sum_{j,l,i=1}^{3} \frac{\partial}{\partial x'_j} \left( J_{\kappa} \frac{\partial x'_j}{\partial x_i} \frac{\partial (d_i C)}{\partial x'_l} \right).
\]

(3.13)

Next, let's apply the differentiation matrices to the discrete version of the convection-diffusion equation (3.13). These matrices can be presented in the following form
\[ D_{x_1'} \equiv D_{x_1'}^{(1D)} \otimes 1_{N_2} \otimes 1_{N_3}, \quad D_{x_2'} \equiv 1_{N_1} \otimes D_{x_2'}^{(1D)} \otimes 1_{N_3}, \quad D_{x_3'} \equiv 1_{N_1} \otimes 1_{N_2} \otimes D_{x_3'}^{(1D)}, \]

where \( 1_{N_j} \) is an \( N_j \times N_j \) identity matrix, \( D_{x_j'}^{(1D)} \) is a matrix SBP operator, and \( N_j \) is the number of Legendre-Gauss-Lobatto (LGL) points per direction in a given element \([27]\). Now, let’s define the diagonal matrix which contains the metric Jacobian as

\[ J_{\kappa} \equiv \text{diag} \left( J_{\kappa}(x'(1)), \ldots, J_{\kappa}(x'(N_{\kappa})) \right), \]

while the diagonal matrix of the metric terms, \( \left[ J_{\kappa} \frac{\partial x_j'}{\partial x_i} \right] \), must be chosen so that it is a discretization of

\[ \text{diag} \left( J_{\kappa} \frac{\partial x_j'}{\partial x_i}(x'(1)), \ldots, J_{\kappa} \frac{\partial x_j'}{\partial x_i}(x'(N_{\kappa})) \right), \]

where \( N_{\kappa} \equiv N_1 \cdot N_2 \cdot N_3 \) is the number of all nodes in \( \kappa \)th element, the discretization on which of (3.13) can be expressed as

\[
\frac{J_{\kappa}}{d\tau} \frac{dc_k}{dt} + \sum_{j,i=1}^{3} v_i D_{x_j'} \left[ J_{\kappa} \frac{\partial x_j'}{\partial x_i} \right]_\kappa c_k = \sum_{j,i,l=1}^{3} d_i D_{x_j'} J_{\kappa}^{-1} \left[ J_{\kappa} \frac{\partial x_j'}{\partial x_i} \right]_\kappa \left[ J_{\kappa} \frac{\partial x_l'}{\partial x_i} \right]_\kappa D_{x_l'} c_k + SAT_{\kappa},
\]

where \( SAT_{\kappa} \) is the SATs vector used for the imposition of the boundary conditions and the creation of inter-element linkages in a weak sense. \([39, 40]\). Generally, the \( SAT_{\kappa} \) vector can be decomposed into inviscid and viscous terms, i.e.

\[ SAT_{\kappa} = SAT_{\kappa}^{(I)} + SAT_{\kappa}^{(V)}. \]

Unfortunately, we cannot guarantee the stability of the scheme (3.14). However, this issue is can be addressed by a firmly-established splitting of the inviscid terms into one-half of the inviscid terms in (3.10) and one-half of the inviscid terms in (3.11).
(for those interested, an example can be found in [40]), while we should consider the viscous terms in strong conservation form. In the continuous case, this procedure results in

$$J_\kappa \frac{\partial C}{\partial t} + \frac{1}{2} \sum_{j,i=1}^{3} \left\{ \frac{\partial}{\partial x'_j} \left( J_\kappa \frac{\partial x'_j}{\partial x_i} v_i C \right) + J_\kappa \frac{\partial x'_j}{\partial x_i} \frac{\partial}{\partial \xi_l} (v_i C) \right\}$$

$$- \frac{1}{2} \sum_{j,i=1}^{3} \left\{ v_i C \frac{\partial}{\partial \xi_j} \left( J_\kappa \frac{\partial x'_j}{\partial x_i} \right) \right\} = \frac{3}{2} \sum_{j,l,i=1}^{3} \left\{ \frac{\partial}{\partial x'_j} \left( J_\kappa \frac{\partial x'_j}{\partial x_i} \frac{\partial (d_i C)}{\partial x'_l} \right) \right\},$$

where the last summation on the left-hand side equals zero by the GCL conditions (3.12). Furthermore, the construction of a stable semi-discrete form can be performed similarly to the split form (3.15), i.e. the inviscid portion of (3.10) and (3.13) can be discretized using $D_{x'_j}$, $J_\kappa$, and $\left[ J_\kappa \frac{\partial x'_j}{\partial x_i} \right]_\kappa$. After the averaging the results and adding the viscous terms stemmed from the viscous part of (3.13), we will get the following expression

$$J_\kappa \frac{dc_\kappa}{dt} + \frac{1}{2} \sum_{j,i=1}^{3} v_i \left\{ D_{x'_j} \left[ J_\kappa \frac{\partial x'_j}{\partial x_i} \right]_{\kappa} + \left[ J_\kappa \frac{\partial x'_j}{\partial x_i} \right]_{\kappa} D_{x'_j} \right\} c_\kappa$$

$$- \frac{1}{2} \sum_{j,i=1}^{3} \left\{ v_i \text{diag} (c) D_{x'_j} \left[ J_\kappa \frac{\partial x'_j}{\partial x_i} \right]_{\kappa} 1_\kappa \right\} =$$

$$\sum_{j,i,l=1}^{3} d_i D_{x'_j} J^{-1}_\kappa \left[ J_\kappa \frac{\partial x'_j}{\partial x_i} \right]_{\kappa} \left[ J_\kappa \frac{\partial x'_l}{\partial x_i} \right]_{\kappa} D_{x'_j} c_\kappa + SAT_\kappa,$$

where $1_\kappa$ is a vector of ones of size $N_\kappa$.

The semi-discrete form (3.16) emphasizes the discrete GCL relations which are analogous to the continuous GCL conditions (3.12) and can be presented as

$$\sum_{j=1}^{3} D_{x'_j} \left[ J_\kappa \frac{\partial x'_j}{\partial x_i} \right]_{\kappa} 1_\kappa = 0, \quad i = 1, 2, 3.$$

Subject to the conditions (3.17), we will get the telescoping, stable, semi-discrete form
which can be expressed as

\[
\begin{align*}
&\mathcal{J}_\kappa \frac{dc_\kappa}{dt} + \frac{1}{2} \sum_{j,i=1}^{3} v_i \left\{ D_{x'_j} \left[ J \frac{\partial x'_j}{\partial x_i} \right]_\kappa + \left[ J \frac{\partial x'_i}{\partial x_i} \right]_\kappa D_{x'_i} \right\} c_\kappa = \\
&\sum_{j,i,l=1}^{3} d_i D_{x'_j} J^{-1} \left[ J \frac{\partial x'_j}{\partial x_i} \right]_\kappa \left[ J \frac{\partial x'_l}{\partial x_i} \right]_\kappa D_{x'_l} c_\kappa + SAT_\kappa.
\end{align*}
\]

(3.18)

**Remark 1.** Arbitrary construction of order approximations of the metric terms will cause the non-preservation of the linear stability of semi-discrete operators for hyperbolic systems with the constant coefficient. Thus, the metric terms must satisfy the discrete GCL relations (3.17) to guarantee the semi-discrete forms’ stability.

However, a close analysis of (3.18) shows that it is possible to construct a spatial discretizations for conforming interfaces based on SBP-SAT operators that preserve a free-stream and allow to optimize the volume metric terms. For a review of the construction of the inviscid and viscous SATs and the optimization of the metric terms, the reader is referred to [27,41,42].

### 3.4 Conclusion

In this chapter, a description of the FDM method was given along with all the needed mathematical expressions. An overview of the fully-discrete entropy stable discontinuous collocation methods implemented in the SSDC framework. With this, necessary information on the numerical methods used was provided.
Chapter 4

Numerical results

4.1 Introduction

The next chapter will provide a reader with one-dimensional and three-dimensional numerical results that were obtained after applying the FDM scheme. Verification of the FDM method will be performed by comparing the FDM results with the SSDC solver results which, as already was mentioned above, is based on the PETSc toolkit. Error plots will be demonstrated in this chapter, and a their comparison for all major cases will be given.

4.2 One-dimensional results

In this section, we will present results for the one-dimensional problem. To be able to produce the results, we utilize both FDM and the SSDC frameworks. Further analysis will be made based on the outcome.

4.2.1 Free diffusion case

The first reviewed case is a diffusion-based MC system. This represents the simplest option of the MC system having only the presence of diffusion in the channel. For the analysis of the impact of diffusion on the communication process, the mathematical model (2.1) was taken with IC (2.2) and BC (2.3). Due to the simplicity of the above problem (2.1)-(2.3), an expression for the molecule concentration was derived analytically in (2.4) with $\text{dim}=1$. The diffusion constant characterizing the speed of
the chaotic propagation of information molecules along the channel \( D = 450 \text{ nm}^2/\mu\text{s} \) (or \( D = 4.5 \times 10^{-10} \text{ m}^2/\text{s} \)) will remain unchanged in this and all following simulations. Moreover, the time interval \( t \in \{t_{\text{init}} = 5, t_{\text{end}} = 500\} \mu\text{s} \) in all simulations will be maintained the same. Regarding the one-dimensional case, two possible scenarios for choice of the boundaries will be addressed: \( x \in \{-1700, 1700\} \) nm and \( x \in \{-3000, 3000\} \) nm. Next, other parameters which are needed for simulations of both frameworks will be listed separately. Speaking about the FDM scheme, we should mention that the convergence conditions must be met [22]. In this regard, \( \Delta x = 2 \) and \( \Delta t = 0.5 \cdot 10^{-3} \) were chosen to guarantee the convergence of the method. The location of the pointwise Tx, \( x_{\text{tx}} \), which releases \( N = 10^4 \) molecules at the beginning of time interval, is assumed to be at the origin.

![Figure 4.1: Concentration of molecules \( C^*(x,t) \) [molecules/m] versus time \( t[\mu\text{s}] \) at different locations of the receiver \( x_{\text{rx}} \) for the 1D diffusion simulation.](image)

On the above graph, three different cases are demonstrated which correspond to different Rx locations: \( x_{\text{rx}} \in \{300, 400, 500\} \) nm (or \( \{3, 4, 5\} \cdot 10^{-7} \) m). These results were obtained for boundaries \( x \in \{-1700, 1700\} \) nm, the extension of the boundaries,
in this case, will only improve the accuracy of the solution, which will be demonstrated below. The FDM solution exactly matches the advection-diffusion-reaction SSDC (ADR SSDC) and analytical solutions, which we will call AS on the graphs. It can be seen that the farther the Rx is from the Tx, the lower the concentration peak value. Moreover, the peak occurs later in time and is less pronounced. Now let’s look on the Fig. 4.2 where the errors in max norm: \[ \|E\|_\infty = \max_{1 \leq j \leq m} |E_j| = \max_{1 \leq j \leq m} |U_j - u(x_j)| \] are presented, where \( U_j \) and \( u(x_j) \) are numerical and analytical solution, respectively.

Figure 4.2: Global \( L^\infty \)-error versus time \( t \) [\( \mu s \)] for the 1D diffusion simulation.

On the above graph, two different cases for both frameworks are shown. FDM 1 and ADR SSDC 1 refers to the case when both methods started at \( t_{\text{init}} = 5 \ \mu s \) and the boundaries of the region are \( x \in \{-1700, 1700\} \ \text{nm} \), whereas FDM 2 and ADR SSDC 2 refers to the case of starting methods at \( t_{\text{init}} = 5 \ \mu s \) with increased boundaries \( x \in \{-3000, 3000\} \ \text{nm} \). We can conclude that the ADR SSDC framework is more predictable in both cases than the FDM scheme, and the error of ADR SSDC decreases with time. As for the FDM scheme, we can see that it gives a better error for increased boundaries, which related to the fact that the analytical solution
(2.4) was obtained under the unboundedness of the domain assumption 2.3. Despite the improved results of the FDM method, overall, it is less resistant to changes in parameters.

### 4.2.2 Advection-diffusion case

In this paragraph, we will consider the case of the molecular system where advection is added to the channel. Consider the problem (2.5) with initial and boundary conditions (2.2)-(2.3), which has an analytical solution (2.6) where \( \text{dim} = 1 \). The same location of the Tx as in the free diffusion case will be taken. The flow direction in this simulation and all the cases afterward is assumed to be towards the Rx. The molecular concentration \( C^*(x,t) \) is presented at the Rx location \( x_{rx} = 400 \) nm.

![Figure 4.3: Concentration of molecules \( C^*(x,t) \) [molecules/m] versus time \( t/\mu s \) with different flow velocities for the 1D advection-diffusion simulation.](image)

Fig. 4.3 shows three different cases which correspond to different flow velocities: \( v \in \{0, 2, 5\} \) nm/\( \mu s \) (or \( v \in \{0, 2, 5\} \cdot 10^{-9} \) m/\( \mu s \)). We see that the maximum values of the molecular concentration differ from each other. These differences are affected by
the change of flow velocity from \( v = 0 \) to \( v = 5 \text{ nm/\mu s} \): the molecules tend to reach the Rx faster as the speed increases. Therefore, with choosing the bigger flow velocity, the molecular concentration peak increases at the Rx. From the above figure, one can see that the FDM numerical results and analytical ones coincide perfectly.

Next we will compare the errors in max norm: 

\[ \|E\|_\infty = \max_{1 \leq j \leq m} |E_j| = \max_{1 \leq j \leq m} |U_j - u(x_j)| \]

for both frameworks: FDM and ADR SSDC.

![Figure 4.4: Global \( L^\infty \)-error versus time \( t [\mu s] \) for 1D advection-diffusion simulation under the different flow velocities and \( x \in \{-1700, 1700\} \text{ nm} \).](image)

From the above, it can be easily seen that more stable results can be obtained using the ADR SSDC method, whereas the results of the FDM scheme depend on the choice of flow speed. One of the possible explanations to this effect is that the FDM, although more precise in principle, handles the boundary conditions with worse accuracy. Let’s next expand the boundaries and analyze the performance of the FDM versus ADR SSDC method.
Figure 4.5: Global $L^\infty$-error versus time $t$ [µs] for 1D advection-diffusion simulation under the different flow velocities and $x \in \{-3000, 3000\}$ nm.

In Fig. 4.5 the results with extended boundaries are presented. It can be seen from the graph that increase in the boundaries postponed the rise of the error in the FDM.

### 4.2.3 Advection-reaction-diffusion case

Next, let’s consider the problem when information molecules can undergo degradation reaction moving along the channel. Thus, we will analyze the case, including all the main processes: diffusion, advection, and reaction. A model of this case was presented in the second chapter in PDE (2.8). The FDM scheme will be applied to this problem with initial and boundary conditions (2.2)-(2.3). We are interested in the behavior of the solution in one-dimensional space, that is, in the analytical solution (2.9) we need to replace $\text{dim} = 1$. The same Tx location $x_{tx} = 0$ will be taken as in the previous two cases. Due to the fact that we need to analyze the influence of the presence of the degradation reaction in the channel, all other parameters will be fixed, such as the
diffusion coefficient $D = 450 \text{ nm}^2/\mu s$ and the flow speed $v = 1 \text{ nm}/\mu s$. The resulting concentration of molecules $C^*(x, t)$ will be taken at the Rx location $x_{rx} = 400 \text{ nm}$ as in the previous advection-diffusion case.

![Figure 4.6: Concentration of molecules $C^*(x, t)$ [molecules/m] versus time $t/\mu s$ with different degradation rate constants for the 1D advection-reaction-diffusion simulation.](image)

In Fig. 4.6, different graphs are plotted, which correspond to three degradation rate constants: $k_f \in \{0, 1, 2\} \cdot 10^{-2} \text{ 1/}\mu s$. The correctness of the FDM method can be inferred from the graph since the analytical and FDM numerical solutions do not differ from each other. With a decrease in the degradation rate constant, we can see how the arrival time of the most concentrated formation of molecules at the Rx increases. It can be noticed that in spite of this fact molecule concentration is going up with changing rate constant from $k_f = 2 \cdot 10^{-2} \text{ 1/}\mu s$ to $k_f = 0$. The presence of a reaction, or more specifically, rising of the degradation constant, affects the decrease in the time of cleaning the channel from information molecules. This effect can be beneficial for reducing inter-symbol interference (ISI). For example, in [32], a
technique for the ISI reduction was implemented using enzymes. Moreover, a newer method was employed in [33], where the FDM was applied to a system of nonlinear PDEs in a one-dimensional environment, and it was demonstrated that the presence of a water autoionization reaction affects the reduction of the ISI.

Next, we will compare the errors of both methods in max norm: $\| E \|_\infty = \max_{1 \leq j \leq m} |E_j| = \max_{1 \leq j \leq m} |U_j - u(x_j)|$, where $U_j$ and $u(x_j)$ are numerical and analytical solution, respectively.

![Figure 4.7: Global $L^\infty$-error versus time $t$ [\mu s] for 1D advection-reaction-diffusion simulation under the different rate constants and $x \in \{-1700, 1700\}$ nm.](image)

According to the Fig. 4.7, it can be easily noticed that ADR SSDC framework gives a more accurate solution compared to the FDM scheme. One can see the cumulative effect of advection, reaction, and diffusion on the error plot. It can be noticed that adding the reaction process into simulation resulted in the divergence of error graphs for the ADR SSDC framework, whereas for the first two cases, errors for ADR SSDC were consistent among all cases considered. Based on previous results, we can assume that, in this case, the presence of chemical reactions in the channel
has a greater characteristic effect on the communication process than the physical properties of the channel (flow rate) and the distance between the elements of the communication system.

![Figure 4.8: Global $L^\infty$-error versus time $t \, [\mu s]$ for 1D advection-reaction-diffusion simulation under the different rate constants and $x \in \{-3000, 3000\}$ nm.](image)

Looking at the error graph for full simulation (advection, diffusion and reaction) with expanded boundaries, we can once again see the performance the FDM scheme improved, while the performance of ADR SSDC remains relatively similar when compared to the limited boundaries case.

**Convergence study**

In the next section, the convergence study for both methods will be presented. Let’s start with the analyzing of the FDM method results.

**FDM method** In this paragraph, we will perform an FDM mesh convergence study to demonstrate the truncation order of this method explicitly. From the expressions
(3.1)-(3.7), we can see that the central difference is used to discretize our derivatives in space and the forward Euler for discretization in time. Thus, we are dealing with an explicit Forward Time Centered Space (FTCS) scheme, which gives the first-order of accuracy in time and second-order rate of convergence in space, which will be confirmed in this paragraph. Let’s start by analyzing the FDM convergence rate table, which is presented below.

<table>
<thead>
<tr>
<th>Levels</th>
<th>$L^1$</th>
<th>Rate</th>
<th>$L^2$</th>
<th>Rate</th>
<th>$L^\infty$</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, $N=25$</td>
<td>0.7329</td>
<td>-</td>
<td>0.2462</td>
<td>-</td>
<td>0.1486</td>
<td>-</td>
</tr>
<tr>
<td>1, $N=40$</td>
<td>0.4754</td>
<td>-0.92</td>
<td>0.1252</td>
<td>-1.44</td>
<td>0.0558</td>
<td>-2.08</td>
</tr>
<tr>
<td>2, $N=50$</td>
<td>0.3775</td>
<td>-1.03</td>
<td>0.0887</td>
<td>-1.54</td>
<td>0.0351</td>
<td>-2.08</td>
</tr>
<tr>
<td>3, $N=100$</td>
<td>0.1862</td>
<td>-1.02</td>
<td>0.0310</td>
<td>-1.52</td>
<td>0.0087</td>
<td>-2.01</td>
</tr>
<tr>
<td>4, $N=200$</td>
<td>0.0923</td>
<td>-1.01</td>
<td>0.0108</td>
<td>-1.52</td>
<td>0.0021</td>
<td>-2.02</td>
</tr>
<tr>
<td>5, $N=425$</td>
<td>0.0421</td>
<td>-1.04</td>
<td>0.0034</td>
<td>-1.55</td>
<td>0.0004</td>
<td>-2.07</td>
</tr>
<tr>
<td>6, $N=850$</td>
<td>0.0186</td>
<td>-1.18</td>
<td>0.0010</td>
<td>-1.71</td>
<td>9.38E-05</td>
<td>-2.26</td>
</tr>
<tr>
<td>7, $N=1700$</td>
<td>0.0071</td>
<td>-1.38</td>
<td>0.0003</td>
<td>-1.92</td>
<td>1.90E-05</td>
<td>-2.3</td>
</tr>
</tbody>
</table>

Table 4.1: Global $L^1$, $L^2$, and $L^\infty$-errors when solving the 1D advection-reaction-diffusion equation using $N$ elements

The initial grid of the simulation was chosen to be divided into $N=25$ elements, which labeled as level 0 in the first column of the above table. A sequence of simulations with a different number of mesh elements was performed to get the convergence rates. It can be seen from the 4.1 that almost all convergence rates are gradually approaching the value ”2” with increasing in a number of elements $N$. Thus, it confirms that the FDM method gives the second-order of accuracy approximation for all selected $L^1$, $L^2$, and $L^\infty$-errors.

**ADR SSDC solver** In this paragraph, we will investigate the order of convergence of the $h/p$-adaptive approach by performing a grid convergence study of an ADR SSDC method. Let’s first analyze the convergence rates by discussing the values from the table presented below.
<table>
<thead>
<tr>
<th>Levels</th>
<th>$L^1$</th>
<th>Rate</th>
<th>$L^2$</th>
<th>Rate</th>
<th>$L^\infty$</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, $N=4$</td>
<td>7.45E-02</td>
<td>-</td>
<td>1.73E-03</td>
<td>-</td>
<td>2.02E-01</td>
<td>-</td>
</tr>
<tr>
<td>1, $N=8$</td>
<td>2.78E-02</td>
<td>-1.42</td>
<td>6.25E-04</td>
<td>-1.47</td>
<td>6.77E-02</td>
<td>-1.58</td>
</tr>
<tr>
<td>2, $N=16$</td>
<td>5.14E-03</td>
<td>-2.44</td>
<td>1.15E-04</td>
<td>-2.44</td>
<td>1.23E-02</td>
<td>-2.46</td>
</tr>
<tr>
<td>3, $N=24$</td>
<td>4.34E-04</td>
<td>-6.10</td>
<td>1.10E-05</td>
<td>-5.79</td>
<td>1.37E-03</td>
<td>-5.41</td>
</tr>
<tr>
<td>4, $N=32$</td>
<td>9.81E-05</td>
<td>-5.17</td>
<td>2.12E-06</td>
<td>-5.72</td>
<td>2.60E-04</td>
<td>-5.78</td>
</tr>
<tr>
<td>5, $N=40$</td>
<td>6.09E-05</td>
<td>-2.14</td>
<td>1.30E-06</td>
<td>-2.19</td>
<td>1.44E-04</td>
<td>-2.65</td>
</tr>
<tr>
<td>6, $N=48$</td>
<td>4.22E-05</td>
<td>-2.01</td>
<td>9.00E-07</td>
<td>-2.02</td>
<td>9.91E-05</td>
<td>-2.05</td>
</tr>
<tr>
<td>7, $N=56$</td>
<td>3.10E-05</td>
<td>-2.00</td>
<td>6.61E-07</td>
<td>-2.00</td>
<td>7.20E-05</td>
<td>-2.07</td>
</tr>
<tr>
<td>8, $N=64$</td>
<td>2.37E-05</td>
<td>-2.01</td>
<td>5.07E-07</td>
<td>-1.99</td>
<td>5.49E-05</td>
<td>-2.03</td>
</tr>
<tr>
<td>9, $N=98$</td>
<td>1.01E-05</td>
<td>-2.00</td>
<td>2.16E-07</td>
<td>-2.00</td>
<td>2.30E-05</td>
<td>-2.04</td>
</tr>
<tr>
<td>10, $N=128$</td>
<td>5.93E-06</td>
<td>-1.99</td>
<td>1.27E-07</td>
<td>-1.99</td>
<td>1.34E-05</td>
<td>-2.02</td>
</tr>
<tr>
<td>11, $N=256$</td>
<td>1.48E-06</td>
<td>-2.00</td>
<td>3.17E-08</td>
<td>-2.00</td>
<td>3.31E-06</td>
<td>-2.02</td>
</tr>
</tbody>
</table>

Table 4.2: Global $L^1$, $L^2$, and $L^\infty$-errors when solving the 1D advection-reaction-diffusion equation using $N$ elements, each of which has a local order of approximation, $p = 1$.

Table 4.2 demonstrates the global $L^1$, $L^2$, and $L^\infty$-errors at a final time $T = 500 \mu s$ and the values of the rates of convergence. The initial grid, i.e., the level 0 in the above table, is assumed to be divided into 4 elements with assigned local polynomial degree $p = 1$ of the solution in each element. A sequence of simulations with meshes, labeled as “1”, “2”,... “11” in the first column of the table, was performed. Inspecting these results, it can be seen that the ADR SSCD method is clearly convergent and the order of accuracy is confirmed to be approximately $"p + 1"$. Let’s analyze the convergence rates for the case when the local order of approximation in each element is changed to be $p = 2$. 
Table 4.3 demonstrates the global $L^1$, $L^2$, and $L^\infty$-errors at a final time $T = 500$ µs and the values of the rates of convergence. The choice of the number of grid elements will be the same as in the previous case but the local polynomial degree of the solution equals the $p = 3$ in each element. Then, the simulations with the different numbers of the mesh elements were performed, results of which labeled as “1”, “2”,... “11” in the first column of the above table. Analyzing these results, it can be seen that the ADR SSCD method is clearly convergent, and the order of accuracy is confirmed to be approximately ”$p+1$”. Let’s consider the convergence rates for the case when the local order of approximation in each element is changed to be $p = 3$.  

<table>
<thead>
<tr>
<th>Levels</th>
<th>$L^1$ Rate</th>
<th>$L^2$ Rate</th>
<th>$L^\infty$ Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, $N=4$</td>
<td>1.28E − 02</td>
<td>-</td>
<td>2.75E − 04</td>
</tr>
<tr>
<td>1, $N=8$</td>
<td>2.53E − 03</td>
<td>-2.34</td>
<td>5.43E − 05</td>
</tr>
<tr>
<td>2, $N=16$</td>
<td>1.72E − 03</td>
<td>-0.56</td>
<td>3.72E − 05</td>
</tr>
<tr>
<td>3, $N=24$</td>
<td>1.48E − 04</td>
<td>-6.05</td>
<td>3.21E − 06</td>
</tr>
<tr>
<td>4, $N=32$</td>
<td>4.77E − 06</td>
<td>-11.94</td>
<td>1.04E − 07</td>
</tr>
<tr>
<td>5, $N=40$</td>
<td>3.26E − 07</td>
<td>-12.02</td>
<td>9.04E − 09</td>
</tr>
<tr>
<td>6, $N=48$</td>
<td>1.81E − 07</td>
<td>-3.23</td>
<td>5.18E − 09</td>
</tr>
<tr>
<td>7, $N=56$</td>
<td>1.10E − 07</td>
<td>-3.23</td>
<td>3.25E − 09</td>
</tr>
<tr>
<td>8, $N=64$</td>
<td>7.22E − 08</td>
<td>-3.15</td>
<td>2.17E − 09</td>
</tr>
<tr>
<td>9, $N=98$</td>
<td>1.89E − 08</td>
<td>-3.15</td>
<td>6.02E − 10</td>
</tr>
<tr>
<td>10, $N=128$</td>
<td>8.25E − 09</td>
<td>-3.10</td>
<td>2.70E − 10</td>
</tr>
<tr>
<td>11, $N=256$</td>
<td>9.84E − 10</td>
<td>-3.07</td>
<td>3.38E − 11</td>
</tr>
</tbody>
</table>

Table 4.3: Global $L^1$, $L^2$, and $L^\infty$-errors when solving the 1D advection-reaction-diffusion equation using $N$ elements, each of which has a local order of approximation, $p = 2$.  

Table 4.3 demonstrates the global $L^1$, $L^2$, and $L^\infty$-errors when solving the 1D advection-reaction-diffusion equation using $N$ elements, each of which has a local order of approximation, $p = 2$.  

Table 4.3 demonstrates the global $L^1$, $L^2$, and $L^\infty$-errors at a final time $T = 500$ µs and the values of the rates of convergence. The choice of the number of grid elements will be the same as in the previous case but the local polynomial degree of the solution equals the $p = 3$ in each element. Then, the simulations with the different numbers of the mesh elements were performed, results of which labeled as “1”, “2”,... “11” in the first column of the above table. Analyzing these results, it can be seen that the ADR SSCD method is clearly convergent, and the order of accuracy is confirmed to be approximately ”$p+1$”. Let’s consider the convergence rates for the case when the local order of approximation in each element is changed to be $p = 3$.  

<table>
<thead>
<tr>
<th>Levels</th>
<th>$L^1$ Rate</th>
<th>$L^2$ Rate</th>
<th>$L^\infty$ Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, $N=4$</td>
<td>1.28E − 02</td>
<td>-</td>
<td>2.75E − 04</td>
</tr>
<tr>
<td>1, $N=8$</td>
<td>2.53E − 03</td>
<td>-2.34</td>
<td>5.43E − 05</td>
</tr>
<tr>
<td>2, $N=16$</td>
<td>1.72E − 03</td>
<td>-0.56</td>
<td>3.72E − 05</td>
</tr>
<tr>
<td>3, $N=24$</td>
<td>1.48E − 04</td>
<td>-6.05</td>
<td>3.21E − 06</td>
</tr>
<tr>
<td>4, $N=32$</td>
<td>4.77E − 06</td>
<td>-11.94</td>
<td>1.04E − 07</td>
</tr>
<tr>
<td>5, $N=40$</td>
<td>3.26E − 07</td>
<td>-12.02</td>
<td>9.04E − 09</td>
</tr>
<tr>
<td>6, $N=48$</td>
<td>1.81E − 07</td>
<td>-3.23</td>
<td>5.18E − 09</td>
</tr>
<tr>
<td>7, $N=56$</td>
<td>1.10E − 07</td>
<td>-3.23</td>
<td>3.25E − 09</td>
</tr>
<tr>
<td>8, $N=64$</td>
<td>7.22E − 08</td>
<td>-3.15</td>
<td>2.17E − 09</td>
</tr>
<tr>
<td>9, $N=98$</td>
<td>1.89E − 08</td>
<td>-3.15</td>
<td>6.02E − 10</td>
</tr>
<tr>
<td>10, $N=128$</td>
<td>8.25E − 09</td>
<td>-3.10</td>
<td>2.70E − 10</td>
</tr>
<tr>
<td>11, $N=256$</td>
<td>9.84E − 10</td>
<td>-3.07</td>
<td>3.38E − 11</td>
</tr>
</tbody>
</table>
Table 4.4 demonstrates the global $L^1$, $L^2$, and $L^\infty$-errors at a final time $T = 500 \mu s$ and the values of the rates of convergence. The initial grid, i.e., the level 0 in the above table, is assumed to be divided into 4 elements as in the previous cases with assigned local polynomial degree $p = 3$ of the solution in each element. A sequence of simulations with varying numbers of mesh elements, results of which correspond to the labels “1”, “2”,.. “11” in the first column of the table, was performed. Inspecting these results, it can be seen that the ADR SSCD method is clearly convergent. Moreover, it can be noted that we get consistency super-convergence of the ADR SSDC method, which means that the order of accuracy is higher than “$p + 1$”.

4.3 Three-dimensional results

In the following section, we will analyze the performance of our methods in a 3D environment. The above-mentioned results were discussed in [21]. Further, we will discuss the results of all three-dimensional simulations in the same order as was done for one-dimension cases.
4.3.1 Free diffusion case

Let’s move on to 3D numerical results and first talk about the simplest case of a molecular system, which is a diffusion-based MC system. Here, like in the case of one-dimensional system, we will analyze the model (2.1). Having the simplified initial and boundary conditions (2.2)-(2.3), an analytical solution can be obtained from (2.4) by replacing \( \text{dim} = 3 \). The boundaries of the simulation environment are set to be \( x_{\text{min}} = \{-1000, -1000, -1000\} \) nm and \( x_{\text{max}} = \{1000, 1000, 1000\} \) nm. These boundaries should be considered the same for all future 3D simulations. Moreover, the number of information molecules \( N = 10^4 \), released from the origin \( x_{\text{tx}} = (0, 0, 0) \) at the begging of the time interval remains the same as in the one-dimensional case as well as the value of the diffusion coefficient \( D = 450 \text{ nm}^2/\mu\text{s} \). Regarding the FDM scheme parameters, the same step size in all directions \( \Delta x = \Delta y = \Delta z = 20 \), and the time step \( \Delta t = 10^{-1} \) will remain unchanged throughout the simulation cases.

![Figure 4.9: Concentration of molecules \( C^*(x, t) \) [molecules/m^3] versus time [\( \mu\text{s} \)] at different Rx locations \( x_{\text{rx}} \) for the 3D diffusion simulation.](image)
The Fig. 4.9 shows three different cases of the diffusion-based MC, which vary by the receiver location: \( \mathbf{x}_{rx} = (x, 0, 0) \), where \( x \in \{300, 400, 500\} \) nm (or \( \{3, 4, 5\} \cdot 10^{-7} \) m). One can see that the process, characteristically, is similar to the one of a 1D case except that the concentration peaks are more visible. Nevertheless, looking at the FDM results starting at roughly 240 \( \mu \)s, it can be noticed how the solution begins to diverge from analytical and ADR SSDC solutions. This can be a consequence of the choice of insufficiently big boundary conditions of the 3D environment. A beneficial effect of the expansion of the boundaries was analyzed in the 1D diffusion case.

4.3.2 Advection-diffusion case

In addition to the diffusion process, an advection may be present in the channel. We can analyse the three-dimensional simulation of that process. Advection-diffusion problem is described by (2.5) which will be studied in the next section along with initial and boundary conditions (2.2)-(2.3). An analytical solution for this problem is equivalent to the one-dimensional one (2.6), except for the parameter \( \text{dim}=3 \). All numerical results will be presented at the Rx location \( \mathbf{x}_{rx} = (400, 0, 0) \) nm.
In Fig. 4.10, three options for the advection constant $\mathbf{v} = (v_1, 0, 0)$ with $v_1 = \{0, 2, 5\}$ nm/µs (or $\{0, 2, 5\} \cdot 10^{-9}$ m/µs) were chosen, which had an impact on the final numerical results like in the one-dimension advection-diffusion problem. By comparing these results with the three-dimensional diffusion case, it can be seen how concentration grows at the Rx location $\mathbf{x}_{\text{rx}} = (400, 0, 0)$ nm with increasing the flow velocity. Overall, the general behavior resembles that of a one-dimensional case with the exception of the appearance of a visible error starting from about 300 µs.

### 4.3.3 Advection-reaction-diffusion case

The last case that will be considered in this work represents an intercommunication of all significant processes, including diffusion, advection, and reaction. Analogously to the one-dimensional advection-reaction-diffusion case, the PDE (2.8) will be numerically solved in conjunction with everywhere utilized conditions (2.2)-(2.3). Along with the rest problems of this work, this problem was solved analytically (2.9) with
dim=3. The transmitter and the receiver locations are assumed to be the same as in above examined cases. In this part, the studying of the effect of a chemical reaction on the three-dimensional process will be presented; therefore, the diffusion constant and an advection parameter $v = (1, 0, 0) \text{ nm/\mu s}$ will stay unchanged.

Figure 4.11: Concentration of molecules $C^*(x, t)$ [molecules/m$^3$] versus time [\mu s] with different degradation rate constants for the 3D advection-reaction-diffusion simulation.

An impact of the degradation reaction on the performance of MC system is demonstrated on Fig. 4.11. Numerical results of three variants of the simulations are demonstrated on the graph. The degradation constants $k_f \in \{0, 1, 2\} \cdot 10^{-2} \text{ 1/\mu s}$ were chosen in the FDM and ADR SSDC methods. Better performance of the ADR SSDC framework in this case can be confirmed by the absence of an error throughout the time interval, whereas, a noticeable deviation from the analytical solution appears at about 200 \mu s in the FDM solution. Like one-dimensional case, it can be clearly noticed that the presence of a reaction affects the final result significantly. An effect of advection and diffusion can be considered negligible compared to the reaction in
that case. The causes of the decrease in the peak concentration at the receiver with a choice of a larger degradation constant are the same as in the one-dimensional case. Moreover, with increase in degradation rate constant, the tail of the molecule concentration decreases faster, which can lead to ISI reduction [21]. In MC, ISI is an important performance-degrading factor. It can be harmful for a correct detection due to the random motion of particles in the diffusive channels [15]. The ISI issue was already mentioned in one-dimensional case but it is more visible for the environment, that is closer to the real one. In [34], a three-dimensional MC simulator called MUCIN is demonstrated which was used to analyze a consecutive symbol sending under different modulation schemes by utilizing an ISI mitigation block in a similar to the real environment.

### 4.4 Conclusion

Numerical results were presented for one-dimensional and three-dimensional cases. Cross-verification was performed for the FDM and ADR SSDC methods. Error plots were analyzed for one-dimensional environment. The confirmation of the superiority of the ADR SSDC solver for this problem was provided.
Chapter 5

The results of the coupling the Navier-Stokes solver with the advection-reaction-diffusion model

5.1 Introduction

In this chapter, the numerical results of the coupling of the advection-reaction-diffusion equation with the Navier-Stokes equation will be presented.

5.2 The problem formulation

Oftentimes, one might be interested in targeted drug delivery in the human body as it is one of the most attractive and fast-growing application fields of MC. Specifically, we are interested in transporting drugs through blood vessels such as veins and arteries. Arteries are of various types: the largest artery, aorta, is coming from the left ventricle of the heart, from which large arteries branch out into arterioles, which in turn branch many times to the smallest vessels - capillaries. Further, the capillaries located in the tissues of different organs pass blood into the venules connected with veins. The diversity of capillary vessels affects the nature of the flow in each of them. In more detail, the walls of the arteries, for example, are much thicker and more elastic than the walls of the veins to cope with the high pressure of the blood coming straight from the heart. Moreover, contractions of the muscles of the veins contribute to the movement of blood towards the heart. Thus, the blood flow in most vessels will very rarely be uniform, as in the advection equation that we analyzed earlier. It is more likely, that boundary layers, unsteady flow, and turbulence will occur. Under this
condition, it is necessary to use a more complex model that will take into account the different behaviors of blood flows to describe the transportation of the drugs in the capillaries. Using the Navier-Stokes equation, we will be able to find the bloodstream velocity at the desired vessel. Having the pre-calculated blood velocity field ready, we can put it in the advection-reaction-diffusion model and run the simulation to get the final molecular concentration. In [35], a similar approach was defined where two different models were designed, such as the cardiovascular network model and the drug propagation network model, which allow the analysis of particulate drug delivery systems. By solving a Navier-Stokes equation, the blood velocity profile was analytically calculated in any artery of the chosen cardiovascular system, provided we have the information about blood flow (cardiac input) caused by the heart-pumping action.

5.3 Verification of the proposed method

In this chapter, we will verify the correctness of the new method, which we will call "Coupling of the Navier-Stokes with the Advection-Diffusion-Reaction (CNSADR)" constructed under the PETSc toolkit, which is implemented in the nondimensional form and couples an advection-reaction-diffusion model with Navier-Stokes solver. Before performing the complex simulations, we need to make sure that the new method works correctly. Thus, we will perform the code verification by comparing the numerically obtained concentration values using this new method with those obtained via the FDM method and the ADR SSDC solver under the same conditions. The results of the simulations of the advection-diffusion-reaction model in the one-dimensional and three-dimensional environment will be demonstrated.
5.3.1 One-dimensional results verification

The first step to test the new CNSADR SSDC method is applying it to the ADR model. The choice of the mentioned model can be explained by the fact that it is the most realistic of all three models considered in this work. Next, we will present the graph with the simulation results where the uniform flow is taken as a solution to the Navier-Stokes equation, which is subsequently used as an advection parameter in the ADR model.

![Graph](image)

Figure 5.1: Concentration of molecules $C^*(x,t)$ [molecules/m] versus time [µs] with different degradation rate constants for the 1D advection-reaction-diffusion simulation.

The description of the results presented in Fig. (5.1) will be omitted due to the fact that it has already been discussed in paragraph 4.2.3. It can be seen that the numerical results of the CNSADR SSDC method coincide with analytical solutions. Therefore, we can proceed to the next step of code verification, which is the three-dimensional simulation.
5.3.2 Three-dimensional results verification

In this paragraph, the performance of the CNSADR SSDC method will be tested in a three-dimensional environment for the same ADR model selected in the previous paragraph. The three-dimensional simulation results presented above have already been analyzed in paragraph 4.3.3. Thus, it will not be posted here as in the one-dimensional verification case.

![Figure 5.2: Concentration of molecules $C^*(x, t) \text{[molecules/m}^3]\text{] versus time [µs]}$ with different degradation rate constants for the 3D advection-reaction-diffusion simulation.](image)

As can be noted from Fig. (5.2), the CNSADR SSDC results are well-matched with analytical and ADR SSDC solutions for all taken degradation rate constants up to the 300 µs. One can see the presence of the minor divergence of the numerical solution from the analytical one in the absence of a reaction case by the end of the time interval, which is different from the other two cases. Thus, the obtained results confirm the developed CNSADR SSDC method’s correctness, which will be used for further simulations in this work.
5.4 Simulation in the two-dimensional pipe under the parabolic velocity profile

5.4.1 System model

In this section, the two-dimensional simulation in the rectangular domain will be performed with the presence of the parabolic velocity profile (Poiseuille flow profile) in the channel using the CNSADR SSDC solver. We will undertake a comparative analysis of two numerical results obtained with the presence of the uniform flow and the Poiseuille flow, respectively, under the same conditions. As was already discussed in 5.2, the velocity profile in vessels will not be uniform in the real human cardiovascular system. The last simulation of this work will be performed in a 2D stationary pipe with a fluid flowing along it located between two parallel plates. In the MC terms, this pipe will play the role of a channel in which molecules propagate through the blood, for example, a vein or an artery. Moreover, let’s consider the steady flow inside the stationary pipe, which causes the zero velocity at the upper and the lower boundaries. Therefore, we will have different from the uniform flow where the fluid is almost not moving near the boundaries, whereas the flow velocity is the highest in the middle of the pipe. Namely, the parabolic velocity gradient will occur where the velocity starts to increase from the top boundary to the middle of the pipe and then decreases again to zero at the bottom boundary. According to the mentioned assumptions, the Poiseuille flow will be formed in the pipe caused by the pressure in the incompressible Newtonian fluid in the laminar flow. The laminar flow condition holds almost for all blood vessels, except for large arteries where the high stochasticity of the flow occurs, and the flow behavior should be regarded as turbulent [36]. For our simulation to be more realistic, the boundaries of the domain will correspond to the real dimensions of the cephalic vein. The length of a mentioned vein is ranging from 3.5 to 8.2 cm with a mean of 4.8+/-.7 cm, while the diameter
range is from 0.1 to 1.2 cm having the mean of 0.8 +/- 0.1 cm [37]. Therefore, we will choose close to the average values of the length \( l = 6 \text{ cm} \) (or \( 6 \cdot 10^7 \text{ nm} \)) and the diameter \( d = 0.6 \text{ cm} \) (or \( 6 \cdot 10^6 \text{ nm} \)) for the next simulation.

![Figure 5.3: The MC system with the presence of the Poiseuille flow.](image)

5.4.2 Numerical results

**Advection-diffusion process**

Consider a two-dimensional advection-diffusion process presented in the rectangular channel where \( x \in \{x_{\text{min}} = 0, x_{\text{max}} = 6\} \cdot 10^7 \text{ nm} \) and \( y \in \{y_{\text{min}} = 0, y_{\text{max}} = 6\} \cdot 10^6 \text{ nm} \) bounded from the top and the bottom by two parallel plates. The Tx located at \( x_{\text{tx}} = (0, 3) \cdot 10^6 \text{ nm} \) produces instant release of \( N = 1 \cdot 10^6 \) information molecules at the beginning of the time interval \( t \in \{t_{\text{init}} = 1.5, t_{\text{end}} = 1800\} \cdot 10^8 \mu\text{s} \). Thus, we will find the molecular concentration \( C^*(x,t) \) by solving 2.5 having the Poiseuille flow profile with \( \mathbf{v}_{\text{max}} = (v_{\text{max}1}, 0) \text{ nm/\mu s} \), where \( v_{\text{max}1} = \{1, 3, 4\} \cdot 10^{-4} \text{ nm/\mu s} \) and the initial condition 2.2, homogeneous Neumann boundary conditions:

\[
\text{BC}_1 : \frac{\partial C}{\partial x} = 0 \text{ for } x = x_{\text{min}}, \text{ and } y_{\text{min}} < y < y_{\text{max}} \tag{5.1}
\]

\[
\text{BC}_2 : \frac{\partial C}{\partial x} = 0 \text{ for } x = x_{\text{max}}, \text{ and } y_{\text{min}} < y < y_{\text{max}} \tag{5.2}
\]

and the following no slip boundary conditions:
BC_3: C = 0 for \( y = y_{\text{min}} \), and \( x_{\text{min}} \leq x \leq x_{\text{max}} \) \hspace{1cm} (5.3)

BC_4: C = 0 for \( y = y_{\text{max}} \), and \( x_{\text{min}} \leq x \leq x_{\text{max}} \) \hspace{1cm} (5.4)

Along with the above assumptions, a diffusion coefficient was chosen based on the study, which contains the measurement of biomolecular diffusion coefficients in blood plasma [38]. The average value \( D = 7 \cdot 10^{-10} \text{ m}^2/\text{s} \) (or \( 7 \cdot 10^2 \text{ nm}^2/\mu\text{s} \)) will be used from the approximate range of the diffusion coefficients \( 1 \times 10^{-10} - 14 \times 10^{-10} \text{ m}^2/\text{s} \) corresponding to all measured substances.

Figure 5.4: Concentration of molecules \( C^*(x, t) \) [molecules/m\(^2\)] versus time [minutes] with different flow velocities profiles of the 2D advection-diffusion model.

In Fig. 5.4, the fourth-order of accuracy numerical solutions for both flow velocity profiles are presented. The pink lines are the numerical solutions of the advection-diffusion equation with the constant uniform flow assumption. Three black curves, in turn, correspond to the results of the advection-diffusion equation where the parabolic
velocity profile is present. From Fig. 5.4, we can observe three numerical solutions for both the Poiseuille and the uniform flow profiles, which differ by flow velocities constants \( \mathbf{v} = (v_1, 0) \) with \( v_1 = \{1, 3, 4\} \cdot 10^{-4} \text{ nm/\mu s} \) (or \( \{1, 3, 4\} \cdot 10^{-7} \text{ m/s} \)). As for the Poiseuille flow, the changing of \( \mathbf{v} \) leads to a change in the maximum velocity \( \mathbf{v}_{\text{max}} \) in the middle of the pipe. The molecular concentration \( C^*(x, t) \) molecules/m\(^2\) plotted versus time \( t \) minutes at the Rx location \( x_{\text{rx}} = (15, 3) \cdot 10^6 \text{ nm} \) (Fig. 5.3). It can be noted from the figure that the maximum values of the molecular concentrations at the Rx are higher in the case of the uniform flow than the Poiseuille flow profile. Moreover, the higher the flow velocity in both profiles, the earlier in time, the peak concentration of molecules is formed at the Rx. At the same time, the tails for the uniform flow profile decrease faster with time than for the parabolic velocity profile, which can be useful for reducing ISI in a practical system. From the last observed difference, we can assume that if we have the more realistic, i.e., a more complex character of the flow, it will affect the channel noise and is more likely to increase it. Thus, it is necessary to search for the appropriate techniques to reduce this effect, thereby weakening the ISI effect.

**Advection-reaction-diffusion process**

Chemical reactions play an important role in MC systems, as was already discussed in the previous chapters. This paragraph contains the comparison of the numerical results of simulations in which advection, reaction, and diffusion processes under the uniform and the Poiseuille flow profiles are presented. In the next two-dimensional simulation, the same parameters as in the previous advection-diffusion case for the dimensions of the rectangular domain, the Tx and Rx locations, number of released molecules, and the time interval will be chosen. Under these assumptions, we will find the numerical solution of the problem 2.8 with (5.1)-(5.4) having the Poiseuille flow profile with \( \mathbf{v}_{\text{max}} = (3, 0) \cdot 10^{-4} \text{ nm/\mu s} \), which is the solution of the Navier-Stokes
equation in the CNSADR SSDC solver. The ADR SSDC method, which was utilized for the FDM method verification, will be used to find the solution under the uniform velocity profile.

Figure 5.5: Concentration of molecules $C^*(x, t)$ [molecules/mm$^2$] versus time [minutes] with different flow velocities profiles of the 2D advection-reaction-diffusion model.

In Fig. 5.5, we can see the numerical solutions of the fourth-order of accuracy for both flow profiles which are demonstrated under the four different degradation rate constants $k_f = \{0, 0.5, 2.5, 5\} \cdot 10^{-11} \text{ 1/µs}$ (or $\{0, 0.05, 0.25, 0.5\} \cdot 10^{-4} \text{ 1/s}$). All pink lines correspond to the numerical results of the advection-reaction-diffusion problem under the uniform velocity profile, whereas the black lines relate to the solutions of the same problem under the Poiseuille flow behavior. One can notice that for the all degradation constants, the ADR SSDC uniform results reach a slightly higher molecular concentration value comparing to the CNSADR Poiseuille (P.) flow results. Moreover, one can see that the tails of the results obtained under the uniform flow profile are decreasing faster comparing with the CNSADR P. flow results. It is worth
noting, with an increase in the degradation rate constant, the difference in the decrease of the results tails becomes less noticeable.

5.5 Conclusion

In this chapter, the effects of the parabolic and uniform velocity profiles on both the advection-diffusion and the advection-reaction-diffusion model were discussed. Moreover, we gave a comparative analysis of the obtained results.
Chapter 6

Conclusion

Throughout this thesis, the recently emerging field of molecular communications, the development of which can bring a potential influence to such MC application areas like medicine, healthcare, environment, industry, ICT, has been investigated. The theoretical basis was presented, necessary for understanding this area for those currently in the early stages of studying MC. Along with the theoretical side, the implementation aspects of the main mathematical models were considered. Moreover, the numerical FDM scheme, which deals with the simplest one, two, three-dimensional MC mathematical models, was developed, and the verification of it was performed. During the method verification, the FDM results were compared with the analytical solutions and results obtained by the SSDC solver developed in the AANSLab at KAUST. According to the error plots for the one-dimensional simulation, it was found that the SSDC framework is more accurate and stable than the FDM scheme. Next, by using the FDM and the SSDC solver, the effect of a combination of physical processes as diffusion, advection, and reaction on the molecular communication was analyzed. It is worth noting that analytical solutions for the models were obtained under such simplified system properties as a constant diffusion coefficient, uniform flow, unbounded environment. In more realistic applications, the flow velocity profile is rarely uniform due to the influence of the boundary layers, unsteady flow, or turbulence, which justifies the need to use a more complex model. Thus, in this work, we also consider the one-way coupling between the Navier-Stokes and the advection-reaction-diffusion models. After verifying the coupled model, a comparative analysis of two simulations
in a two-dimensional pipe subject to both uniform and Poiseuille flow profiles was implemented. It was noted that a more realistic Poiseuille flow profile would likely cause increased symbol interference when compared to a wide-spread uniform flow profile. Therefore, it is essential for the future of the MC communications to search for appropriate techniques to reduce the ISI effect. In the very near future, we aim to perform simulations in a complex three-dimensional realistic environment starting from the magnetic resonance imaging (MRI) scan of a human vein. The results will allow us to analyze the effects of the channel’s physical properties on the whole communication process.

Finally, let’s summarize the contributions that were achieved during this work:

- Unified framework for the verification and validation of the analytical solutions proposed so far in the MC field.

- Numerical study of the effect of the boundary condition for the pure advection-diffusion-reaction equation.


- Study of the effect of a laminar Navier-Stokes background flow on the propagation of molecules in the two-dimensional pipe channel where advection, reaction, and diffusion processes are present.
import numpy as np
import math as math
from adv_reac_diff_1D_main import adv_reac_diff_1D
from adv_reac_diff_2D_main import adv_reac_diff_2D
from adv_reac_diff_3D_main import adv_reac_diff_3D
import os
def clear(): return os.system('cls')  # On Windows System
clear()

# class Parameters for the 1D problem

class cl_params_1D:
    def __init__(self, C0, C, C_an, Nx, Nt, D, v, k_f, N, t_zero, x_zero, Rx_loc, x, t, t_init, dt, dx, dim):
        self.C0 = C0
        self.C = C
        self.C_an = C_an
        self.Nx = Nx
        self.Nt = Nt
        self.D = D
        self.v = v
        self.k_f = k_f
        self.N = N
        self.t_zero = t_zero
        self.x_zero = x_zero
        self.Rx_loc = Rx_loc
        self.x = x
        self.t = t
self.t_init = t_init
self.dt = dt
self.dx = dx
self.dim = dim

# class Parameters for the 2D problem
class cl_params_2D:
    def __init__(self, C0, C, C_an, Nx, Ny, Nt, D, v, k_f, N, t_zero, x_zero, Rx_loc, x, y, t, t_init, dt, dx, dy, dim):
        self.C0 = C0
        self.C = C
        self.C_an = C_an
        self.Nx = Nx
        self.Ny = Ny
        self.Nt = Nt
        self.D = D
        self.v = v
        self.k_f = k_f
        self.N = N
        self.t_zero = t_zero
        self.x_zero = x_zero
        self.Rx_loc = Rx_loc
        self.x = x
        self.y = y
        self.t = t
        self.t_init = t_init
        self.dt = dt
        self.dx = dx
        self.dy = dy
        self.dim = dim

# class Parameters for the 3D problem
class cl_params_3D:
def __init__(self, C0, C, C_an, Nx, Ny, Nz, Nt, D, v, k_f, N, 
    t_zero, x_zero, Rx_loc, x, y, z, t, t_init, dt, dx, dy, dz, dim):
    self.C0 = C0
    self.C = C
    self.C_an = C_an
    self.Nx = Nx
    self.Ny = Ny
    self.Nz = Nz
    self.Nt = Nt
    self.D = D
    self.v = v
    self.k_f = k_f
    self.N = N
    self.t_zero = t_zero
    self.x_zero = x_zero
    self.Rx_loc = Rx_loc
    self.x = x
    self.y = y
    self.z = z
    self.t = t
    self.t_init = t_init
    self.dt = dt
    self.dx = dx
    self.dy = dy
    self.dz = dz
    self.dim = dim

# enter the dimension of the environment in which the simulation 
# will be performed (two options: one-dimensional, two-dimensional 
# or three-dimensional)

dim = int(input(f'Please enter the dimension of the environment in 
    which the simulation will be performed (1, 2 or 3): '))

# the number of transmitted molecules N=10^4 by default
N = 10**4

# the diffusion coefficient equals D=450 by default
D = 450

# the flow velocity equals v1=0 by default
v1 = 0

# the forward rate constant equals k_f=0 by default
k_f = 0

# the released time of information molecules from the transmitter
t_zero = 0

# the initial simulation time equals t_init=5 by default
t_init = 5

# the final time equals T_end=500 by default
T_end = 500

if dim == 1:
    # the advection velocity v=v1, where v1=0 by default
    v = v1

    # the receiver location equals Rx_loc=400 by default
    Rx_loc = 400

    # the transmitter location equals x_zero=0 by default
    x_zero = 0

    # the initial coordinate of the one-dimensional interval x_min
    # =-1700 by default
    x_min = -1700

    # the end coordinate of the one-dimensional interval x_max=1700
    # by default
    x_max = 1700

    # the step size in the x direction dx=1 by default
    dx = 2

    # the number of grid points per side
    Nx = int((x_max-x_min)/dx)

    # the grid in space
    x = np.linspace(x_min, x_max, Nx+1)
# the time step
dt = 0.5*10**(-3)

# the number of iterations in time
Nt = int(np.ceil((T_end-t_init)/dt))

# the temporal vector used for initial condition
C0 = np.zeros(Nx+1)

# an analytical solution
C_an = np.zeros((Nx+1, Nt+1))

# the numerical solution
C = np.empty(Nx+1)

# the grid in time
t = np.linspace(t_init, T_end, Nt+1)

# creating a class with Parameters
p = cl_params_1D(C0, C, C_an, Nx, Nt, D, v, k_f, N, t_zero, x_zero, Rx_loc, x, t, t_init, dt, dx, dim)

# run the simulation with all needed parameters
adv_reac_diff_1D(p)

elif dim == 2:

    # the advection velocity v=[v1,0], where v1=0 by default
    v = np.array([v1, 0])

    # the receiver location equals Rx_loc=[400,0] by default
    Rx_loc = [400, 0]

    # the minimum boundaries of the 2D simulation environment x_min
    #=[-1300, -1300] by default
    x_min = [-1300, -1300]

    # the maximum boundaries of the 2D simulation environment x_max
    #=[1300, 1300] by default
    x_max = [1300, 1300]

    # the transmitter location equals x_zero=[0,0] by default
    x_zero = np.array([0, 0])

    # step sizes in all directions of the 2D simulation environment dx = dy = 20
# time step
dt = 0.2 * math.pow(10, -2)

# the number of grid points per side in all directions of the 3D simulation environment
Nx, Ny = int((x_max[0]-x_min[0])/dx), int((x_max[1]-x_min[1])/dy)

# the grid in x direction
x = np.linspace(x_min[0], x_max[0], Nx+1)

# the grid in y direction
y = np.linspace(x_min[1], x_max[1], Ny+1)

# an empty 2D coordinate
coordinate = []

# the number of iterations in time
Nt = int(np.ceil(np.abs(T_end-t_init)/dt))

# an analytical solution
C_an = np.zeros((Nx+1, Ny+1, Nt+1))

# the temporal vector used for initial condition
C0 = np.zeros((Nx+1, Ny+1))

# the numerical solution
C = np.empty((Nx+1, Ny+1))

# the grid in time
t = np.linspace(t_init, T_end, Nt+1)

# creating a class with Parameters
p = cl_params_2D(C0, C, C_an, Nx, Ny, Nt, D, v, k_f, N, t_zero, x_zero, Rx_loc, x, y, t, t_init, dt, dx, dy, dim)

# run the simulation with all needed parameters
adv_reac_diff_2D(p)

eif dim == 3:
    # the advection velocity v=[v1,0,0], where v1=0 by default
    v = np.array([v1, 0, 0])

    # the receiver location equals Rx_loc=[400,0,0] by default
Rx_loc = [400, 0, 0]

# the minimum boundaries of the 3D simulation environment x_min
=x_min = [-1000, -1000, -1000]

# the maximum boundaries of the 3D simulation environment x_max
=x_max = [1000, 1000, 1000]

# the transmitter location equals x_zero=[0,0,0] by default
x_zero = np.array([0, 0, 0])

# step sizes in all directions of the 3D simulation environment
dx = dy = dz = 20

# time step
dt = 1*math.pow(10, -1)

# the number of grid points per side in all directions of the 3D
simulation environment
Nx, Ny, Nz = int(((x_max[0]-x_min[0])/dx), int(((x_max[1]-x_min[1])/dy), int(((x_max[2]-x_min[2])/dz)

# the grid in x direction
x = np.linspace(x_min[0], x_max[0], Nx+1)

# the grid in y direction
y = np.linspace(x_min[1], x_max[1], Ny+1)

# the grid in z direction
z = np.linspace(x_min[2], x_max[2], Nz+1)

# an empty 3D coordinate
coordinate = []

# the number of iterations in time
Nt = int(np.ceil(np.abs(T_end-t_init)/dt))

# an analytical solution
C_an = np.zeros((Nx+1, Ny+1, Nz+1))

# the temporal vector used for initial condition
C0 = np.zeros((Nx+1, Ny+1, Nz+1))

# the numerical solution
C = np.empty((Nx+1, Ny+1, Nz+1))
# the grid in time
t = np.linspace(t_init, T_end, Nt+1)

# creating a class with Parameters
p = cl_params_3D(C0, C, C_an, Nx, Ny, Nz, Nt, D, v, k_f, N,
t_zero,
                 x_zero, Rx_loc, x, y, z, t, t_init, dt, dx, dy,
dz, dim)

# run the simulation with all needed parameters
adv_reac_diff_3D(p)

else:
    print("There is no other option. Please enter 1, 2 or 3.")

import numpy as np
import math as math

# function for calculating initial condition and analytical solution
def an_sol(p, x, t):
    norm2 = np.linalg.norm(x-p.v*(t-p.t_zero)-p.x_zero)**2
    return C_analytical

import numpy as np
from time_step_1D import time_step_1D
from analytical_solution import an_sol
import os
clear = lambda: os.system('cls')  # On Windows System
clear()

def adv_reac_diff_1D(p):
    print("Please wait the results.")
    # error
    Error = np.zeros(p.Nt+1)
# file for saving the results
f = open("result_adv_reac_diff_1D.txt","w+")

# initial condition
for j in range(0, p.Nx+1):
    coordinate = p.x[j]
    p.C0[j] = an_sol(p, coordinate, p.t_init)

# index of the receiver on the grid
Rx_ind_x = int(np.where(p.x == p.Rx_loc)[0])

# an analytical solution at initial time
p.C_an[:,0] = p.C0[:]

# updating the error at the initial time step
Error[0] = np.linalg.norm(p.C_an[:,0]-p.C0[:], np.inf)

# print the results if needed
print(str(float(p.t_init)) + " " + str(float(p.C0[Rx_ind_x])) + " " + str(float(p.C_an[Rx_ind_x, 0])) + " " + str(float(Error[0])) + "\n")

# saving results to a file
f.write(str(float(p.t_init)) + " " + str(float(p.C0[Rx_ind_x])) + " " + str(float(p.C_an[Rx_ind_x, 0])) + " " + str(float(Error[0])) + "\n")

# time loop
for m in range(1, p.Nt+1):
    # updating the results
    p.C0, p.C, p.C_an = time_step_1D(p, m)

    # updating the error
    Error[m] = np.linalg.norm(p.C_an[:,m]-p.C0[:], np.inf)

    # print the results
    print(str(float(p.t[m])) + " " + str(float(p.C0[Rx_ind_x])) + " " + str(float(p.C_an[Rx_ind_x, m])) + " " + str(float(Error[m])) + "\n")

    # saving results to a file
    f.write(str(float(p.t[m])) + " " + str(float(p.C0[Rx_ind_x])) + " " + str(float(p.C_an[Rx_ind_x, m])) + " " + str(float(Error[m])) + "\n")
import numpy as np
import math as math
from analytical_solution import an_sol
from time_step_2D import time_step
import os
def clear():
    return os.system('cls')  # On Windows System
clear()

def adv_reac_diff_2D(p):
    print("Please wait the results.")
    # error
    Error = np.zeros(p.Nt + 1)
    # file for saving the results
    f = open("result_adv_reac_diff_2D.txt", "w+")
    # index of the receiver on the grid
    Rx_ind = [int(np.where(p.x == p.Rx_loc[0])[0]), int(np.where(p.y == p.Rx_loc[1])[0])]
    # loop in space in x direction
    for i in range(0, p.Nx + 1):
        # loop in space in y direction
        for j in range(0, p.Ny + 1):
            # creating a 2D coordinate
            coordinate = np.array([p.x[i], p.y[j]])
            # initial condition
            p.C0[i, j] = an_sol(p, coordinate, p.t_init)
            # an analytical solution at initial time
            p.C_an[:, :, 0] = p.C0[:, :]
            # updating the error at initial time step
            Error[0] = np.linalg.norm(p.C_an[:, :, 0] - p.C0[:, :])
            # print the results
```python
print(str(float(p.t_init)) + " " + str(float(p.C0[Rx_ind[0], Rx_ind[1]])) + " " + str(float(p.C_an[Rx_ind[0], Rx_ind[1], 0])) + " " + str(float(Error[0])) + "\n")
# saving the results in file
f.write(str(float(p.t_init)) + " " + str(float(p.C0[Rx_ind[0], Rx_ind[1]])) + " " + str(float(p.C_an[Rx_ind[0], Rx_ind[1], 0])) + " " + str(float(Error[0])) + "\n")
# time loop
for m in range(1, p.Nt+1):
    # updating the results
    p.C0, p.C, p.C_an = time_step(p, m)
    # updating the error
    Error[m] = np.linalg.norm(p.C_an[:,:,:]-p.C0[:,:, :])
    # print the results
    print(str(float(p.t[m])) + " " + str(float(p.C0[Rx_ind[0], Rx_ind[1]])) + " " + str(float(p.C_an[Rx_ind[0], Rx_ind[1], m])) + " " + str(float(Error[m])) + "\n")
    # saving the results in file
    f.write(str(float(p.t[m])) + " " + str(float(p.C0[Rx_ind[0], Rx_ind[1]])) + " " + str(float(p.C_an[Rx_ind[0], Rx_ind[1], m])) + " " + str(float(Error[m])) + "\n")
f.close()
```

```python
import numpy as np
import math as math
from analytical_solution import an_sol
from time_step_3D import time_step
import os
clear = lambda: os.system('cls') # On Windows System
clear()

def adv_reac_diff_3D(p):
    print("Please wait the results.")
```
# error
Error = np.zeros(p.Nt+1)

# file for saving the results
f = open("result_adv_reac_diff_3D.txt","w+")

# index of the receiver on the grid
Rx_ind = [int(np.where(p.x == p.Rx_loc[0])[0]), int(np.where(p.y == p.Rx_loc[1])[0]), int(np.where(p.z == p.Rx_loc[2])[0])]

# loop in space in x direction
for i in range(0, p.Nx+1):
    # loop in space in y direction
    for j in range(0, p.Ny+1):
        # loop in space in z direction
        for k in range(0, p.Nz+1):
            # creating a 3D coordinate
            coordinate = np.array([p.x[i], p.y[j], p.z[k]])
            # initial condition
            p.C0[i, j, k] = an_sol(p, coordinate, p.t_init)

    p.C_an[:, :, :, 0] = p.C0[:, :, :]

    # updating the error at initial time step
    Error[0] = np.linalg.norm(p.C_an[:, :, :, 0] - p.C0[:, :, :])

    # print the results
    print(str(float(p.t_init)) + " " + str(float(p.C0[Rx_ind[0], Rx_ind[1], Rx_ind[2]])) + " " + str(float(p.C_an[Rx_ind[0], Rx_ind[1], Rx_ind[2], 0])) + " " + str(float(Error[0])) + "\n")

    # saving the results in file
    f.write(str(float(p.t_init)) + " " + str(float(p.C0[Rx_ind[0], Rx_ind[1], Rx_ind[2]])) + " " + str(float(p.C_an[Rx_ind[0], Rx_ind[1], Rx_ind[2], 0])) + " " + str(float(Error[0])) + "\n")

# time loop
for m in range(1, p.Nt+1):
    # updating the results
    p.C0, p.C, p.C_an = time_step(p, m)
# error the error
Error[m] = np.linalg.norm(p.C_an[:, :, :, m]-p.C0[:, :, :])

# print the results
print(str(float(p.t[m])) + " " + str(float(p.C0[Rx_ind[0],
Rx_ind[1], Rx_ind[2]])) + " " + str(float(p.C_an[Rx_ind[0],
Rx_ind[1], Rx_ind[2], m])) + " " + str(float(Error[m])) + "\n")

# saving the results in file
f.write(str(float(p.t[m])) + " " + str(float(p.C0[Rx_ind[0],
Rx_ind[1], Rx_ind[2]])) + " " + str(float(p.C_an[Rx_ind[0],
Rx_ind[1], Rx_ind[2], m])) + " " + str(float(Error[m])) + "\n")
f.close()

from analytical_solution import an_sol
import numpy as np

# function for calculating numerical and analytical solution at each
time step
def time_step_1D(p, m):
    # loop in x direction
    for i in range(p.Nx+1):
        if (i == 0 or i == p.Nx):
            p.C[i] = 0
        else:
+ p.C0[i-1])/(p.dx*p.dx)) - p.dt * (p.v*(p.C0[i+1] - p.C0[i-1])
/(2*p.dx)) - p.dt*p.k_f*p.C0[i]
            coordinate = p.x[i]
        p.C_an[i,m] = an_sol(p, coordinate, p.t[m])
    p.C0 = p.C.copy()
    return p.C0, p.C, p.C_an

from analytical_solution import an_sol
import numpy as np
# function for calculating numerical and analytical solution at each time step

def time_step(p, m):
    dx2 = p.dx*p.dx
    dy2 = p.dy*p.dy
    # loop in space in x direction
    for i in range(p.Nx+1):
        # loop in space in y direction
        for j in range(p.Ny+1):
            if (i == 0 or i == p.Nx) or (j == 0 or j == p.Ny):
                p.C[i, j] = 0
            else:
                coordinate = np.array([p.x[i], p.y[j]])
                p.C_an[i, j, m] = an_sol(p, coordinate, p.t[m])
    p.C0 = p.C.copy()
    return p.C0, p.C, p.C_an

from analytical_solution import an_sol
import numpy as np

# function for calculating numerical and analytical solution at each time step

def time_step(p, m):
    dx2 = p.dx*p.dx
    dy2 = p.dy*p.dy
    dz2 = p.dz*p.dz
    # loop in space in x direction
for i in range(p.Nx+1):
    # loop in space in y direction
    for j in range(p.Ny+1):
        # loop in space in z direction
        for k in range(p.Nz+1):
            if (i == 0 or i == p.Nx) or (j == 0 or j == p.Ny) or (k == 0 or k == p.Nz):
                p.C[i, j, k] = 0
            else:
                coordinate = np.array([p.x[i], p.y[j], p.z[k]])
                p.C_an[i, j, k, m] = an_sol(p, coordinate, p.t[m])
        p.C0 = p.C.copy()
return p.C0, p.C, p.C_an
REFERENCES


