Multiscale Modeling in Systems Biology

Methods and Perspectives

ADRIEN COULIER
In the last decades, mathematical and computational models have become ubiquitous to the field of systems biology. Specifically, the multiscale nature of biological processes makes the design and simulation of such models challenging. In this thesis we offer a perspective on available methods to study and simulate such models and how they can be combined to handle biological processes evolving at different scales.

The contribution of this thesis is threefold. First, we introduce Orchestral, a multiscale modular framework to simulate multicellular models. By decoupling intracellular chemical kinetics, cell-cell signaling, and cellular mechanics by means of operator-splitting, it is able to combine existing software into one massively parallel simulation. Its modular structure makes it easy to replace its components, e.g. to adjust the level of modeling details. We demonstrate the scalability of our framework on both high performance clusters and in a cloud environment.

We then explore how center-based models can be used to study cellular mechanics in biological tissues. We show how modeling and numerical choices can affect the results of the simulation and mislead modelers into incorrect biological conclusions if these errors are not monitored properly. We then propose CBMOS, a Python framework specifically designed for the numerical study of such models.

Finally, we study how spatial details in intracellular chemical kinetics can be efficiently approximated in a multiscale compartment-based model. We evaluate how this model compares to two other alternatives in terms of accuracy and computational cost. We then propose a computational pipeline to study and compare such models in the context of Bayesian parameter inference and illustrate its usage in three case studies.

Keywords: computational systems biology, high performance computing, Bayesian inference, stochastic simulation, cell mechanics
To my family, and all my friends.
List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.


III Sonja Mathias, Adrien Coulier, Andreas Hellander. CBMOS: a GPU-enabled Python framework for the numerical study of center-based models. *bioRxiv 2021.05.06.442893*, 2021. (Submitted)


Reprints were made with permission from the publishers.
## Contents

1 Introduction .................................................................................................................. 11

2 Intracellular Chemical Dynamics ................................................................................. 15
   2.1 The Chemical Master Equation ........................................................................... 15
   2.2 Simulation and Sampling Methods ....................................................................... 16
   2.3 Reaction-Diffusion Models .................................................................................. 18
      2.3.1 Lattice Methods ............................................................................................ 20
      2.3.2 Off-lattice Methods ...................................................................................... 21
   2.4 Challenges in Modeling Spatial Stochastic Chemical Kinetics .............................. 22

3 Agent-Based Modeling for Multicellular Systems ......................................................... 25
   3.1 On-Lattice Models ............................................................................................... 26
      3.1.1 Cellular Automata ......................................................................................... 26
      3.1.2 Cellular Ports Models ................................................................................... 27
   3.2 Off-Lattice Models ............................................................................................... 28
      3.2.1 Center-Based Models .................................................................................... 28
      3.2.2 Deformable Cell Models .............................................................................. 30
   3.3 Outlook .................................................................................................................. 30

4 Multiscale Modelling of Multicellular Tissues ............................................................... 33
   4.1 Multiscale Simulations with Orchestral ................................................................. 33
   4.2 CBMOS, a Python Framework for the Numerical Analysis of Center-Based Models .................................................................................................................... 37
   4.3 Multiscale Modeling, Model Comparison and Parameter Inference for Stochastic Chemical Kinetics ................................................................. 38
   4.4 Outlook .................................................................................................................. 42

5 Conclusion .................................................................................................................... 43

Sammanfattning på svenska ............................................................................................. 45

Author’s Contributions ..................................................................................................... 47

Acknowledgments ............................................................................................................. 49

References ....................................................................................................................... 51
List of abbreviations

The following table summarizes all the abbreviations found in this thesis, their meaning, as well as the page where they first appear.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>Approximate Bayesian Computation</td>
<td>39</td>
</tr>
<tr>
<td>ABM</td>
<td>Agent-Based Model</td>
<td>25</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
<td>37</td>
</tr>
<tr>
<td>CA</td>
<td>Cellular Automaton</td>
<td>26</td>
</tr>
<tr>
<td>CME</td>
<td>Chemical Master Equation</td>
<td>16</td>
</tr>
<tr>
<td>CP</td>
<td>Cellular Potts</td>
<td>27</td>
</tr>
<tr>
<td>CTMC</td>
<td>Continuous Time Markov Chain</td>
<td>15</td>
</tr>
<tr>
<td>DTPM</td>
<td>Discrete Time Particle Method</td>
<td>21</td>
</tr>
<tr>
<td>FSP</td>
<td>Finite State Projection</td>
<td>16</td>
</tr>
<tr>
<td>GFRD</td>
<td>Green’s Function Reaction Dynamics</td>
<td>21</td>
</tr>
<tr>
<td>GPGPU</td>
<td>General Purpose Graphics Processing Unit</td>
<td>37</td>
</tr>
<tr>
<td>ODE</td>
<td>Ordinary Differential Equation</td>
<td>12</td>
</tr>
<tr>
<td>PDE</td>
<td>Partial Differential Equation</td>
<td>12</td>
</tr>
<tr>
<td>RDME</td>
<td>Reaction-Diffusion Master Equation</td>
<td>20</td>
</tr>
<tr>
<td>SBML</td>
<td>Systems Biology Markup Language</td>
<td>22</td>
</tr>
<tr>
<td>SSA</td>
<td>Stochastic Simulation Algorithm</td>
<td>16</td>
</tr>
</tbody>
</table>
1. Introduction

Science may be described as the art of systematic over-simplification — the art of discerning what we may with advantage omit.

Karl Popper, 1992

Mathematical models are important tools in biology, making it possible to formalize the current knowledge of a system, compare it to experimental observations, predict the outcome of future experiments and test new hypotheses [139]. Ideally, mathematical models should not only be able to answer yes or no questions, but also to be able to quantify effects, uncertainties and temporal variations. The field of systems biology focuses on the interplay between biological components at different scales and can be used to address questions about biological processes driving the growth of an embryo, or the development of a cancerous tumor [129]. With the continuous increase in capacity, computers act as a catalyst to biological models and are paramount to systems biology, making it possible, to increase our understanding of biological systems through simulations and data analysis [116].

Computational systems biology is fundamentally multiscale, with microscale interactions between proteins and genes inside single cells influencing the shape of tissues and organs at the macroscale, and vice versa. For instance, in most animal species, the proper segmentation of the body during embryogenesis is believed to be driven by synchronized biochemical clocks inside individual cells [108]. Understanding the interplay between these scales is crucial to answer questions about developmental biology. Furthermore, stochastic noise emerges at every scale, either as undesirable random variations, or as a process essential to the correct functioning of biological organisms [132]. As an example, noise has been shown to be critical for robust, sustained oscillations in circadian clocks [136]. The accumulated effects of both multiscale interactions and stochastic fluctuations makes quantitative modeling one of the current major challenges in computational systems biology [104].

A large number of models have been developed at different scale, at various degrees of complexity, and each with their own set of advantages and
drawbacks. At the intracellular level, the number of interacting molecules will usually dictate what type of model should be used. When the system involves *sufficiently many* molecules, the law of large numbers will gradually make stochastic variations negligible. In that case, deterministic models, based on Ordinary Differential Equations (ODE), may be a good approach. On the other hand, when the number of interacting molecules is small, for instance when the modeler is interested in the interactions between genes and transcription factors, stochastic variations will have a fundamental impact on the system’s dynamics, calling for discrete stochastic models [64]. Likewise, the speed of diffusion — how fast molecules randomly spread through the cell — may guide the choice of model. When diffusion is fast, molecules will tend to be uniformly distributed throughout the cell. On the other hand, when diffusion is slow, molecules will tend to cluster in some areas of the cell. In that case models including spatial details, for instance based on Partial Differential Equations (PDE) or spatial stochastic models, may be needed [24].

At the tissue scale, available methods can be classified into roughly two categories. Continuum population models, where reaction-diffusion equations are used to model the chemicals produced and consumed by the cells, and discrete individual based models, where cells are modeled as distinct agents interacting with their environment and with one another. Although this thesis will mostly focus on discrete models, it is possible to combine the two approaches in hybrid models [7]. These models are particularly useful to study the dynamics of tissues in morphogenesis. Most notably, both continuous models and cell-based models have proven very successful when modeling cancer growth [29, 97]. When it comes to discrete models, there is a wide range of methods available. However, contrary to the intracellular scale, only few guidelines are available to modelers when it comes to choosing which path to take. Concretely, this means that in practice modelers will rely mostly on experience.

Bringing the intracellular and the tissue scale together is challenging. Yet, in some cases this might be critical to reveal the true mechanisms behind biological phenomena such as cancer or embryogenesis. In particular, simulating such models together is technically complex, and it is notably difficult to determine which models are adequate for the problem at hands. In a recent preprint [54], Fletcher and Osborne described seven current challenges of multiscale multicellular modeling of tissues, going from the conceptual level, such as how to build these models, down to the technical level, e.g. how to implement and simulate these models on high performance computing architectures. The aim of this thesis is to address these challenges. In Paper I, we develop a multiscale framework to bring together existing specialized simulation software into one distributed program. In Paper II and III, we study a specific class of discrete cell-based model, de-
termine how modeling choices influence the results of these models, and implement a high performance Python framework specifically designed for their numerical study. In Paper IV, we propose a multiscale model capturing some spatial aspects of the cell and compare it to two other modeling approaches. Finally, in Paper V, we implement a computational pipeline to compare these three approaches in the context of Bayesian parameter inference.

This thesis is organized as follows: in Chapter 2 we review current methods for simulating intracellular chemical kinetics at different scales. Chapter 3 is an introduction to agent based modeling for cellular mechanics. Finally in Chapter 4 we detail the contributions of each paper presented in this thesis and how they contribute to solving the challenges presented in [54].
2. Intracellular Chemical Dynamics

In this chapter, we give an introduction on how stochastic chemical kinetics at the intracellular level can be modeled at various levels of fidelity. These models are used to study the interactions between the different components inside the cell, such as proteins and genes. We leave the traditional ordinary differential equations aside and focus mainly on discrete stochastic models. In this formalism, it is the molecule counts for each species that are of interest, rather than their concentrations. In Section 2.1, we present the equation underlying well-mixed models. Section 2.2 is dedicated to practical methods to approximate the solution of this equation, and Section 2.3 describes several approaches to study spatially inhomogeneous systems.

2.1 The Chemical Master Equation

Let us consider a system made of $n$ chemical species and let $\mathbf{x} \in \mathbb{N}^n$ be the molecular count for each species. Let $\mathbf{X}(t)$ be the stochastic process representing the temporal evolution of this molecular count. Let $p(\mathbf{x}, t)$ be the probability distribution of $\mathbf{x}$ as a function of time. Species can react with one another through any of $m$ chemical reactions. For such a chemical reaction $r_i$, let $\mathbf{v}_i \in \mathbb{Z}^n$ be its stoichiometric vector, describing which species will be consumed or produced when this reaction fires, and let $a_i(\mathbf{x}) \in \mathbb{R}^+$ be the propensity of reaction $r_i$, or probability per unit time of reaction $r_i$.

As an example, let us consider a small system consisting of three species $A$, $B$ and $C$, starting with ten $A$-molecules, three $B$-molecules, and zero $C$ molecules. At the initial state, we would then have $\mathbf{x} = [10, 3, 0]$. Let us now consider a single chemical reaction $A + B \rightarrow C$. Its stoichiometric vector would be $\mathbf{v}_1 = [-1, -1, 1]$.

Importantly, the propensities only depend on $\mathbf{X}(t)$, the current state of the system, and not on the current time itself, nor on previous states. Additionally, we assume no two reactions can fire simultaneously. As such, the system fulfills the Markov property and can thus be modeled as a Continuous Time Markov Chain (CTMC) $[61, 62]$, where states are indexed by $\mathbf{x}$ and transition rates are given by the propensity functions. Thus, the time between two consecutive chemical reaction will be exponentially distributed. An important assumption when deriving these rates is that the system is well-mixed, i.e. particles are uniformly distributed across the whole...
domain. A detailed justification of the propensity functions can be found in [61, 64].

Following the CTMC formalism, the temporal probability distribution of $x$ is the solution of the Kolmogorov forward equation, also referred to as the Chemical Master Equation (CME):

$$\frac{\partial}{\partial t} p(x,t|x_0,t_0) = \sum_{i=1}^{m} a_i(x - \nu_i) p(x - \nu_i|x_0,t_0) - a_i(x) p(x,t|x_0,t_0), \quad (2.1)$$

for a given initial state $x_0$ and initial time $t_0$.

Unfortunately, solving this equation numerically is generally intractable due to the number of possible states. The Finite State Projection method (FSP), developed by Munsky et al. [102] and later improved by Burrage et al. [25], makes it possible to solve the CME numerically with some error bounds. In the FSP, unlikely states — relative to an error criterion — are ignored so that solving the equation comes down to computing the exponential of the Markov chain’s transition matrix. The gist of the method is the derivation of a guarantee that the truncated solution is close enough to the exact solution, with respect to the error bounds. A detailed review of direct methods for solving the CME can be found in [46].

In the thermodynamic limit, i.e. when the number of molecules and the size of the system tend to infinity while the species’ concentrations remain constant, the CME can be approximated in various ways [64]. A detailed study of these approximations and how they relate to each other can be found in [53].

2.2 Simulation and Sampling Methods

For most practical applications, however, solving the CME numerically is intractable, even with the FSP. This is due to the exponential scaling of the number of states with respect to the number of species, or curse of dimensionality. This needs not be a fatality as complementary techniques have been developed to generate realizations of the system evolving in time, as shown on Figure 2.1 on page 19. These methods not only make it possible to approximate summary statistics of $X(t)$, such as its moments, but also give additional insights into the temporal evolution of the system. In this section, we briefly review the most popular simulation methods for CTMC.

Developed by Gillespie in [61, 62], the Stochastic Simulation Algorithm (SSA) is probably the most popular sampling method for the CTMC in systems biology. Given the current state $x$, we sample when the next reaction is going to fire, and which reaction it will be. Specifically, let $r_j$ be the next reaction and let $t + \tau$ the time when it happens. Let $p(\tau, j|x,t)$ describe
the joint probability distribution of these two random variables. It can be shown that this density is given by:

\[ p(\tau, j | x, t) = e^{-a_0(x)} \tau a_j(x), \]  (2.2)

where \( a_0(x) = \sum_{i=1}^{m} a_i(x) \).

Given two random numbers \( u_1 \) and \( u_2 \), drawn from a uniform distribution \( U(0, 1) \), it is then possible to sample the time and the nature of the next reaction such as

\[ \tau = -\frac{\log(u_1)}{a_0(x)} \]  (2.3)

and \( j \) is the smallest integer so that

\[ u_2 \leq \sum_{i=1}^{j} \frac{a_i(x)}{a_0(x)}. \]  (2.4)

Repeating this process until the end of the simulation, we can then generate a statistically exact trajectory from the CME. This procedure is summarized in Algorithm 1. Figure 2.1 shows ten realizations for a system involving two species (proteins and mRNA), simulated with the SSA. In this systems, mRNA and proteins form a negative feedback loop, where the binding of proteins to genes represses the transcription of mRNA. With suitable parameters, this process tends to generate stochastic oscillations in both mRNA and protein numbers. These oscillations appear clearly when looking at the individual trajectories, but not when looking at the solution of the CME. By generating many realizations of this process, we can approximate the first moments and other summary statistics of \( X(t) \).

**Algorithm 1:** Stochastic Simulation Algorithm (direct method)

**Input:** The end time \( T \), the initial state of the system \( x \)
Set \( t \leftarrow 0; \)
while \( t < T \) do
  Compute all \( a_i(x) \) and \( a_0(x) = \sum_{i=1}^{m} a_i(x) \);
  Draw \( u_1 \) and \( u_2 \) from a uniform distribution \( U(0, 1) \);
  Compute \( \tau \) and \( j \) according to Eqs. 2.3 and 2.4;
  Update \( x \leftarrow x + v_j \) and \( t \leftarrow t + \tau; \)
  Save \((x, t)\);

When the total propensity function \( a_0(x) \) is high, \( \tau \) will take small values. Thus the system will only advance by small time steps, making the SSA inefficient. Introduced in [63], \( \tau \)-leaping is a technique that partially addresses this problem. Let \( \tau \) be a time interval\(^1\) so that the propensity of

---

\(^{1}\)NB: despite having the same notation in the literature, this time interval is unrelated to the time of the next reaction.
each reaction is nearly constant during that interval:

\[ a_j(x) \approx \text{constant in } [t, t + \tau] \quad \forall j \in [1, m]. \]  

(2.5)

It can be shown that the number of reaction firings in that time interval follows a Poisson distribution with mean \( a_j(x) \tau \). Instead of firing one reaction at a time like in the SSA, the algorithm will then leap forward by a fixed time \( \tau \) at every step and sample how many times each reaction has been triggered in that interval. The main difficulty is to select an adequate leap interval \( \tau \): if it is too short, many leaps will contain no reactions and the efficiency of the algorithm will be negatively affected. If it is too long, the simulation becomes inaccurate and there is a high risk that too many reactions are triggered during one \( \tau \)-interval, i.e. more reactants than available will be consumed, leading to inaccurate results or even negative populations. Many \( \tau \) selection methods have been developed to tackle this issue, for instance by Cao et al. [28]. When \( a_0(x) \) is high despite the number of reactants being low, however, \( \tau \)-leaping will bring little improvement, even with the best efforts to choose the step size. Hybrid methods, such as the Slow-Scale SSA Method [27], have been proposed to handle these specific cases [87].

As of today, StochKit [113], Gillespy [2], Virtual Cell [101], COPASi [76] and StochSS Live! [81] are some of the most popular software to study well-mixed biochemical systems.

### 2.3 Reaction-Diffusion Models

In Section 2.1, we introduced the CME and the main assumption it relies on, namely, the well-mixedness of the system. When this condition holds, molecules are uniformly distributed across the whole cell and the time between consecutive reactions is long enough that molecules can diffuse evenly before the next reaction. Sometimes, the well-mixed assumption is too strong, either because some dynamics at the microscopic level are not captured by the CME, or because of the geometry of the cell that is not captured in the model [50, 92, 123, 124, 127]. Spatial models have been developed to tackle this issue. These models include more details about how molecules diffuse in the cell and keep track of their position in time, at the cost of increased computational time. In this section we review some of the most common reaction-diffusion models. These four approaches are summarized in Figure 2.2. For a more detailed review about available approaches, the reader is referred to [24].
Figure 2.1. Example of trajectories generated with the SSA for a small genetic regulatory network involving two species (proteins and mRNA). The trajectories shown in lighter colors illustrate the stochastic variation between individual realizations of the system.

Figure 2.2. Four different approaches to modeling reaction-diffusion system: (a) space is discretized in a microscopic lattice and each grid point can only be occupied by one molecule at a time. Each molecule then diffuses following a random walk and reacts with its neighbors with some probability. (b) Space is discretized in a mesoscopic lattice where molecules can share the same grid point. Molecules only react with molecules in the same voxel and diffuse randomly to neighboring voxels. (c) Time is discretized and the new position of each molecule is sampled at every time step according to Brownian motion. If molecules are closer than some reactivity threshold, they trigger a chemical reaction with some probability. (d) Protective domains are drawn around single molecules and pairs of molecules. The time of the next exit or reaction event is sampled and the system is updated accordingly.
2.3.1 Lattice Methods

In this section we describe so-called Lattice Methods, where space is discretized into a grid and the molecule’s position is restricted to these grid points. First, we present Microscopic Lattice Methods, where the size of the voxels is approximately the size of a molecule. We then present methods based on the Reaction-Diffusion Master Equation framework (RDME), where the mesh is usually coarser and voxels can contain several molecules.

Microscopic Lattice

In this framework, grid sites can only be occupied by one molecule at a time. Molecules then move randomly to neighboring grid sites and may react with their neighbors, depending on the rules set by the modeler. In essence, this model can be classified as a cellular automaton, where “agents” (in our case molecules) move on a grid and change state according to a set of rules defined by the modeler.

This approach is well suited for dealing with molecule-scale phenomena — such as macromolecular crowding or volume exclusion — as it provides a way to efficiently model the volume occupied by each molecule [12]. Available Software for microscopic lattice methods include Spatiocyte [11, 12] and GridCell [22]. For an analysis of performance and accuracy of this method in the Spatiocyte software, the reader is referred to [32].

The Reaction-Diffusion Master Equation

In the RDME framework [122, 44], we allow many molecules to occupy the same voxel. Each voxel is then assumed to be well-mixed and chemical reactions can only take place within the same voxel. Molecules can diffuse to neighboring voxels. Essentially, the RDME is a generalization of the CME, where \( m \) reactions can take place in each individual voxel and where diffusion events between neighboring voxels are modeled as extra reactions [64].

Generating sample trajectories of the RDME with the SSA, however, poses new challenges and solving them has been an active field of research in recent years. First, the number of voxels drastically increases the number of diffusive events to consider, which makes the traditional approaches based on the SSA or \( \tau \)-leaping impractical. The next subvolume method tackles this issue by first sampling the voxels undergoing a chemical reaction and then sampling which reaction occurs [45]. Second, in its original version, the RDME was restricted to Cartesian grids, making it ill-suited for complex geometries. Extending it to unstructured grids was proposed in [47], although in that case computing the jump rates between voxels poses some challenges [95, 17]. Finally, two new issues arise when the mesh size decreases: molecules become less and less likely to be found in the same voxel, or in other words bimolecular reactions will fire more and more sel-
dom, and reaction rates become inaccurate for bimolecular reversible reactions as the equilibration time diverges. The first problem was solved by Isaacson by allowing chemical reactions to span several neighboring voxels [78, 79]. Solutions to the second problem have been proposed by adapting the reaction rates to take into account the diffusion constant and the size and geometry of the mesh [49, 72].

Available software based on the RDME framework include MesoRD [71, 51], StochSS [42], URDME [41], StoSpa2 [16] and STEPS [73]. For a more detailed review on the RDME, the interested reader is referred to Smith and Grima [118].

2.3.2 Off-lattice Methods

In this section we assume particles diffuse continuously in space through Brownian motion following Fick’s second law [20]:

\[
\frac{dp}{dt}(r,t) = D \Delta p(r,t)
\] (2.6)

where \( p \) is the probability distribution of a molecule at position \( r \) and at time \( t \), and \( D \) is the diffusion constant associated with that molecule. Additionally, in the Smoluchowski formalism, molecules are modeled as hard spheres, reacting upon collision with some probability. The methods presented in this section do not rely on discretized space, but keep track of the exact position of each particle throughout the simulation. In this section, we present two approaches to generate realizations of this model.

**Discrete Time Particle Methods (DTPM)**

The first approach consists in discretizing time and computing the position of new molecules at each new time step according to Equation 2.6. The main challenge here is to compute the probability that two molecules collided within one time step given their positions before and after this step. Given the computational cost of solving the equations from the Smoluchowski model, most implementations choose a simpler approach, where particles react if they end up within some distance of one another. Although it is an approximation, this approach has shown good results for small enough time steps [9].

Smoldyn [9, 8] and MCell [121, 83] are popular software capable of simulating such a system, although MCell is designed for even finer scales.

**Green’s Function Reaction Dynamics**

The Green’s Function Reaction Dynamics method (GFRD) [135], and its recently improved version eGFRD [119], take another approach where analytical formulas for the reaction probabilities are derived using Green’s functions.
Specifically, spherical protective domains are built around single particles and pairs of particles. The simple geometry of these domains makes it possible to efficiently sample the time at which the particles will exit this volume or trigger a reaction. These events are then sorted in a queue and resolved one after the other. When an event is resolved, the positions of the particles are updated and new protective domains are built. The result is a statistically exact realization of the Smoluchowsky model. Of course this is only possible for relatively simple geometries, even though efforts are ongoing to fill this gap [119].

2.4 Challenges in Modeling Spatial Stochastic Chemical Kinetics

There is a clear and well established mathematical hierarchy between all these modeling approaches, from the coarse deterministic ODE system down to the detailed particle-based models. Specifically, there are mathematically clear assumptions defining when each approach is valid. For instance, it is well known that the well-mixed assumption holds when chemical reactions are slow enough relative to the diffusion constant for the molecules to become uniformly distributed between reaction events. However, it is rarely possible to determine with certainty if these assumptions hold in practice. The only option left to determine which approach is most suitable is then trial and error.

Thus, there is a need to compare existing approaches for different test cases and evaluate how each of them perform. Of course, this requires a common standard to describe models, and a common framework, to make sure they are all simulated under the same conditions. A lot of effort has been put into non-spatial approaches, both to run and compare these models within the same software [101] and to establish the Systems Biology Markup Language (SBML) as a standard format to define these models [77]. Comparing spatial models, however, is hard, as there are no well established standards or common framework to define, run and compare these models. A similar endeavor is thus needed for this category of models [89].

In [82], Johnson et al. set out to compare a wide range of models, from ODEs to detailed particle models. They considered many models, from “unit test cases”, such as bimolecular association in two and three dimensions, to more complex cases, such as the circadian clock model from Vilar et al. [136]. For the simplest cases, these models all show similar results. For more complex and realistic cases, however, they observed that choosing the wrong approach can lead to major outcome discrepancies. Based on these conclusion they presented a framework to help modelers decide which model to use. However, they only present their result for selected parameter sets. How their conclusions would change if the parameters of
their models were chosen differently thus remains an open question. There is therefore a need for systematic studies comparing various modeling approaches over a wide range of parameters. In Chapter 4, we detail how this thesis contributes to solving this issue.
3. Agent-Based Modeling for Multicellular Systems

Multicellular tissues are dynamic bodies where cells are born, differentiate and die. During their lifetime, cells interact with one another through mechanical and biochemical interactions. Understanding how these finescale interactions between individual cells give rise to robust, macroscale phenomena, such as embryonic development or skin patterning, is a central question to developmental biology. Agent-Based Models (ABM), also referred to in the literature as individual-based, or cell-based models, are a popular and common approach to study this question.

In contrast to continuous approaches, where only cell density is taken into account, agent-based approaches consider every individual cells in the tissue. These are modeled as autonomous agents, interacting with their environment and with other agents through a set of predefined rules. These rules can be either based on physics (e.g. when the position of an agent is determined by the forces applying to this agent), or ad hoc (e.g. a cell dies unless it has exactly two or three neighbors). ABMs offer great flexibility and can include a wide range of details at various levels of realism, depending on the task at hand.

In practice ABMs can be used to explore and test hypotheses before committing to expensive wet lab experiments [130]. In [40], Drasdo et al. show how ABMs can be used to model monolayers and multicellular spheroids, and how these models can be use to explain pattern formations in cell development. More generally, ABMs have been very successful at revealing insights into the dynamics of cancer [97, 110], and morphogenesis [55, 67, 128].

In this chapter, we give a brief overview of existing agent-based approaches. ABMs can be classified into two categories, namely on-lattice and off-lattice models. Each category is presented in more detail in Sections 3.1 and 3.2. In Section 3.3, we give a quick overview of the shortcomings of such methods. For an extensive review on available agent-based approaches for cell mechanics, the reader is referred to [134].
Figure 3.1. The four main categories of ABMs. (a) Cells behave as spheres moving on a lattice. Each lattice site can be occupied by at most one cell. (b) Cells can span multiple grid points. They spread and move to minimize their energy potential. (c) Cells are modeled as points moving in continuous space. They interact with their neighbors through adhesion and repulsion forces. Their shape is modeled as spheres or Voronoi polyhedra. (d) The shape of each cell is modeled explicitly and changes according to the pressure forces from neighboring cells.

3.1 On-Lattice Models

3.1.1 Cellular Automata

In the Cellular Automaton (CA) framework, space is discretized into a grid where each cell occupies exactly one voxel, and each voxel can be occupied by at most one cell. Cells can take one out of a finite number of states. Cells then migrate, divide, change state and die according to a set of predefined rules depending on their environment (e.g. the concentration of some chemicals), or on the presence of neighbors in a specific state [38].

The simplest, and arguably most famous example of a cellular automaton is Conway's game of life. There, cells can have only two states, alive or dead, and do not move. A given cell can then change state depending on the number of alive cells in its direct surroundings. Despite its simplicity, this model exhibits an incredible range of dynamic patterns [3].

Cellular automata are simple, and usually easy to implement and cheap to simulate. However, this simplicity can sometimes hinder the realism of these types of models. For instance, a uniform space discretization will lead to glitches in the cell patch, reflecting the geometry of the mesh, as shown in [134]. This can be avoided for instance by using a Voronoi tesselation, or by adapting the rules dictating how cells spread on the mesh [48]. Additionally, when \textit{ad hoc} rules are used, it can be difficult to relate their parameters to physical, experimentally measurable quantities, thus limiting the scope of these models.

CA models have demonstrated their power when it comes to the study of pattern formation, typically on animal skin [38]. A notable, recent example of a successful application of cellular automata is given by Manukyan et al. [93]. In this study, the authors were able to show that the patterns of
skin scales in lizards could in fact be modeled as a cellular automaton for which they identified the governing rules.

3.1.2 Cellular Potts Models

By nature, CAs cannot represent cell shapes, although in some cases it might be a modeling requirement. In the Cellular Potts framework (CP), space is also discretized into voxels, but this time a cell can span several voxels. Cells then spread or contract on this mesh. The main idea of the CP model is to derive a formula to compute the total energy of the system and to explore the states of lowest energy.

In its original form, the total energy is formulated as the Hamiltonian:

\[ H = \sum_{\text{neighbors}} J_{\tau_k, \tau_l} (1 - \delta_{\sigma_k, \sigma_l}) + \lambda \sum_{\sigma} \left( A(\sigma) - A_{\tau(\sigma)} \right)^2 \]  

(3.1)

The first summation, going over all pairs of neighboring voxels \((k, l)\), represents the energy penalty for having two neighboring voxels of different types. \(J_{\tau_k, \tau_l}\) is the contact energy per unit of length between cells of types \(\tau_k\) and \(\tau_l\), \(\sigma\) is the index of a cell (also called spin) and \(\delta\) is the Kronecker delta. The second sum, where \(A(\sigma)\) is the area of cell \(\sigma\) and \(A_{\tau(\sigma)}\) is the target area for cells of type \(\tau(\sigma)\), represents the penalty for having a cell that is too small or too big compared to its target size. \(\lambda\) is a Lagrange multiplier specifying the balance between these two terms.

This approach was originally developed to test the differential adhesion hypothesis and explain how a population including cells of different types would often end up in a configuration where cells of the same type remain close to one another [66, 68, 100]. Modern successful applications of the CP framework include tumor growth, morphogenesis, wound healing and vascular system development [126, 120].

The Metropolis algorithm [19] is used to generate stochastic realizations of this model and goes as follows: at each time step, one spin change is proposed. If this change decreases the Hamiltonian, it is immediately accepted. If not the change is only accepted with probability \(p = e^{-\Delta H/T}\), where \(\Delta H\) is the change in energy after the spin change is applied, and \(T\) is the so-called temperature of the system. A high temperature implies that proposed changes will easily be accepted, even if it places the system in a high-energy, unlikely state. On the contrary, with a low temperature, most proposals will be rejected and the system will tend to stay in local minima. Once enough proposals have been considered, the simulation continues until enough steps have been performed.

It is important to note that, in its original implementation, this model does not forbid cells to be fragmented, which would be a clearly unrealis-
tic outcome. This can be circumvented by only considering changes that preserve the local connectivity of a grid site [43]. As with CAs, CPs are susceptible to shape glitches due to the topology of the spatial discretization. Popular framework for simulating CPs include CompuCell3D [126], Morpheus [120] and Artistoo [140].

3.2 Off-Lattice Models

Although on-lattice models are easy to implement, calibration of these models is often difficult as they rarely rely on measurable physical parameters. Indeed, in reality, cells in a tissue pull and repel each other depending on interfacial tension and on their internal pressure. As a coarse approximation, cells act as if they were connected by springs (Figure 3.1c): when two cells are too close to one another, they repel each other, while if they are too far apart, adhesion forces will bring them back together. In this section, we give an overview of more realistic models, where no lattice are used and the forces between cells are explicitly modeled. Van Liedekerke et al. [133] provide an extensive review of such models.

3.2.1 Center-Based Models

In center-based models, only the cells’ midpoints are considered. The forces applying to these points are then computed depending on the topology of the tissue and the distance between the cells. Using Newton’s second law, we can derive a formula for the speed of each cell, assuming that inertia is negligible:

$$\frac{dx_i}{dt} = \frac{1}{\eta} \sum_{j \neq i} F_{ij}$$

(3.2)

where \(x_i\) is the position of cell \(i\), \(\eta\) is the drag coefficient, and \(F_{ij}\) is the directed force from cell \(j\) to cell \(i\). The resulting system of ODEs can then be solved using standard numerical methods. In Figure 3.2, we illustrate such a model by starting with 25 cells arranged in a honeycomb pattern (left pane) and letting them divide during the simulation. After undergoing a high-compression phase, the cells relax to the configuration as shown on the right pane.

Several approaches have been developed to model interaction forces between cells. In early works [40], linear spring forces were used. However, some remarked that due to the weak repulsion at close distance and the strong adhesion when cells are further apart, simulated cell populations had a tendency to collapse under the strength of adhesive forces [107]. Furthermore, the discontinuity at the maximum interaction distance lead to extra numerical difficulties. For these reasons, forces with strong repulsion
at close distance, and minimizing the discontinuity at the cut-off distance are preferred in more recent models. More detailed approaches also exist, such as the Johnson-Kendall-Roberts force function [34], although at an increased cost in terms of computations [26].

![Figure 3.2. Example of a center-based simulation. Two types of cells, red and blue, start in an initial honeycomb configuration (left pane). During the simulation, they divide in a random direction and give birth to cells of the same type. The system then relaxes to the configuration shown on the right pane.](image)

In the real world, cells only interact directly with their immediate neighbors. Defining this neighborhood when only considering the cell’s midpoints, however, is not entirely straightforward. One approach is to approximate the shape of a cell as a sphere and to define a maximum interaction radius. Only the midpoints within this radius are then considered when computing the forces applying to each cell. In the second approach, the set of neighbors is determined from the Delaunay triangulation of the midpoints. The shape of the cells is then the dual Voronoi diagram of this triangulation. While the first approach is computationally cheaper, it can lead to unrealistic results if the force functions are not chosen with care [107]. The second approach, on the other hand, is more computationally expensive and requires setting up ghost cells to constrain the shape of the real cells [96, 114].

When the purpose is to study the emergent behavior of tissues comprising millions of cells, such as tumor growth, this modeling framework strikes a good balance between ease of use, physical realism and computational efficiency. Center-based models have been used successfully to study 3D models where cells actively proliferate, such as models of the intestinal crypt [96] or monolayer and spheroid growth [39, 58]. However, choosing how to model forces, which numerical solver to use, and understanding how these choices affect the simulation are still open problems. Indeed,
the main specialized open source software supporting center-based models (namely PhysiCell [60], MecaGen [37] and Chaste [35, 99]) all use different force functions with different mathematical properties. Elucidating how these differences affect the simulation requires further studies.

3.2.2 Deformable Cell Models

Sometimes it is necessary to explicitly incorporate the shape of each cell in the model, and track how it varies over time. One approach is to model cells as polygons on a surface [56] (Fig. 3.1d). The forces applying to each vertex are specific to each model but are usually derived from the internal pressure of the cells, surface tension on the membranes, and cell-cell adhesion. Although the core idea behind these models is quite elegant, extra care needs to be taken to ensure that all the polygons are well-formed and no two elements intersect. For that reason, most vertex models have been restricted to two dimensions, or to a specific class of polyhedra [5]. Extending vertex models to general three-dimensional shapes is theoretically possible but is both technically and computationally challenging and remains an open area of research [56, 75]. As a notable example, Farhadifar et al. [52] used a vertex model to study the physics behind pattern formations in Drosophila wing epithelia. They showed how two parameters, line tension and contractility, influence the geometry of the cell layer for various patterns observed experimentally.

Many more approaches have been proposed, modeling cell shape and cell-cell interactions in even more details [55], with e.g. finite elements [31], immersed boundaries [110, 128] or sub-cellular elements [103, 112]. Due to the increased level of details, these models are usually limited to simulate a few hundred to a few thousand cells, if specialized hardware is used [33].

3.3 Outlook

The availability of high-performance computing resources to nonexperts has made computational models ubiquitous to biological modeling in recent years. Contrary to the case of intracellular chemical kinetics seen in Chapter 2, where the mathematical hierarchy between models is well understood, no such ordering has been established in the case of ABMs. Hence, the question of model choice becomes even more important as experience rather than mathematical guidelines guides modelers in choosing an appropriate modeling approach. Thus, there is a need for extensive quantitative comparison of these models.

Care must be taken to ensure that biological conclusions mostly depend on biological assumptions and are robust to implementation and numerical details. Several studies have examined this question in details. In [106],
Osborne et al. compared different modeling approaches in various case studies and identified scenarios where each model type was most appropriate. In [84], Kursawe et al. showed that numerical parameters could have a significant and counter-intuitive influence on the results of vertex models. In [107], Pathmanathan et al. examined tissues simulated with center-based models and found that force function had little influence on the mechanical properties of these tissues, provided the force laws were chosen to avoid instabilities. Finally, in [13] K. Atwell investigated the influence of force function and numerical solver choice on center-based models and found that Chaste’s default parameters were sufficient to generate accurate simulations with the forward Euler method [69]. These contributions demonstrate why the influence of purely numerical parameters and implementation choices cannot always be neglected and must be studied in order to confirm the validity of the results. In Paper II, we provide an in depth study of the effect of force function and numerical solver choice for CBMs.

As we have seen in this chapter, there are many software available to simulate and study cellular mechanics, with Chaste [35, 55] being the only one providing implementations for CA, CP, CBMs and DCMs altogether. However, these software’s main goal is to provide modelers with many realistic biological features (such as cell cycle or substrate modeling). They are therefore ill-suited for numerical studies, which require adjustment of parameters usually unavailable to end users and there is a need for an alternative approach. In Paper III, we propose a Python framework specifically designed for the numerical study of CBMs.

Finally, the recent increase in data available from lab experiments raises the question of how to integrate this data to computational models to allow for accurate predictions and hypothesis testing. As an example, Kursawe et al. [85] studied how data from laser ablation experiments could best be used to calibrate vertex models and make testable predictions. They identified which summary statistics allowed for accurate parameter inference and showed that repeated experiments were crucial to decrease the uncertainty on these parameters.
4. Multiscale Modelling of Multicellular Tissues

In a recent manuscript, Fletcher et al. [54] described seven current challenges in the multiscale modelling of multicellular tissues, namely (i) model construction, (ii) model calibration, (iii) numerical solution, (iv) software and hardware implementation, (v) model validation, (vi) data/code standards and benchmarks, and (vii) comparing modeling assumptions and approaches. In this chapter we detail how the papers presented in this thesis contribute to solving these challenges.

4.1 Multiscale Simulations with Orchestral

Modeling how cells interact with their microenvironment has been thoroughly studied and in fact ABM solvers often include a PDE solver to simulate the diffusion of substrates and how they interact with the cell population [99, 120, 59]. For instance, a multiscale model can be used to study how oxygen intake from the vascular structure influences cancer growth [91]. Coupling cell mechanics with the cells’ internal biochemical dynamics, however, is a difficult problem. As we have seen in Chapter 2 and 3, there are many different approaches to model intracellular chemical kinetics and cell mechanics. There are many high-quality software to simulate cell mechanics and equally many to simulate stochastic chemical kinetics but only a few can handle both scales with sufficient amount of detail. PhysiBoss [86], CoGNaC [111] and MecaGen [37] are such examples. Other ad hoc successful examples of multiscale modeling using CompuCell3D include [74] and [6]. However, these approaches only handle chemical kinetics on a very coarse level, namely with ODEs or Boolean networks. Just as it is not clear which model to choose with intracellular chemical kinetics and with cellular mechanics, it is even less clear how to do so when putting theses two scales together. Developing such multiscale software is technically challenging, first because of the complexity of the models involved, and then because of the computational cost of simulating such system and the need for scalable, high-performance implementations. There is thus a need for a fast modular framework to put together existing software, where one could easily replace its components to adjust the level of detail at different scales.
In Paper I, we develop Orchestral, a framework capable of leveraging the power of existing software to create a distributed multicellular simulation. The simulation is split into two parts, namely intracellular kinetics and cell-cell signaling, which are simulated in turn for a short time step $\Delta t$. Because internal dynamics are independent across cells, they can easily be parallelized. Similarly, pair-wise signaling can also be parallelized across pairs of cells. To ensure our framework can handle any kind of simulation software available in the literature, the setup is divided into independent modules that can be run from the command line. There are three types of modules:

- **Simulation modules:** this is where the bulk of the computations takes place. These modules will typically use publicly available software, and will be executed once at every time step and for each cell and each pair of cells, for the intracellular kinetics simulator and the pair-wise signaling simulator, respectively. One can also use different simulators within the same multiscale simulation (e.g. depending on the cell type).
- **Translation modules:** these modules read the output from the simulation modules and create the input files for the next simulation modules. For instance, after the internal dynamics of all cells have been simulated for a short time step $\Delta t$, translation modules will extract relevant molecules from the output files (typically, molecules on the membrane of the cell) and assemble them to create input files for the pair-wise signaling simulations.
- **Orchestration module:** this module execute each module in the correct order and distribute each task across the available computing resources. It also makes sure the relevant input and output files are available to the modules that need them.

Figure 4.1, right insert, describes the execution flow of Orchestral. In practice, external software (such as Smoldyn [8] or eGFRD [119]) will be used for the simulation modules while the translation modules will be written by the user and depends on how the different models are coupled. The orchestrator is provided but can easily be replaced to take advantage of a specific architecture. Essentially, one can replace any simulation module with any other software. The only requirement is to adapt the translation modules to the new input and output formats.

Although only fixed tissues, where cells do not move nor divide, could be simulated in Paper I, our approach can be extended to include cell mechanics as well. The setup includes three types of simulation modules, namely cell mechanics, intracellular chemical dynamics, and cell-cell signaling. Since chemical kinetics and cell mechanics typically take place over different time scales, we use a second, larger time step to alternate between chemical kinetics (internal and signaling), and cellular mechanics.
Figure 4.1. Execution flow of the Orchestral framework. Simulation modules are represented in blue, translation modules in pink, and the orchestration module is in green. The circle insert contains the modules relative to chemical kinetics within and between cells. There the simulation alternates between internal chemical kinetics ($\chi$) and pair-wise cell signaling ($S$). The translation modules extract the information from the previous simulations and create input files for the next step. At a larger scale, the simulator alternates between cell mechanics ($\phi$) and chemical kinetics, where new translation modules reflect the cells’ position on the signaling topology and update the cell mechanics simulation with information about the internal state of the cells. The orchestrator distributes the computation over the resources available in a task-based manner and makes sure each module is executed in the right order. Additionally, it ensures the input and output files are shared throughout the computing resources.
The main difficulties in this case are that, contrary to internal kinetics that could be distributed over the whole cell population (an embarrassingly parallel task), cell mechanics algorithms are more difficult to parallelize, and that, the topology of the cell population changes every time the cells’ position are updated. Figure 4.1 summarizes the execution flow of Orchestral when cellular mechanics are included. In Figure 4.2 we show several snapshots of such a simulation run with Orchestral, where cellular mechanics and cell divisions are simulated with a cellular automaton, while detailed intracellular dynamics of the Delta-Notch signaling pathway [57] are simulated with Green’s function reaction diffusion dynamics [119].

Figure 4.2. Multiscale simulation of a Delta-Notch signaling model combining a CA model for cell mechanics and GFRD for intracellular chemical kinetics. Snapshots are taken every 11 hours in simulation time.

Multiscale simulations are computationally expensive as they require to accurately reproduce events happening at short time scales while quantities of interest are only observable over longer timescales. Thus, there is a need to efficiently take advantage of high performance computing resources available to the modeler. In Paper I, we implemented a platform agnostic orchestrator using the Dask library [36] and demonstrated its scalability over both high performance computing clusters and the cloud platforms provided by the Swedish National Infrastructure for Computing (SNIC) through the Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX).

Non-reproducibility in science is a big issue [14]. When it comes to computational systems biology, the problem is exacerbated by the fact that there is no well established standard to easily compare results coming from different software [89]. There is thus a need to establish a standard data format for multiscale cellular simulations and to establish benchmarks to systematically compare available software. Although Orchestral does not directly contribute to these goals, it provides an alternative, thus circumventing the problem. Indeed, Orchestral makes it possible to relatively easily replace parts of the simulation while only adapting translation modules and thus to compare different software in multiscale settings. When a standard output format is finally established, there will be no need to adapt the translation modules anymore and it will be possible to replace simulation modules on the spot. All in all, the Orchestral contributes to solving Challenge (iv), (vi) and (vii) proposed by Fletcher et al.
4.2 CBMOS, a Python Framework for the Numerical Analysis of Center-Based Models

Center-based models are a popular modeling approach to study cellular mechanics. Indeed, these models strike a good balance between modeling expressiveness and computational efficiency. Typical software provide a feature-rich environment allowing modelers to incorporate many biological details into their models. The numerical parameters used in theses software, however, remain poorly studied. As an example, Chaste [35], Physicell [60] and MecaGen [37], the three main open source center-based software, each implement a different family of force functions (i.e. cubic, piecewise polynomial, and generalized linear spring) but the detailed effect of the choice of this force function on the simulation is yet to be fully understood.

In Paper II, we quantitatively investigate the impact of force function formulations and numerical solvers on simulations of center-based models. Specifically, we show that even when the time step is chosen so that the simulation is numerically stable, it can still lead to unrealistic simulations, and that this new bound critically depends on the force function in use. We also show that second order methods such as the midpoint method [30] or the Adams–Bashforth [69] method do bring efficiency improvements when care is taken to ensure the force functions are sufficiently smooth.

In Paper III, we extend our work into an open source Python package, CBMOS. Python is a programming language that stands out for its ease of use and simplicity. Indeed, although performance is important, being able to easily refactor code to quickly investigate different setups is sometimes even more important. In the last decade, Python has become a key player in scientific computing and data analysis [70, 98]. In particular the development of the NumPy [70] and SciPy [137] libraries has made it possible to achieve competitive performance in many areas of scientific computing. Specifically, the recent standardization of NumPy’s Application Programming Interface (API) has allowed for the development of a multitude of new specialized implementation of the array programming paradigm [70], such as Dask [36], CuPy [105], or PyData/Sparse [1].

CBMOS provides a simple interface to set up center-based models and vary their numerical parameters, such as force functions, numerical solvers or time steps. Essentially, it is possible to plug in any force function or numerical solver in the model — many of which are already implemented and available in the CBMOS Python package — by only changing the parameters used when creating the model. Internally, CBMOS relies on the array programming paradigm and NumPy to perform the bulk of the computations, which makes its code both efficient and legible. Most notably, it offers support for General Purpose Graphics Processing Units (GPGPU) through
the CuPy library [105], allowing for simulating up to 10,000 cells in a cou-
ple of seconds. Furthermore, the recent development of Google Colabora-
tory [21] makes it possible to run CBMOS simulations in a Jupyter Note-
book accessing cloud and GPGPU resources from a regular web browser.
Overall, CBMOS provides modelers with a user-friendly interface opti-
mized for fast prototyping and the numerical study of center-based mod-
els. Yet, it is not meant to be a competitor to other center-based simulation
software but rather a complementary approach to ensure biological models
are numerically robust.

We exemplify once again the utility of our approach by studying the
potential benefits of using an implicit method, namely the backward Euler
method [69]. We show that, even though it allows for better accuracy for
large time steps, the additional cost of solving a linear system outweighs
the performance benefits of using larger time steps for the same accuracy.
Furthermore, our software is flexible enough to be used in further studies,
for instance on the benefits and costs of using adaptive time stepping or the
effect only resolving cellular events (such as cell division or cell death) in
between time steps or resolving them at the exact time they occur.

When modeling tissues, errors can arise from multiple sources, such as
model choices, implementation details or numerical parameters. Identifying
the source of these errors requires careful investigations and is often
overlooked. In Paper II and III, we contribute to elucidating this question
in the case of center-based models by investigating the influence of model
choices (the force functions) and numerical parameters (the time step length
and the numerical solver used) on the simulation error. Specifically, we
study the stability of various numerical solvers at different time step sizes
and provide advice on how to choose these time steps, not only to ensure
stability but also to avoid unphysical results. Furthermore, by taking ad-
vantages of the Python ecosystem [98] and the NumPy API [70], CBMOS
leverages modern hardware architectures while maintaining flexibility and
ease of use. Being able to run simulations on Google Colaboratory com-
puting resources contributes even further to increasing the availability of
our software to modelers. Overall, Paper II and III contribute to solving
Challenge (iii), (iv) and (vii).

4.3 Multiscale Modeling, Model Comparison and
Parameter Inference for Stochastic Chemical
Kinetics

As we have seen in Chapter 2, although the mathematical hierarchy be-
tween the different modeling approaches for stochastic chemical kinetics is
well established, only limited practical rules are available to modelers when selecting a modeling level. A detailed model will usually be more accurate, but at a significant increase in computational cost compared to simpler alternatives. On the other hand a cheaper model could have lead to the same conclusions, despite the loss of accuracy.

In Paper IV, we design a multiscale two-compartment model based on the CME but capturing essential spatial features of single cells. In this model, the cell is divided into compartments based on biological features, such as the nucleus or the cytoplasm, where each of them is assumed to be well-mixed. Transition rates between these compartments are then derived analytically based on hitting-time analysis [65]. We then compare this model to a coarser model where the entire cell is assumed to be well-mixed, and to a detailed single particle model based on Smoluchowski dynamics. We perform this comparison in the context of a simple canonical genetic network, the negative feedback loop [123], and over a wide range of parameters. We determine in which areas of the parameter space each model performs best and show that our multiscale model does improve accuracy for a marginal cost in terms of computation time compared to the coarse model. We then illustrate how our model can be used to improve accuracy in Bayesian parameter inference.

Bayesian inference is a computational technique to find model parameters that best explain experimental data, given a mathematical model [138]. Given some observed data \( Y_{\text{obs}} \), the probability distribution of the model’s parameters \( \theta \) is given by Baye’s theorem:

\[
p(\theta|Y_{\text{obs}}) = \frac{p(Y_{\text{obs}}|\theta)p(\theta)}{p(Y_{\text{obs}})},
\]

where \( p(\theta) \) is the prior density of the parameters, i.e. the prior knowledge we have about these parameters before collecting the data, \( p(Y_{\text{obs}}) \) is the probability of observing these data, and \( p(Y_{\text{obs}}|\theta) \) is the likelihood, that is, the probability of observing \( Y_{\text{obs}} \) given the parameters \( \theta \). Computing this posterior distribution \( p(\theta|Y_{\text{obs}}) \) then gives the modeler information about which parameters are most likely to reproduce the observed data.

In most cases of practical interest, however, the likelihood is not tractable. Likelihood-free approximations, such as pseudo-marginal Markov Chain Monte Carlo [10] or Approximate Bayesian Computation (ABC) [125] aim to tackle this issue.

The key of ABC is to rely on simulations to estimate the likelihood. Essentially, given some tolerance \( \epsilon > 0 \) and a distance metric \( d \), the likelihood can be approximated by simulating some data \( S_{\text{obs}} \) and using the following formula:

\[
p(\theta|Y_{\text{obs}}) \approx p(\theta|d(Y_{\text{obs}}, S_{\text{obs}}) \leq \epsilon)
\]

39
As $\varepsilon$ tends to zero, it can be shown that Equation 4.2 tends to the likelihood $p(\theta | Y_{\text{obs}})$ [15].

With this tool in hand, the ABC rejection algorithm can be used to sample parameters from the posterior distribution $p(\theta | Y_{\text{obs}})$. The procedure is summarized in Algorithm 2. In practice, the ABC rejection algorithm can be quite inefficient as many candidates parameters $\theta^*$ will be rejected. ABC-SMC [117, 18, 131] and ABC-MCMC [94] are efficient alternatives addressing this issue.

**Algorithm 2: ABC Rejection Algorithm**

```
Input: Tolerance $\varepsilon$, Number of particles $N$
while #accepted particles < $N$ do
    Draw $\theta^*$ from $p(\theta)$;
    Simulate $S_{\text{obs}}(\theta^*)$;
    if $d(Y_{\text{obs}}, S_{\text{obs}}) \leq \varepsilon$ then
        Accept $\theta^*$;
```

In spite of these improvements there are still fundamental limitations to the accuracy of ABC methods. First, it is known that summary statistics must be used to reduce the dimensionality of the observed data $Y_{\text{obs}}$ to avoid the curse of dimensionality. However, these statistics must be sufficient statistics for the ABC algorithm to converge [109]. Second, all the parameters in the model must be identifiable [90] but proving identifiability has only been achieved for relatively simple models [23]. Hence, from a practical point of view, it is often unclear to the modeler which model to choose, with which summary statistics, and how much data they need to ensure accurate parameter inference.

In Paper V, we demonstrate a practical approach to evaluate and compare parameter inference setups. Specifically, we first generate data with a detailed model (Smoldyn) and for many combinations of parameters across the prior distribution. We then run ABC on each of these data sets, using them as synthetic observed data. To reduce the cost of running parameter inference, we approximate the distance function using Gaussian processes, following the approach described by Järvenpää et al. in [80]. The Gaussian processes are trained with the previously generated data. This makes it possible to avoid running any simulation during the inference itself. Simulations are only needed to generate the initial data set. The full procedure is described in Figure 4.3.

Having computed posterior distributions for all our parameter combinations, we can then evaluate the accuracy of these posteriors and map the error to the parameter space. We can then see where ABC is most accurate,
Figure 4.3. ABC pipeline for parameter inference from synthetic observed data. An approximate distance metric is trained based on Gaussian processes and with our simulated data. This distance metric is then fed into the ABC-SMC algorithm to generate the posterior distribution based on this synthetic data.

or compare two setups and see which one is most accurate on average. This result can then be used to determine which setup is likely to give the best results and use it on real observed data. We conclude Paper V with three case studies illustrating our method using a negative feedback genetic regulatory network. We identified which combination of model and distance metric was most accurate, as well as which chemical species were worth measuring with respect to inference accuracy.

How to choose an appropriate model is the first challenge described by Fletcher and Osborne [54]. With Paper IV, we compare three modeling levels and determine specific concrete guidelines as to which model to choose depending on the parameter regime. In Paper V, we propose a procedure to systematically evaluate modeling setups in the context of Bayesian inference. We exemplify this procedure by comparing these three models in a variety of setups and determine which model is likely to perform best. Additionally, these papers contribute to the seventh challenge as they give insight as to how models compare with each other.

ABC is a popular approach to estimate parameters and calibrate models. Although the theory behind ABC is well defined, in practice it can be difficult to determine if the assumptions behind this theory are met. In Paper V we show how the performance of ABC can be evaluated to determine how likely it is to perform accurately in an experimental setting. This contributes to the second challenge.

Determining what to measure in order to validate a given model is difficult. The approach described in Paper V can be used to evaluate the effect the choice of summary statistics has on inference accuracy. Essentially, this gives us full control on what can actually be measured. We can thus determine in advance which type of experiment and data is likely to give us
the most accurate results. A similar approach has been proposed by Schumacher et al. [115] although not in the context of Bayesian inference. The continuing increase in computational power available makes it possible to systematically evaluate a computational model over a wide range of hypotheses and determine which ones are most robust [88]. Thus, this contributes to solving Challenge (v) as it makes it possible to suggest what kind of data should be extracted from laboratory experiments to yield the best results.

4.4 Outlook

Overall, this thesis contributes to all the challenges highlighted by Fletcher and Osborne [54]. Still, there is a long way to go and the journey toward multiscale modeling of multicellular systems is scattered with pitfalls. In this section we take a look at avenues of research that would build on top of this thesis and solve some of these seven challenges.

The Orchestral framework proposed in Paper I has a lot of potential but would greatly benefit from a well established data standard. As of today, and although switching model components is significantly easier than any other approaches, it still requires to translate the input and output files from one format to another. Were a common standard to be established, switching between models would be essentially free from the user’s perspective. Once this milestone has been reached, it will be possible to use Orchestral at its full potential to bring together existing software components into a distributed simulation.

Even though it only includes a limited amount of biological features, we have demonstrated in Paper III that CBMOS can be used to evaluate the robustness of biological conclusions to implementation choices and numerical parameters. Based on the array programming paradigm, it provides a user-friendly interface capable of simulating up to 10,000 cells in a few seconds, enough for most practical cases when prototyping. In the future, CBMOS could be used to investigate the pros and cons of various numerical methods in the context of time adaptive schemes. Furthermore, adding more biological features will increase the range of application of CBMOS even further.

Lastly, in Paper IV and V, we only focused on one genetic network, namely the negative feedback loop. Although this motif is found in many larger and more complex networks, it would be interesting to compare the three model approaches described in these paper over other genetic motifs [4] and see how these models perform, both in terms of simulation error and in the context of Bayesian inference.
5. Conclusion

Biological tissues can be modeled at different levels. At the intracellular scale, stochastic models are important tools to study and understand the chemical dynamics of gene regulatory networks. At the multicellular scale, discrete cell-based models make it possible to gain insight into how individual cells biomechanically interact with one another. Yet, these scales are not independent from each other, rather, they interact in very intricate ways. For example, the interactions between proteins and genes inside the cells dictate how cells will differentiate and change their mechanical properties, while the position of the cells in the tissue will affect signaling with their neighbors. Hence, to gain a deep understanding of how these two scales combine we must be able to create multiscale models including details from both the tissue scale and the intracellular scale.

There are many challenges associated with such multiscale models, ranging from determining which components to combine to how to calibrate them against experimental data and how to implement the simulator.

In this thesis, we make steps toward multiscale simulation of multicellular systems. Specifically, in the case of stochastic intracellular models, we compare two approaches, one based on the well-mixed assumption and the second based on Smoluchowski dynamics, and show that under some conditions the detailed approach can be approximated by a multiscale model based on hitting time analysis. We then build a computational pipeline to systematically evaluate how these three models compare in the context of Bayesian inference and suggest how this pipeline can be used as a preliminary step to help design wet lab experiments, without any requirements on the models used.

In the case of tissue models of cellular mechanics, we implement a high performance framework for the numerical study of center-based models. We use this framework to study the influence of the choice of force function when modeling cell-cell interactions with respect to the time step size and numerical solver used to simulate the system. We show that large time steps can lead to unrealistic results, even when the simulation is numerically stable, and that the largest acceptable step size to obtain realistic results does depend on the type of force function. Additionally, we show that using the backward Euler method, an implicit method, is not beneficial compared to the explicit forward Euler method, despite the gain in accuracy for large time steps.
Finally, we present Orchestral, a modular, distributed framework to bring together existing software specialized for specific scales. Using detailed particle simulations inside cells, we show that this framework does scale for a reasonable amount of cells on both high performance clusters and cloud computing platforms. Most notably, and to the best of our knowledge, this is one of the most detailed multiscale simulation involving both intracellular chemical kinetics and cell mechanics available in the literature.

All in all, this thesis introduces a new multiscale framework and reveals new insights regarding how various modeling approaches compare to one another. This thesis therefore contributes to solving some of the most pressing current challenges of multiscale modeling in biology.

Det finns många svårigheter när man simulerar modeller med flera nivåer, till exempel att bestämma vilka av komponenterna man måste använda på vilken nivå, eller för att kalibera dem mot experimentella data. Den här avhandlingen bidrar till att underlätta användningen av multiskalsimulationer av multicellulära system.


På vävnadsnivån utvecklar vi ett skalbart ramverk för att numeriskt studera så kallade center-based modeller. I artikel II använder vi det här ramverket för att studera hur valet av olika modellparametrar, numerisk metod och tidssteg påverkar simuleringar. Vi bevisar att stora tidssteg leder till icke-realistiska resultat, även när simuleringen är stabil, och att det största acceptabla tidssteget för realistiska simuleringar beror på valet av kraftfunktion. Vi bevisar också att Euler bakåt är mindre effektiv än Euler framåt, även om det är möjligt att ta större tidssteg.

Till slut presenterar vi Orchestral, ett modulärt, distribuerat ramverk för att sammanföra befintliga mjukvaror från olika modelleringsnivåer. Vi be-
visar att ramverket kan simulera en resonabel mängd av celler på högprestandadatorer och beräkningsmoln.

Allt som allt introducerar den här avhandlingen ett nytt multiskalramverk och leder till nya insikter om hur olika modelleringstrategier kan jämföras med varandra. Den här avhandlingen bidrar således till att lösa några viktiga utmaningar i området av multiskalmodellering i biologi.
Author’s Contributions

Paper I
The author of this thesis developed the Orchestral software. The ideas, computational experiments and the manuscript were developed in close collaboration between the two authors.

Paper II
The code as well as computational experiments were developed in close collaborations between the first two authors. The ideas and manuscript were developed in close collaboration between all the authors.

Paper III
The author of this thesis as well as the first author of this article shared the main responsibility of preparing the manuscript, running the computational experiments, and developing and maintaining the CBMOS software. The ideas were developed in close collaboration with the last author.

Paper IV
The author of this thesis performed the mathematical analysis and ran the computational experiments. The ideas were discussed by all three authors and the manuscript was prepared and revised in close collaboration between the author of this thesis and the last author.

Paper V
The author of this thesis designed the computational pipeline and ran the experiments. The ideas and manuscript were developed in close collaboration between all the authors.
Acknowledgments

Even though I am the main contributor of this thesis, this manuscript would not exist if it was not for the help I have received from all my great friends and colleagues throughout this adventure. This section is dedicated to these people.

Andreas, I am incredibly grateful for all the things I have learned with you during the last five years. Many thanks for your support in the darkest of times, thanks for always believing in me and thanks for always being there to save the day before arduous deadlines. Sonja, working with you has been a real delight. This thesis would not be half as good if it was not for your great scientific rigor and your continuous support. Thanks for being my friend, and all the best for your thesis and the end of your own journey. Many thanks to all my other collaborators, Stefan Hellander and Anass Bouchnita, and especially to Marc Sturrock and Prashant Singh, with whom I enjoyed fruitful collaboration and discussions on my last manuscript.

Next I would like to thank all current and former colleagues at the division of scientific computing, who contribute to making the atmosphere at TDB so unanimously friendly and inviting. Special thanks to Emanuel Rubensson, for bringing me back on track when I went astray and helping me recover the will and motivation to complete this journey.

Life would be boring without the greatest of friends. Many thanks to Jean-Baptiste Louvet and Sami Boukortt for the last ten plus years in your company. Thanks for all the time spent together, all the mutual inspiration, and the little breaks on steep alpine hiking trails. To all my other friends, you are not forgotten but you are too numerous to all be listed here, if you are reading these words you know who you are! Thank you! Thanks for all the good times both outdoors and indoors, and for everything I have learned from you all.

Finally, I would like to thank François and Cathy, my parents, Élëna, my beloved sister, and Papi, my grandfather, for their unconditional and indefectible support throughout the years and the cheese and chocolate packages that allowed me to make it through the last sprint. I love you!

This work was supported by the Swedish strategic research programme eSSENCE and the Swedish research council (VR) under award №2015-03964. The computations were performed on resources provided by the
Swedish National Infrastructure for Computing (SNIC) through the Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX) under Project №2018/8–157 and №2019/8–227. Generous travel grants from the Anna Maria Lundin Fund and the MATRIX institute made it possible to present my work and meet with the scientific community in various conferences and workshops all around the world.
References


A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title “Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology”.)