Systems Research Institute
Polish Academy of Sciences

Ph.D. Thesis

Maciej Cytowski
ICM, University of Warsaw

Large Scale Computational Modelling of Cellular Biosystems

Ph.D. Thesis Supervisor

Prof. dr hab. Marek Niezgódka

Ph.D. Thesis Auxiliary Supervisor

dr Zuzanna Szymańska

July 2014

Abstract

Systems Research Institute
Polish Academy of Sciences

Ph.D. Thesis

Large Scale Computational Modelling of Cellular Biosystems

by Maciej Cytowski
ICM, University of Warsaw

Biological processes are inherently very complex and involve many unknown relationships and mechanisms at different scales. Despite many efforts, one still cannot explain all the observed phenomena nor, if necessary, make any desirable changes in the dynamics. Recently, it has become apparent that the opportunity lies in complementing the traditional, heuristic experimental approach with mathematical modelling and computer simulations. Achieving a simulation scale that corresponds for instance to clinically detectable tumour sizes is still a huge challenge, however it is necessary to understand and control complex biological processes. In this thesis we present a novel large scale parallel computational model allowing 3-D simulations of cell colonies growing and interacting with variable environment in previously unavailable tissue scale. The cells are modeled as individuals located in the lattice-free 3-D space. The model incorporates cellular environment modelled in a continuous manner, mathematical description based on partial differential equations is formulated for selected important components of the environment. Discrete and continuous formulations are efficiently coupled in one model and allow considerations on different scales: sub-cellular, cellular and tissue scale. High parallel scalability achieved allows simulation of up to 10^9 individual cells. This large scale computational approach allows for simulations to be carried out over realistic spatial scales up to 1cm in size i.e. the tissue scale. For the first time a single model bridges the gap between intracellular dynamics and the tissue.

Acknowledgements

First and foremost I want to thank my advisors dr Zuzanna Szymańska and Prof. Marek Niezgódka for all their contributions of time, knowledge, ideas and motivation.

This work was supported by the National Science Centre Poland within grant no. DEC-2011/01/D/ST1/04133. All computations were carried out with the support of the "HPC Infrastructure for Grand Challenges of Science and Engineering" and "Centre for Preclinical Research and Technology" projects, both co-financed by the European Regional Development Fund under the Innovative Economy Operational Programme.

I would like to thank Bartosz Borucki, dr Krzysztof Nowiński and the whole ICM's visualization group for their invaluable help with visualization of computational results.

Contents

For Ania, Basia, Franek and the whole Family ...

Chapter 1

Introduction

1.1 Scope of the thesis

Experimental methods alone are very often not sufficient to build a consistent, systematic theory capable of describing biological phenomena. Exploring all the complexities of biological systems may require many experiments to be performed. Moreover, a full understanding of the governing mechanisms with the use of experimental methods alone is nearly impossible due to their highly nonlinear nature. One of the tools extensively used to complement the traditional, experimental approach is mathematical modelling, which aims at the mathematical representation of biological processes using a variety of analytical and computational techniques. Such models have proved to be very successful in many applications e.g. systems biology, bioinformatics, molecular dynamics, neurobiology, tissue and cell modelling.

The main objective of this thesis was to set up a methodology and powerful computational algorithms to address a range of biological phenomena in realistic, clinically detectable scale. As a result a novel, large scale computational approach was developed allowing for simulations to be carried out over spatial scales up to 1cm in size (more than 10^9 cells) i.e. the tissue scale. For the first time a single model bridges the gap between intracellular dynamics and the tissue. The proposed computational model is a general framework for large scale simulations of various biological processes occurring in cellular colonies, that can be further adapted to simulate particular processes such as solid tumour growth, biofilm development or vasculogenesis.

The mathematical model presented in this work is multiscale since it couples i) intracellular processes, ii) whole cellular colonies dynamics and iii) cellular environment.

With respect to i), at the cellular level, the key processes that are modelled are division, differentiation, apoptosis and interactions between cells. These processes are regulated by cascades of biochemical reactions called signalling pathways. Regulatory proteins, whose production is triggered by signalling pathways, initiate or modify the processes of cell division and death. For instance, the loss of control over cell division is linked to mutations of genes encoding proteins involved in the regulation of the cell cycle. As a result of these mutations, intracellular signalling pathways act in an altered way leading to further pathological changes in the organism (these pathways may become a therapeutic goal in the future). In turn, cell differentiation, both normal and pathological, influences the signalling pathway dynamics, which leads to subsequent changes at the cellular level. Taking into account the connections of the models at the cellular and intracellular scale is necessary in order to describe the function of cells.

With respect to ii), processes and dynamics of cellular biosystems are modelled in this work with individual-based approach. Biological cells are discrete entities and their precise spatial location within a cell colony has an impact on whether they grow and divide. They are influenced by signals from other cells, external factors such as nutrient concentration, stress and of course their own internal signalling pathways. Individual-based models proved to be very successful in modelling many biological phenomena related to cellular processes, since they analyse spatial and temporal dynamics at the level of the cell, linking individual behaviour with the macroscopic one. Individual-based approach provides scope to include different cell types, intracellular structures and processes and the spatial location of individual cells. Moreover, just like actual biological cells, each computational cell can have a stochastic nature.

Individual-based models can be lattice-based or lattice-free. Lattice-free models however, being more biologically realistic (since the cells are not constrained by a pre-defined underlying grid), are much more computationally expensive. The model presented in this work is of the lattice-free type and enables simulations to be carried out in 3-D.

With respect to iii), the cellular environment is modelled by continuous approach. Mathematical equations are formulated to describe external chemical fields and temperature. This is a multiscale modelling approach which is usually called hybrid since discrete and continuous descriptions are used to model biological phenomena at different scales. In such modelling approach, it is necessary to precisely define the coupling between scales. In addition, the computational methods used to solve discrete and continuous models are significantly different, and those differences are even more exposed in the case of the parallel computational approach.

Although mathematical modelling in biology led to the development of many computer simulation tools, none of these tools is capable of simulating cellular systems on the clinically detectable scale. We believe that modelling complex and multiscale biological processes that involve many unknown relations and mechanisms on such scales requires the development of high performance computational (HPC) tools. It is especially important for biological systems in which the properties of individual cells considered at extremely high resolutions can influence the dynamics and geometry of a large population of cells. The increasing need for accuracy has led to the development of highly complex models in the field of mathematical biology. However, most of the published models focus on theoretical modelling issues so to speak, neglecting the computational implementation corresponding to the real scale of the problem. In many cases, the correctness and usefulness of these models is assessed on the basis of only smaller example problems, due to high computational complexity. In this work, we do the opposite, i.e. we place the emphasis on the efficient implementation of the generic lattice-free individual-based model of cell colony dynamics rather than modelling particular biological processes. The ongoing research will be carried out on HPC systems. The computer code was developed and optimized on emerging HPC architectures and can be used to simulate biological processes on previously unachievable scales. The computational methodology described in this work, thanks to its wide and universal formulation and HPC approach, can be used in further development of existing mathematical models and their successful transition to higher, realistic scale. In fact, in our work we present examples of applications of our approach in simulations of phenomena such as tumour growth [1, 2].

1.2 Outline of the thesis

In Chapter 2, we present mathematical modelling background of the thesis. We cover the most important modelling approaches: continuum-level modelling in Section 2.1, discrete-level modelling in Section 2.2 and finally the hybrid modelling in Section 2.3.
The state of the art in High Performance Computing is presented in Chapter 3. We discuss current technology trends in Section 3.1 and describe the most important software challenges in Section 3.2.
In Chapter 4 we formulate our generic mathematical model of cellular biosystems. We describe the cellular system model in Section 4.1, the continuous global fields model in Section 4.2 and interior cell dynamics model in Section 4.3.
Main results of this thesis are presented in Chapter 5. Computational methods are covered in details in Section 5.1, with additional supplementary materials attached as two appendices A and B. Performance measurements of the resulting implementation together with usage details are presented in Section 5.2. Finally, in Section 5.3 we discuss how results of simulations can be analysed and visualized.

In Chapter 6 we present a number of reference applications of the model, especially those related to solid tumour growth modelling.
We finish the thesis in Chapter 7 by formulating conclusions and discussing very interesting future project.

Chapter 2

Mathematical Modelling Background

In recent years, modern biology and medicine have undergone (and are still undergoing) a genuine change from a descriptive, data-poor science to a data-rich, law-like behaviour science. At the same time, mathematical modelling has also a tremendous impact on our understanding of complex biological systems and became a very important complement to the traditional, heuristic experimental approach.

Mathematical modelling in biology (i.e. mathematical biology) aims at the mathematical representation of biological processes, using a variety of applied mathematical techniques. Assumptions on the mathematical nature of complex biological processes can be used to verify and understand the dynamics of those processes and their relationship with identified parameters. Such an approach is especially valuable when appropriate computational implementations of mathematical models and their results obtained by computer simulations are compared with experimental results. Further analysis may lead to coupled modelling where experiments and computer simulations refine and improve themselves by providing a feedback. Moreover mathematical modelling may involve specific aspects and phenomena that are not possible to be carried out during experiments in laboratories. As a result a completely new understanding of biological mechanisms may be formulated.

Mathematical modelling in biology is a highly interdisciplinary research procedure. Building a mathematical model which closely describes and simulates biological processes requires cooperation of a number of experts from different research fields: mathematics, biology, medicine, physics and computer science. In particular, the design and validation of models based on quantitative experimental data, designing new experiments, finding new mathematical methods allowing for an appropriate description of the process and its analysis and, finally, numerical simulation and optimization of treatment requires the cooperation of specialists in various fields. Progress in overcoming these barriers is already noticeable, the use of mathematical models in oncology is slowly becoming recognized as a research tool.

It is beyond the scope of this work to provide a detailed overview of mathematical models used in biology. However, in this chapter we briefly describe three most important approaches, namely: continuum, discrete and hybrid modelling. To put our work in appropriate context, the emphasis here will be placed on the most recent mathamatical models, their applications and highest achievable computing scale.

A very comprehensive overview of the multiscale modelling of biological systems is presented in [3]. Various models developed especially for the case of tumour growth are also described in [4].

2.1 Continuum-level modelling

Continuum methods are one of the most important modelling approaches at tissue scale. These methods are derived from principles of continuum mechanics to describe tissue components as continuous fields by means of differential equations. One of the first models of this type was presented by Greenspan in the mid-70's [5, 6]. This classical work introduced the model based on ordinary and partial differential equations describing development of cell cultures and growth of solid tumours. Recent models are usually based on a set of reaction-diffusion equations which describe the cell density [7, 8] and the cellular environment [9–12]. Depending on the specific biological phenomena modelling of the environment might include the following components: extracellular matrix, matrix-degrading enzymes, nutrients, oxygen or temperature. Among a variety of mathematical formulations used in continuum modelling of biological processes it is especially worth mentioning those based on level-set methods [13] and the multiphase approach [14].

The continuum modelling approach proved to be very useful in the modelling of the general tumour growth process [15–17]. Some models of this type take into account the heterogeneity of the tissue [18, 19], which is known to have a crucial impact on the processes of tumour development. Especially hypoxic microenvironment has been modelled in the number of papers resulting in a necrotic core occurring inside tumour mass [16, 20]. Hypoxic tumour cells induce new blood vessels to develop from existing vascular network in the lack of nutrients and oxygen. This process is known as tumour-induced angiogenesis and has been modelled in the number of continuous approaches, including the classical work by Balding and McElwain [21]. In some works, blood vessels are modelled by continuous equations describing vascular densities rather than vessel morphology [22, 23]. In other approaches vessels are represented as discrete line segments or continuous curves [24].

One of the main drawbacks of continuum modelling is the inability of controlling individual cell properties. Going down to the scale of a single cell is sometimes necessary when modelling certain biological phenomena (e.g. cancer which starts with mutations in one cell or a small group of cells). On the other hand model parameters in the continuum approach seem to be easier to obtain and control compared to the discrete case. Additionally there exist a number of fast numerical solvers which can be easily adapted to solve corresponding mathematical systems.

2.2 Discrete-level modelling

In recent years many individual-based models to study spatiotemporal dynamics of cellular biosystems have been developed. In this approach, individual cells are explicitly modelled and their individual properties are constantly updated according to a set of biophysical rules. Two main classes of discrete, or individual-based, models can be distinguished: lattice-based (cellular automata) and lattice-free. Further classification depends mainly on how a single cell, its shape, size and interactions are modelled. A very comprehensive overview of individual-based models was given by Drasdo *et al.* in [25]. Here we will list the most outstanding individual-based modelling approaches.

Cellular automaton models are still one of the most popular approaches used in mathematical biology studies. Cells are represented by discrete points assigned to lattice nodes. Very often variations in cell size and shape are not considered. However, cells can change their location in the lattice e.g. in the presence of a nutrient field (chemotactic movement). The main advantage of these algorithms is that they are relatively easy to implement due to their simplicity. Cellular automaton models have been extensively used in simulations of many aspects of tumour growth such as avascular tumour growth [26, 27], brain tumour growth [28], tumour invasion [29] and tumour angiogenesis [30].

Dormann and Deutsch [31] developed a lattice-gas cellular automaton model for simulations of avascular tumour spheroids. In this modelling approach, so called velocity channels are associated with each node of the lattice. Channels specify direction and velocity magnitude. Multiple channels per lattice node are allowed and therefore in lattice-gas models multiple cells can occupy a single lattice point. Lattice-gas cellular automaton model was recently used by Hatzikirou *et al.* to describe the glioma cell invasion process [32] as well as general interaction laws in cell populations [33, 34].

In the Glazier-Graner-Hogeweg (GGH) model, which was derived from Cellular Potts Model, each cell is treated individually and can occupy a finite set of lattice points. Therefore each cell has a finite volume and a deformable shape. The GGH model has many applications e.g. Graner and Glazier [35] simulated the cell sorting mechanism, Stott *et al.* [36] used the GGH model to simulate the growth of multicellular avascular tumours to the steady state and very recently Poplawski *et al.* [37] studied the morphological evolution of 2-D avascular tumours with the GGH model.

In lattice-free models cells are modelled as individual free moving entities and their position in space is not constrained to lattice nodes. Ramis-Conde *et al.* [38] developed a lattice-free model of the cancer cell invasion process. Interactions between cells are described with the use of potential function which incorporates both attractive and repulsive forces. This discrete approach was further developed into a hybrid model where additionally the cellular environment is modelled in a continuous way. Lattice-free methods are computationally much more complex than standard individual-based models like cellular automata. Computing interaction between cells usually requires the identification of neighbours for each cell separately, which is a computationally demanding task in the case of simulations of colonies with a large number of cells.

A highly detailed modelling approach was recently developed by Rejniak [39, 40], where each cell is modelled using the so called immersed boundary method. The boundary of each cell is represented as an elastic membrane. Adhesion forces are modelled as discrete linear springs between such realistic-shape cells. The model can also take into account external nutrient fields and extracellular liquid. It should be noted that this method is extremely computationally expensive which restricts its usability to case studies consisting of approximately 100 cells. The applications of the cell-immersed boundary method are, among others, studying of preinvasive intraductal tumours [41], simulations of formation of epithelial acini [42] and growth of avascular tumours [43].

The main advantage of individual-based methods is related to the possibility of translating detailed biological processes (e.g. cell-cycle phases, cell's signalling pathways) into dynamics and the development of cell populations or tissues. The main disadvantage is the computational cost which increases rapidly with the number of simulated cells. Lowengrub *et al.* state that currently individual-based simulations of a 1 mm^3 tumour spheroid consisting of several hundred thousand cells are not feasible due to very high computational complexity [4].

2.3 Hybrid modelling

One of the most important classes of models in mathematical biology and a very important research direction for the future are hybrid models. Discrete interactions between biological organisms cannot be captured in detail by the continuum approach and likewise global population mechanisms are very difficult to capture by the discrete approach. New models which address this issue combine discrete and continuum approaches to study biological phenomena that involve different scales. Tissue is represented as the sum of discrete cells while components of cellular environment (e.g. nutrients) are represented as continuum substrates. Hybrid modelling has many advantages over the traditional approaches since it has potential to combine their best features in a single model. It is known that this type of modelling has capabilities to simulate biological phenomena that cannot be captured by discrete and continuum approaches separately. Hybrid models have been widely used in mathematical oncology, many aspects of tumour growth such as early avascular tumour growth [44, 45], tumour invasion [38, 46, 47],or tumour angiogenesis [24, 48, 49].

Anderson in [50] showed that by using special microenvironmental properties and mutation algorithms within a hybrid model it is possible to prove that the dynamics of tumour invasion is directly correlated with the microenvironment in which the tumour grows. This result has important clinical implications.

It is possible to include some intracellular kinetics into each individual cell in hybrid approach. This is the main advantage of the hybrid over pure continuum approaches. Intracellular dynamics can be described with systems of ordinary differential equations (ODEs) and therefore the whole model can gain a genuine multiscale character. First work, where authors describe intracellular dynamics of proteins involved in regulation of adhesion can be found in [51]. A hybrid model can also include the consideration of fibres of the extracellular matrix. It is particularly important in case of solid tumour growth, cancer invasion and cellular motility phenomena. Preliminary works on this topic can be found in [52].

One of the biggest disadvantages of hybrid modelling is its very high computational complexity. For instance, studying a milimeter-sized tumour spheroid with classical hybrid approach requires a system with approximately 10^6 discrete cells. This is far beyond computational capabilities of currently known models. There are basically two ways to overcome this issue. First, some assumptions of the model can be simplified. Lowengrub *et al.* [4] list few examples of novel hybrid modelling where the tumour itself is described using both continuum and discrete approaches. For instance, Kim *et al.* [44] presented a hybrid model where only the so-called proliferating tumour rim was modelled by the discrete approach. On the other hand, the problem of high computational complexity of hybrid models can be addressed by selecting appropriate algorithms and by efficient implementation on high performance computing systems. This is the approach presented in this work.

Chapter 3

High Performance Computing: state of the art

Nowadays High Performance Computing (HPC) is one of the key forces driving the development of computational sciences and allowing to address the most challenging problems of science, engineering and industry. The continuing need for the even more detailed description of phenomena around us and increasing scale of mathematical models require the use of high-performance computers and tools and development of scalable algorithms and applications. This is particularly important in the case of scientific disciplines where "time-to-solution" is a key parameter. This includes time-critical decision making, real-time information processing, but also applications related to personalized medicine where a specific treatment or operation is to be planned based on the computational result.

We constantly observe very important breakthroughs both in computational sciences and supercomputer technologies. Especially, over the last 20 years, there has been a great progress in the field of software and algorithmic solutions in the field of HPC. A very good example are scientific libraries that implement the latest developments in the field of parallel algorithms for solving systems of linear algebraic equations which currently present high scalability on few hundreds of thousands of computer cores [53]. Although the progress in various fields was very valuable, a great deal of productivity has also been lost due to the lack of joint planning, coordination and proper integration of software solutions with the latest technology. While computing systems are becoming more and more complex, the efficient utilization of their theoretical peak performance by scientific computing is becoming more and more difficult. The largest supercomputers are currently built upon millions of cores and are very often equipped with additional hardware accelerators (e.g. Graphics Processing Units). As a result, there is a very narrow group of applications that can use the power of entire systems efficiently, with the most outstanding achievements in the category of high peak performance of real world applications being awarded every year by ACM Gordon Bell Prize [54]. Many of today's analyses and reports highlight that a significant development in the field of algorithms, programming languages and models as well as software solutions is yet to come in order to support the unprecedented technological aspects of the future exascale systems [55].

There is a very important and clear distinction between capability and capacity computing. The largest supercomputers based on emerging computer architectures aim primarily at capability computing, which can be thought of as using maximum CPU and memory resources jointly coupled with a high performance interconnect to solve a single extremely complex problem. Examples include scientific applications in the field of earth sciences, e.g. weather forecasting, where supercomputers are used to solve large scale problems in the shortest amount of time that no other computer is able to achieve. On the other hand, capacity computing is usually described as using a lot of cost-effective computational nodes to solve many smaller computational problems e.g. problems which can be solved with the use of a single computing node. Grid environments, which were very common in recent years, are best examples of capacity computing approaches. For the rest of this work, the term HPC will be used only in the context of capability computing.

The impact of modern HPC solutions is not only about making the same old computer codes and applications run faster. Much greater impact is in making completely new discoveries possible. One of the main objectives of this work was to enable simulations in the field of mathematical biology on previously unavailable scale through the use of novel HPC algorithms, solutions and technology. This chapter aims to present the main challenges of conducting computations on an extremely large scale. We describe current state and future predictions of HPC with special emphasis on technology and programmability. We identify the most important constraints that need to be put on an application in order to achieve high sustained performance on current peta- and future exascale systems. Based on these observations we have made appropriate decisions on the algorithmic and implementation approach used in this work.

3.1 Technology trends and their impact

Latest trends in the development of HPC are best shown by the ranking and comprehensive analysis available within the Top500.org [56], the biggest resource of historical and current knowledge of HPC technology. The list is published twice a year (in June and

November) since 1993 and gathers computers ranked by their performance on the Linpack Benchmark [57]. The list of awarded computing systems on the Top500.org can be easily analysed by grouping resources based on e.g. processor architecture, interconnect design, operating system or geographical location. In this section, we briefly describe the most important technological aspects of modern HPC technologies and current trends in their development. Our analysis is based on one of the latest Top500.org list published in November 2013. It should be noted that the Top500.org ranking changes very dynamically. The number 500 computing system in the list of November, 2013 was ranked as no. 363 in the previous edition of the ranking (June, 2013). The evolution of technology is also very dynamic, especially in the top ten of the list.

The Linpack Benchmark produces a single value - the average number of double precision floating point operations per second of computations, the measure of computer performance denoted by Flop/s. Currently, the largest supercomputers in the world have the theoretical peak performance of tens of peta Flop/s (or PFlop/s), where 1 PFlop/s = 10^{15} Flop/s. As petascale systems are currently deployed in many research centres and academia, the supercomputing world is getting prepared for the next level of performance: the exascale systems. Exascale will be very hard to achieve. Although the U.S. Department of Energy (DOE) has already announced a goal of building an exascale system in 2018, the number of technological challenges that need to be faced is still very high. Research centres and computer vendors all around the world have formed many consortia and projects to address the most urgent issues of this development, for example the International Exascale Software Project [55]. Predictions have been made for future technology of exascale systems and among other recognized parameters of those systems the most critical ones are the following:

- number of cores in the range of 100 million to 1 billion,
- clock rates limited to the range of $1 - 2$ GHz in order to meet energy efficiency requirements,
- highly multicore and multithreaded processors and hundreds of cores per die,
- hierarchical memory with capacity reaching 128 petabytes,
- hardware-supported thread context switching,
- enhanced fault tolerance allowing for dynamical reconfigurations of partitions,
- active power management to reduce the power consumption of currently unused cores,
- hardware supported synchronization mechanisms.

The exascale will be also very hard to achieve without a significant breakthrough in power consumption. One of the U.S. DOE studies [58] developed a strawman for an exascale system to be available in 2015. Among many important drawbacks the system designed in this study would have a power consumption close to 70 megawatts which might be available only in few national computer centres worldwide.

Multicore and multithreading

Since the early 1970s, we believed that the increase in capacity of microprocessors is following the legendary Moore's Law, which stated that the number of transistors that can be placed on an integrated circuit increases exponentially, doubling approximately every two years. For decades the clock rates of computer microprocessors were getting higher and higher. Computer scientists and enthusiasts created a number of Moore's Law variants. Some of them were addressing memory or the speed of the interconnect. Figure 3.1 shows a nice correlation between the world fastest known CPUs and the Moore's Law over the last decades.

It is expected that Moore's Law scaling in the number of transistors will continue only untill the end of the next decade and will be mainly related to the decrease in transistor size. However, as transistor sizes are getting smaller and smaller, laws of physics seem to dominate over the many years of Moore's Law theory and practice. The main reason for the change is related to the end of the so called *"Dennard scaling"*. Dennard in 1974 [60] first described the effect that as transistors got smaller and smaller the power density was constant. In other words, simplifying this observation, if the size of the transistor has been reduced twice, the power it used was reduced four times. Unfortunately this is no longer valid due to the static power losses which increased rapidly and are threatening the so called thermal runaway.

Modern HPC systems introduce parallelism on the chip level and rely on multicore and multithreaded architectures. According to the latest Top500.org statistics more than 75% of the list is occupied by systems which are built upon processor architectures with more than 6 cores per socket and this trend will continue. The most modern chips from IBM (Power8), Intel (Haswell) and AMD (Warsaw) are built of up to 12, 18 and 16 cores respectively. It is predicted that future exascale systems will be based on highly multicore processor architectures with hundreds of cores per socket. Another significant trend is associated with the introduction of support for additional hardware threads within a single computer core. This is particularly important in the case of the IBM Power Architecture family. Both Power7 and PowerA2 processors introduce a 4-way simultaneous multithreading mechanism (SMT) which allows four threads to execute

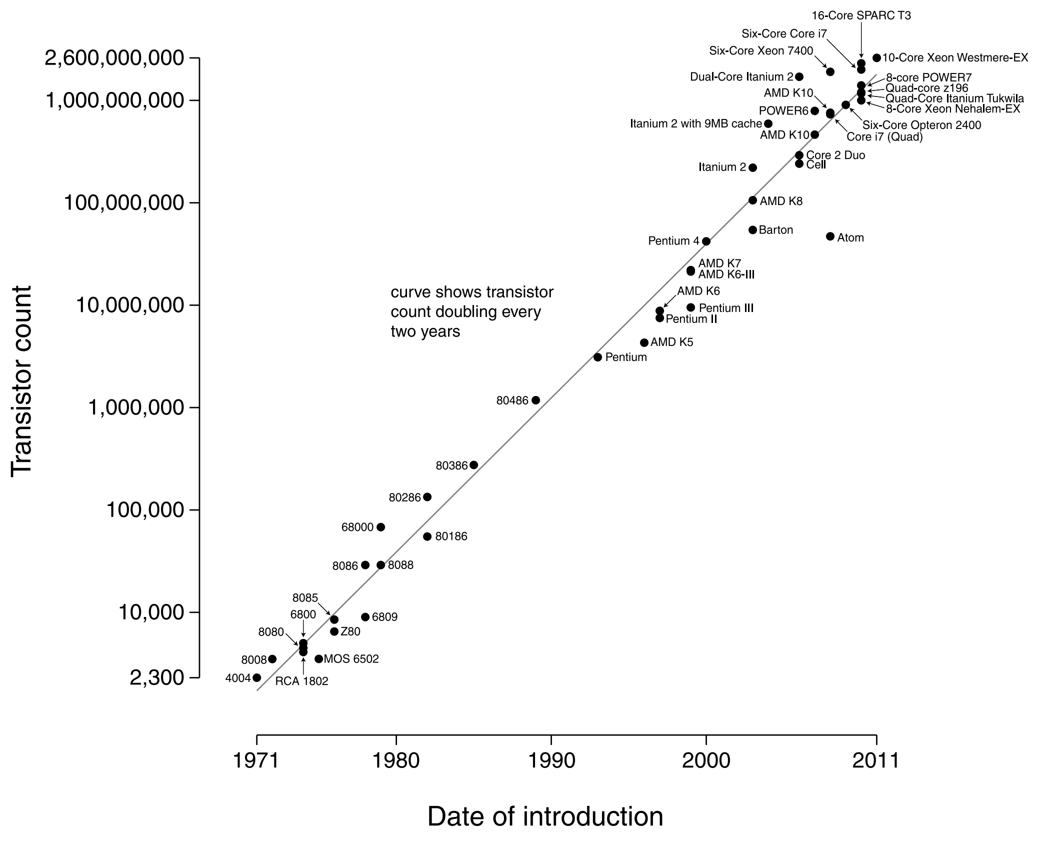


Figure 3.1: Plot of CPU transistor counts against dates of introduction. Logarithmic scale is used. Line represents the transistor count growth as given by the Moore'a law. (author: Wgsimon, license: CC-BY-SA 3.0) [59].

simultaneously on the same core. In fact, as we have shown in our previous work [61] [62] chosen applications and algorithms may benefit from using SMT mechanisms.

Hardware accelerators and heterogeneous architectures

There is a growing number of novel computer architectures based on modern co-processors or hardware accelerators, e.g. general purpose Graphical Processing Units (GPUs). Many HPC specialists claim that supercomputer architectures will have to be completely redesigned to enter the exascale era and that accelerator-based systems are one of the approaches towards it.

Approximately 10% of all computing systems on the latest Top500.org list are accelerator- or co-processor-based, with four such systems being in the top ten. The market share is consumed mainly by two major manufacturers, NVIDIA and Intel. One of the most recent products from NVIDIA - Tesla K40, provides an impressive peak double precision floating point performance of approximately 1.43 TFlop/s, which is beyond the reach of general purpose CPU architectures. However, such high performance comes at a price - the Tesla K40 GPU is built upon 2880 CUDA cores which introduces an additional level of massive parallelism within the computational nodes.

Another novel accelerator architecture is the Many Integrated Core (MIC) architecture introduced by Intel in 2010. Its current implementation, known as Intel Xeon Phi, has already been introduced in a few supercomputing sites, especially in the Top500.org number 1 system - Tianhe-2, which has 48,000 of MIC chips. The most powerful version of the Intel Xeon Phi is built of 61 compute cores with approximately 1.24 GHz clock speed. It supports up to 244 execution threads and has a theoretical performance of 1.2 Tflop/s.

Currently the most important bottleneck of accelerator-based technologies is related to low bandwidth between the accelerator's local memory and the main memory of the system. As a result, systems based on such devices present very small measured vs. peak performance ratios on the level of approximately 50%. One of the proposed solutions to this problem is to integrate, to the maximum possible extent, the accelerator chip with the main CPU and RAM memory of the node. Such solutions have already shown to be very promising in the case of the Cell architecture introduced in 2008 by IBM [63]. The RoadRunner supercomputer based on the PowerXCell8i architecture (a revised Cell processor variant with increased double precision performance) presented a rather high rate of 76% of theoretical peak performance achieved with the Linpack Benchmark. The Cell architecture was also a very significant attempt to address the problem of growing power consumption of HPC systems. Systems based on this architecture have been repeatedly ranked among the most energy efficient HPC systems within the Green500.org list [64] [65]. The Cell architecture is also an example of a promising hardware design that did not survive the test of time and did not have its successor. Among other things, this was related to the fact that the development of new and porting of existing software solutions for the Cell architecture was extremely complex. Programming tools and languages did not keep up with the revolutionary changes in the design of computer processors. Although the usefulness of such architectures for scientific computing has been shown in several cases [66] [67] [68] [69], deep development both in technology and programmability of such systems is yet to come.

Massive parallelism

General purpose CPUs and hardware accelerators are the basic building blocks of today's supercomputers. In modern HPC systems, the computational nodes are typically equipped with two to four processors, which in summary gives a rather high number of cores per node. Each of such computing nodes, depending on the design of the system, could be additionally equipped with an hardware accelerator. Currently the largest supercomputer in the world, the Tianhe-2 system installed at the National Supercomputer Center in Guangzhou (China), is built upon 16,000 computing nodes. Each node comprises two Intel Ivy Bridge Xeon processors and three Intel Xeon Phi accelerators, which sums up to 195 x86 cores within a node and 3.1 million computing cores in the whole system. This unprecedented high number of cores and heterogeneity of the system is extremely hard to use in an efficient way with the use of current computational algorithms, tools, programming languages and models.

It should be also noted that the heterogeneity is not necessary to build a petascale class supercomputer. The number 3 system on the Top500.org list is the Sequoia supercomputer based on the IBM Blue Gene/Q architecture [70]. The system is based on the PowerA2 processor which has 16+1 cores, with one core dedicated to operating system services and 16 cores intended for computations. The Sequoia system has more than 1.5 million cores and more than 20 PFlop/s of theoretical peak performance.

High performance interconnect mechanisms

The interconnectivity of computational nodes is an extremely important factor determining the scalability of scientific applications and algorithms at high scales. The statistics of the latest Top500.org list indicate that the most powerful systems are mainly based on the high speed custom interconnect or the high speed infiniband network (approximately 72% of the performance share in total). It is worth mentioning that the custom interconnect, which is mainly produced by companies like Cray, IBM or Fujitsu, is very often based on the torus network topology (3-D, 5-D or even 6-D). Other technologies like Gigabit or 10 Gigabit Ethernet are still very popular, but their practical use in scientific computing is significantly limited. More than 98% of Ethernet-based computers on the latest Top500.org list have been declared as industry segment systems. All of these systems are based on the Linux operating system and x86 architecture and represent rather simple cluster solutions dedicated to capacity computing.

3.2 Software challenges at extreme scales

Complex hardware design of current petascale systems and predictions for the technology evolution in the future exascale era have a significant impact on the development of completely new programming models and parallel paradigms for HPC. Without a fundamental change in many areas, existing software solutions will not be able to fully exploit the power of computational systems and in consequence provide an appropriate scale for the exploration of the greatest challenges of computational sciences. In this chapter we list the most important software challenges that need to be overcome as well as the few solutions that are already used with success on current large scale HPC systems.

Hybrid programming models

As the number of computational cores in HPC systems is continuously growing, programming techniques which expose the hierarchical structure of those systems are becoming more and more important. The mixed Message Passing Interface (MPI) [71] and OpenMP [72] programming model has become a technology standard and is a recommended parallel programming model for many of the leading HPC platforms including Cray XT and IBM Blue Gene families. Such an approach allows to greatly reduce the number of MPI processes involved in the calculations. This is known to be very important at extreme scales due to the difficulties in performing synchronization and collective communication between the large number of MPI processes [73]. Moreover, the usage of hardware threads support mechanisms like SMT is much more natural and efficient when a multithreading programming model is used within the computational node [74].

Introduction of accelerators and co-processors has also an important impact on the programmability and productivity. Applications developed for this type of architectures typically use vendor libraries (e.g. CUDA) to offload chosen compute-intensive algorithms on the accelerator. This produces another level of parallelism for the programming model and requires a significant amount of work for the programmer. During the last three years, new solutions that aim to create a common programming language for different types of accelerators have been proposed and developed. Among them, the most promising projects are the OpenCL [75] and OpenACC [76] initiatives.

Regardless of the chosen programming model, developing new applications for current heterogeneous computing systems is still very complex. A good practice is to design your application so that the most compute-intensive algorithms are located in separate modules. These modules should have a very simple form, so that it would be rather simple to rewrite them in any novel programming paradigm. On the other hand, the MPI library is still the first choice for implementation of communication schemes between nodes. The MPI standard is still developing and in its current version (v3.0) it addresses many aspects previously identified as bottlenecks for computations on extremely large numbers of cores.

Fault tollerance

The fault tolerance mechanisms are currently one of the major issues in HPC since it is anticipated that exascale systems will experience various kinds of faults many times per day. Failure rates reported today for large scale IBM Blue Gene/P installations are of the order of 0.001 failures per year per socket. Those measurements projected for much larger systems consisting of millions of sockets yields socket failure rates on the order of minutes. Since future HPC systems will be based on a number of such unreliable components, it will be extremely difficult to control and increase their reliability. This problem must be also addressed by operating systems, tools, libraries and even by scientific applications. In particular, the use of efficient and optimized checkpoint/restart mechanisms, which today is considered a good practice, will soon become absolutely necessary.

Novel algorithms

Exascale capability cannot be achieved without a significant redesign of operating systems, libraries, tools and most importantly software and computational algorithms [77][78]. Current software approaches as well as a large number of parallel algorithms will be inadequate in enabling future applications on exascale systems. For the very first time we will experience a technology evolution, which will require all algorithmic approached to be readdressed. First of all computational models must be redesigned to expose the locality and reduce the amount of communication. Especially extreme scalability requires the amount of synchronization between processes and implicit or explicit barriers to be reduced. Surprisingly the communication (i.e. on- and off-chip communication, inter-node communication) is the main source of energy consumption in large scale systems. On the other hand reducing communication is often seen as a code performance tuning issue but it should be considered as part of the algorithm design. Good examples of research on communication-specific algorithms are [79], [80] and [81].

Novel programming languages

Future algorithms and software will have to be tightly integrated with hardware through a rich set of hardware Application Programming Interfaces (e.g. to address the problem of resilience). Moreover computer systems will incorporate hardware accelerators alongside standard computational cores and algorithms will have to address the issue of the bottleneck introduced by the data transfer crossing the accelerator boundaries. This will result in prohibitively large costs of programmability. Therefore a significant development of computational languages and tools is highly desired. This issue was addressed by the DARPA High Productivity Computing Systems (HPCS) programme [82] launched in 2002 to offer funding for industry and academia to research the development of computing systems which focus on high productivity along with high performance. As a result a few completely new programming paradigms (e.g. Chapel, Fortress, X10) were designed [83].

Chapter 4

The Framework: Generic Model of Cellular Biosystems

4.1 Cellular System

Like many other physical phenomena, also biological processes can be simulated with the use of discrete representation based on interacting objects located in 3-D space. In particular individual-based lattice-based models are one of the most extensively used modelling tools [31]. Much more accurate lattice-free systems are considered to be too computationally expensive to handle even the basic biological properties and dynamics of cells in higher scales. However thanks to continuous development and enhancement of high performance computing it becomes possible to address the complexity and dynamics of very large biological colonies with the use of lattice-free models. In this section we present the main assumptions of our lattice-free model used to describe cellular colonies and their dynamics.

Cell description

We consider cells as free objects that reside in three-dimensional space. The number of cells in the model is referred to as NC. Each cell c_i position is described by its Cartesian coordinates $(x_{c_i}, y_{c_i}, z_{c_i})$. Each cell size and shape is described by a sphere centred at $(x_{c_i}, y_{c_i}, z_{c_i})$ with radius r_{c_i}. Both the position and size of a cell can change in time. We assume that everywhere in time all cells are located inside a 3-D box defined by their left-bottom-near point $(x_{min}, y_{min}, z_{min})$ and right-top-far point $(x_{max}, y_{max}, z_{max})$. If during simulation the cell is moved outside defined 3-D box as a result of force field computations, it is removed from the system and not considered in further computations.

Cells interact with each other. For every cell c_i its maximum intercellular interaction distance is defined by a neighbourhood sphere $B_{\epsilon_{ci}}(c_i)$ centred on $(x_{c_i}, y_{c_i}, z_{c_i})$ with a radius of ϵ_{ci}.

Cell cycle and mitosis

The growth of a population of cells is based on the growth and division of individual cells. In order to proliferate, a cell must undergo an ordered series of reactions that allow for the duplication of its contents and finally the splitting into two new daughter cells. These processes of duplication and division are called, collectively, the cell division cycle (or simply the cell cycle), see Fig. 4.1.

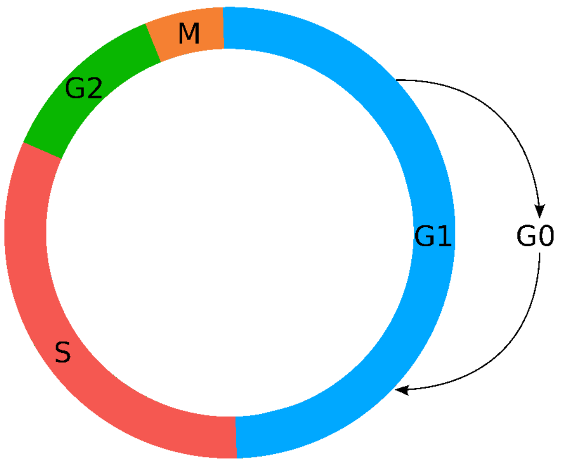


FIGURE 4.1: Schematic illustration of the eukaryotic cell cycle.

The eukaryotic cell cycle is divided into four phases: M-phase, when the actual division of the cell occurs, G_1-phase, which is a phase of cell growth that occurs after the end of the division, and prior to DNA duplication, S-phase, when nuclear DNA is duplicated, and G_2-phase, when a cell completes preparation for cell division (see Fig. 4.2).

The G_1-phase allows additional time for a cell to grow and to monitor its environment. If conditions are particularly unfavourable, instead of entering S-phase a cell can enter a resting state - G_0-phase, where it remains until conditions improve and it continues the cell cycle. The first cell cycle checkpoint, so called G_1/S checkpoint, is located at the end of the G_1-phase, when a cell takes a decisive step to either enter S-phase and to initiate DNA duplication, or to undergo apoptosis (i.e. to die).

During G_2-phase the synthesis of proteins needed to carry out cell division occurs. This also provides a safety gap, allowing the cell to ensure that DNA duplication is completed before mitosis [84]. The second checkpoint, so called G_2/M checkpoint, is located in the

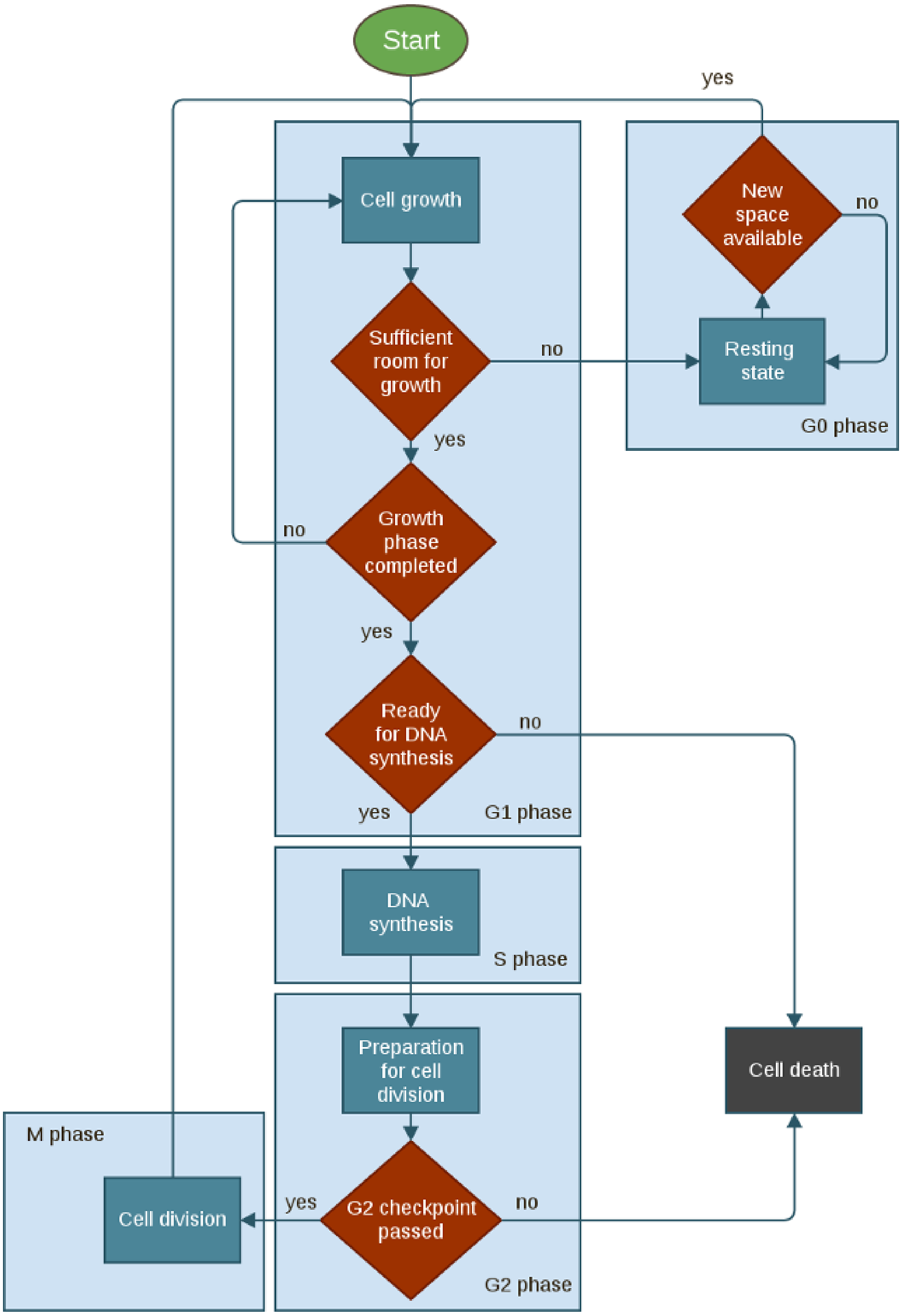


FIGURE 4.2: Sequence of events leading to cell division. Decision points: *"Sufficient room for growth"*, *"Growth phase completed"* and *"Ready for DNA synthesis"* corresponds to the G_1/S checkpoint, decision point *"G2 checkpoint passed"* corresponds to the G_2/M checkpoint. Decision Point *"New space available"* corresponds to the cell decision whether it can re-enter the G_1 phase.

G_2-phase. Passing through this checkpoint, a cell decides whether it is ready to enter M-phase and if not it undergoes apoptosis. During M-phase, at first, the cell's nucleus divides into two (a process known as mitosis) and later the whole cell splits into two (a process known as cytokinesis).

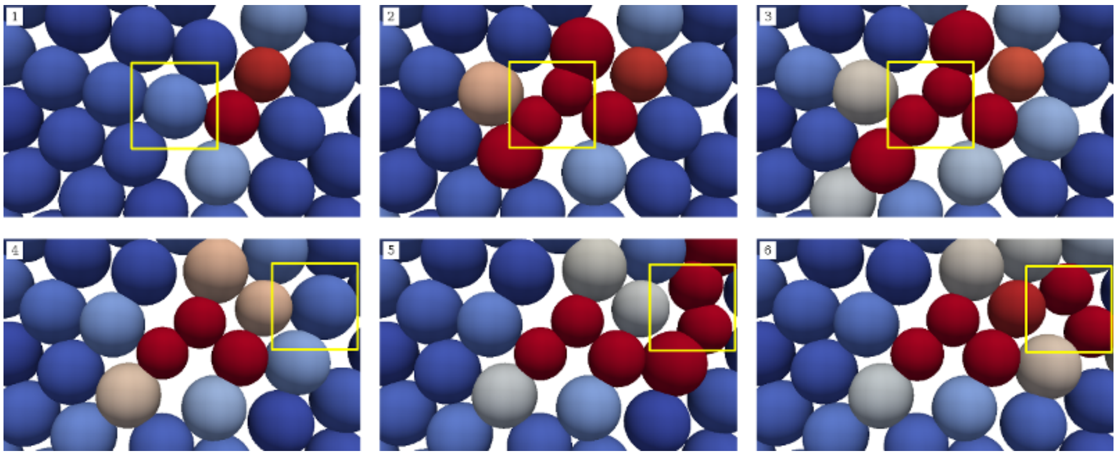


Figure 4.3: Figures showing in detail the process of mitosis in two given cells (indicated by a yellow frame) each dividing into two daughter cells in a cellular colony.

We assume that during cytokinesis the parent cell c_i of radius r_{c_i} splits into two daughter cells of radius $\frac{1}{\sqrt[3]{2}}r_{c_i}$ (symmetric division), chosen to preserve mass conservation. Daughter cells are located at a distance of $\frac{r_{c_i}}{2}$ from the center of the parent cell in a randomly chosen direction (see Fig. 4.3). At any time, each cell of the model can be in one of the different states: G_1-phase, S-phase, G_2-phase, M-phase or G_0-phase. The exact length of each cell cycle phase for a respective cell is taken randomly from the interval $[T \cdot (1-V), T \cdot (1+V)]$, where T is the average length of the phase, and V is the variance. Each cell during the simulation has a certain probability P_{c_i} of being marked for programmed cell death (apoptosis) at the nearest checkpoint. The exact value of P_{c_i} depends on the particular cell type, e.g. in the example simulation presented in this work we assume that tumour cells have down-regulated their apoptotic pathways (i.e. they do not undergo apoptosis).

Living cells continuously interact with their environment. In our model, at each time step and regardless of the current cell cycle phase, the cell samples its own environment and if the conditions are appropriate the cell interact with its environment and continues its life cycle until the next time step. If the conditions become unfavourable and the cell is not able to pass through the cell cycle, the cell ceases proliferation and becomes quiescent. Further deterioration of conditions leads to cell death.

Cell-cell interaction

The interactions between cells in the system are described by a modified Hertz model as proposed in [85]. A decreasing distance between the centres of cells results in an attractive interaction related to adhesive forces. Furthermore, experiments suggest that cells have only limited compressibility which gives rise to repulsive interaction. The potential $V_{c_i c_j}$ between two adjacent cells c_i and c_j of radii r_{c_i} and r_{c_j} in this model is a combination of repulsive and attractive forces and is given by:

$$V_{c_i c_j} = \underbrace{(r_{c_i} + r_{c_j} - d_{c_i c_j})^{\frac{5}{2}} \frac{1}{5E_{c_i c_j}} \sqrt{\frac{r_{c_i} r_{c_j}}{r_{c_i} + r_{c_j}}}}_{repulsive} + \underbrace{A_{c_i c_j}}_{adhesive}$$

The first term of the above formula relates to repulsive interactions modelled with the Hertz formula. Here $d_{c_i c_j}$ is the distance between two cells and $E_{c_i c_j}$ is computed with the use of Young moduli E_{c_i} and E_{c_j} and Poisson ratios ν_{c_i} and ν_{c_j}. The parameter values of Young moduli for each cell were randomly chosen from the range of 2100 ± 100 Pa in accordance with [86]. The Poisson ratio is assumed to be equal to 0.33 as suggested in [87].

The second term $A_{c_i c_j}$ of the potential formula relates to the adhesive forces and is given by:

$$A_{c_i c_j} = \rho_m D_{c_i c_j} 25 k_B T,$$

where k_B is the Boltzmann constant, T is the temperature, $D_{c_i c_j}$ is the contact area between cells c_i, c_j and ρ_m is the density of surface adhesion molecules in the contact zone [88], which we assume is given for a specific cell type.

The potential function value for cell c_i is them given by the sum:

$$V_{c_i} = \sum_{c_j \in B_{\epsilon_{ci}}(c_i)} V_{c_i c_j}. \tag{4.1}$$

Cell movements

The motion D_{c_i} of cell c_i is based only on the potential function. The aforementioned formulation of the potential function allows us to model two very important types of cell-cell interactions. First of all adhesive forces which allow cells to be bound to other cells or extracellular components of tissue (so called extracellular matrix - ECM). Secondly repulsive forces which appear when cell cytoskeletons and membranes are significantly stressed due to attractive forces and the decreasing distance between cells' centres.

Cells can change their position after each simulation step. The displacement velocity vector is computed by deriving the potential function, i.e.:

$$D_{c_i} = -\nabla V_{c_i}.$$

The other mechanisms which might affect the dynamics of cell colonies (e.g. chemotaxis, haptotaxis) are not currently treated within the present model. In the general case cells will additionally tend to move with respect to gradient of substances in their environment, see [89] for 2-D example.

4.2 Global Fields

We consider not only the dynamics of the cells themselves, but also its mutual influence on the environment, modelled in a continuous manner. Therefore, the previously described discrete model of cellular dynamics is extended with the mathematical description of the cellular environment. First, we introduce a general equation governing the external field in the cells' environment:

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - \lambda u + f - g. \tag{4.2}$$

Here $u(\mathbf{x}, t)$ denotes concentration of the field (e.g. nutrient, oxygen, temperature) whereas f and g are source and sink/uptake functions of the external field by the individual cells. Both functions f and g are calculated based on cell density approximated at each node of the discretization grid. In this way we obtain a coupling of the discrete model describing the dynamics of cellular colonies and the continuous model describing the environment. Implementation details are described in Section 5.1.2.

The above Equation 4.2 is used to model global nutrient and chemical fields e.g. oxygen, glucose and hydrogen ion similarly as it was presented in works by Gerlee *et al.* [90] [47]. In such case Equation 4.2 is equipped with fixed value Dirichlet boundary conditions and initial condition defined by the constant function corresponding to average concentration of a given field in healthy tissue. In the case of the oxygen field we additionally assume that $\lambda = 0$ and $f = 0$.

For applications related to the hyperthermia treatment we can also model the temperature global field. In such case we impose different boundary conditions: Robin boundary conditions on five faces of the cubic domain and fixed value Dirichlet boundary conditions with temperature set to 42 degrees Celsius. As for the initial condition, we set the initial temperature in the tissue to 36 degrees Celsius. Such a setting corresponds to the case where the tissue is heated from one chosen side. Also, in the case of temperature global field we assume that $\lambda = 0$, $f = 0$ and $g = 0$. Example results of temperature field computations are presented in Figure 4.4.

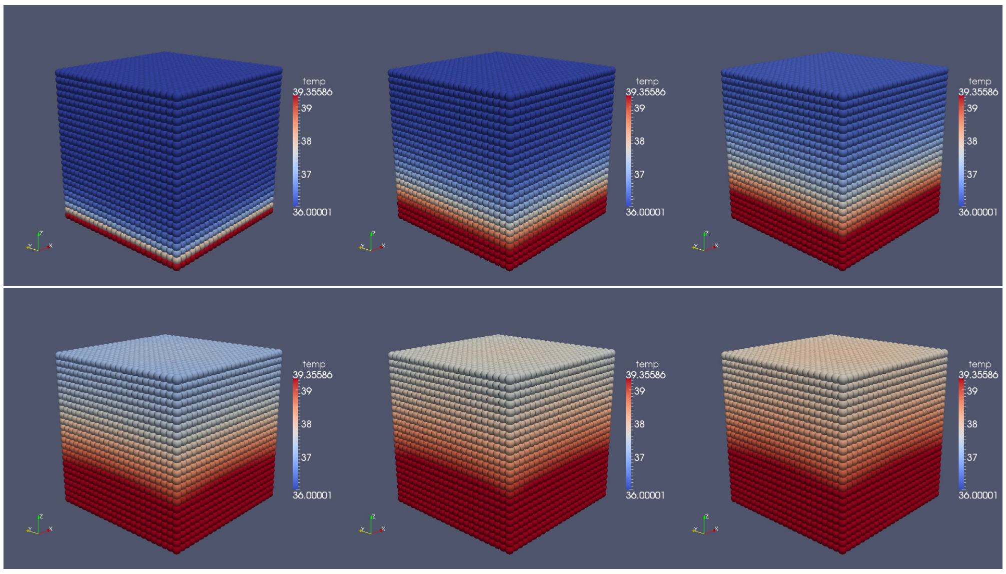


FIGURE 4.4: Example results of temperature global field simulated with specific boundary conditions imposed to model the hyperthermia treatment.

4.3 Interior Cell Dynamics

The key feature of the model is its multiscale character - in addition to modelling individual cells we also model certain aspects of their intracellular dynamics. In particular, the model is constructed in such a way that it allows for the examination of the influence of intracellular processes on the dynamics of the whole cell population.

The evolution of a cell precisely depends on the expression of genes contained in its nucleus. In response to changes in the external or internal cellular environment, the expression of particular genes and consequently protein synthesis changes [91]. Signalling pathways can be seen as intracellular cascades of biochemical reactions that transmit molecular signals from receptors on the cellular membrane to the interior of the cell up to the cell nucleus. Inside the nucleus the signalling pathways influence gene expression constituting natural regulatory systems that, on one hand, ensure cell resistance to random changes in its condition (i.e. preserve cell homeostasis), and on the other, ensure a proper response to external forcing (i.e. a direct response to the environment) [92]. A schematic diagram of the signalling pathway is shown in Figure 4.5.

In many diseases signalling pathways act in a distorted way, leading to pathological changes. An example of a disease, which is highly multi-scale by nature is cancer. It begins with changes in the genome, resulting in impaired functioning of the signalling pathways, which in turn is the cause of an impaired processes at the cellular level such as the processes of proliferation or apoptosis (programmed cell death). This in turn leads to the formation of structures such as solid tumours that affect functioning of tissue and organs. Initially in the cells, later in the tissues, organs, and finally they may may disturb the functioning of the entire body. Therefore influencing the dynamics of signalling pathways is a promising therapeutic objective.

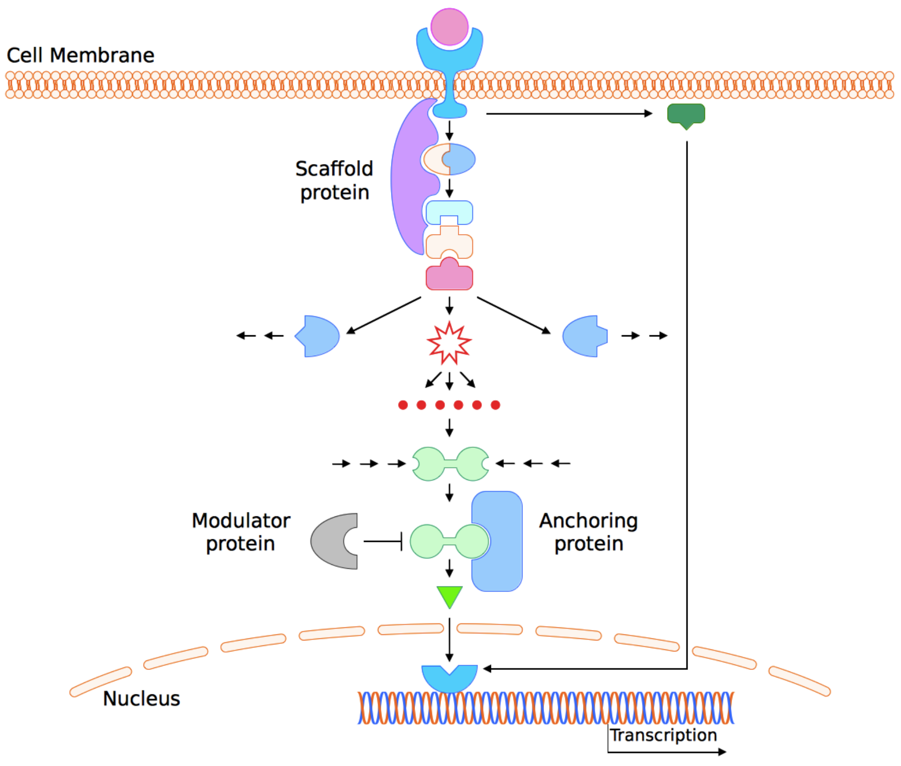


Figure 4.5: A signaling pathway from a cell surface receptor to the nucleus.

From the modelling point of view, signalling pathways are usually described with use of large systems of ordinary differential equations (ODE). To solve such systems one can relatively easily implement for instance the fourth order Runge-Kutta algorithm or use one of many available solvers. However to investigate the mutual influence of intracellular protein concentrations and cellular colony dynamics one has to combine the description on intracellular level with the description on cellular one. If the approach adopted for the description of a cell colony is continuous, i.e. with use of partial differential equations or integro-differential equations, then the coupling of models at intracellular and cellular scales causes a lot of difficulties, especially if the intracellular protein synthesis is dependent on the cell cycle phase. In this context, a hybrid approach that allows to assign into each cell its independent model of signalling pathways provides an ideal solution to this problem and the optimal framework for such multi-scale modelling (see Figure 4.6).

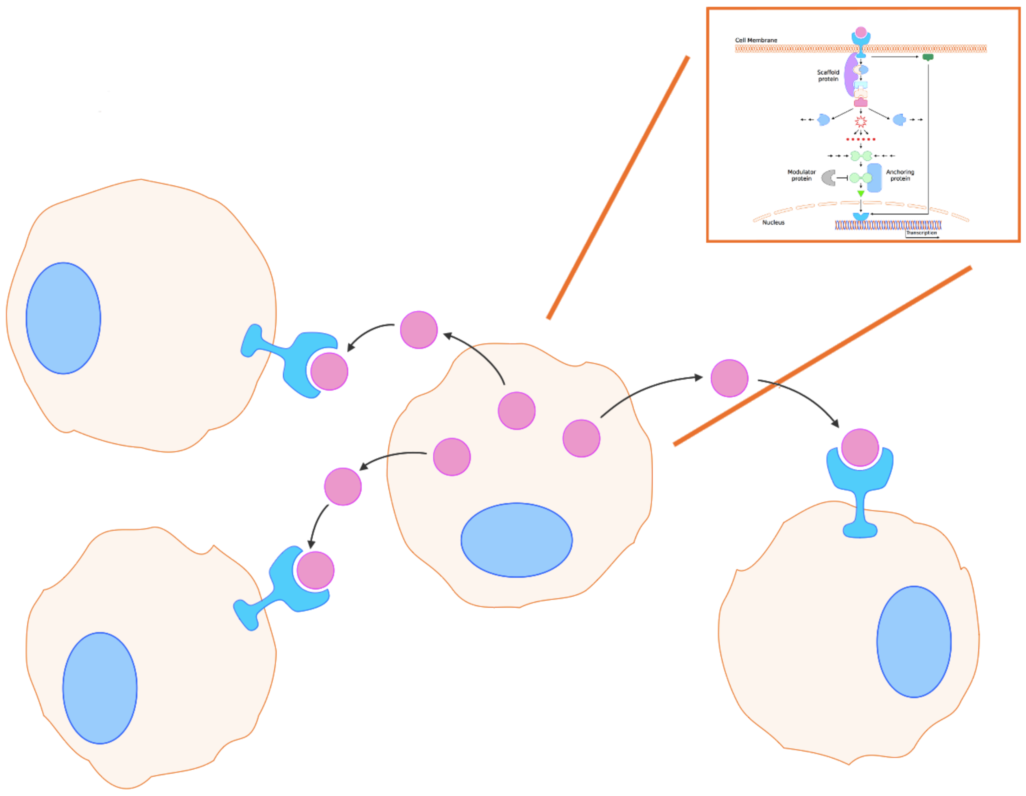


FIGURE 4.6: Schematic diagram of multiscale model of cellular interactions. The model consists of i) intracellular signalling pathway that governs the production and secretion of signalling molecules (pink dots) ii) cellular individual based model.

Chapter 5

Main Results

5.1 Computational methods

This section presents details of the computational methods that we have used to translate the mathematical model into computer language. Large scale simulations at tissue scale are both highly memory- and compute-intensive. Therefore the proposed computational framework was designed to work on novel supercomputing architectures and parallelized from scratch. Computational algorithms have been carefully selected to ensure high accuracy on one hand and efficient parallelization and high scalability on the other. Performance of the resulting computer model was tested in various set-ups, e.g. for different domain decomposition scenarios. Consecutive steps of implementation, performance analysis and optimization led us to the development of a highly scalable simulation environment. Implementation details and basic usage information are also presented here.

A schematic simulation diagram is presented in Scheme 1. Simulations are made up of successive iterations, which computational nature will be described next.

5.1.1 Computational methods for cellular system

Domain decomposition

Each iteration of the simulation begins with the domain decomposition step (Step 1, Scheme 1) which distributes cells across available parallel processes. We have selected the most appropriate decomposition method based on the analysis of resulting load balancing. In our simulations we use two dynamical decomposition algorithms interchangeably: the Recursive Coordinate Bisection (RCB) method and the Peano-Hilbert

Scheme 1 Computational scheme of the simulation

```
$iter \leftarrow 0$
while $iter \leq max\_iter$ do
  Step 1: Perform domain decomposition
  Step 2: Build tree
  Step 3a: Find exchange regions and initiate data exchange
  for all local cells do
    Step 4a: Find cell's neighbours ← local data
    Step 5a: Compute potential and density functions ← local data
  end for
  Step 3b: Wait until data exchange is finished
  for all local cells do
    Step 4b: Find cell's neighbours ← remote data
    Step 5b: Compute potential and density functions ← remote data
  end for
  Step 6: Interpolate cells to global fields grid
  Step 7: Compute global fields
  Step 8: Interpolate global fields to cells
  for all local cells do
    Step 9: Update cells' cycle
    Step 10: Compute forces and move cells to their new positions
  end for
  Step 11: Output results and statistics
end while
```

Space Filling Curves (HSFC) method. Both decomposition mechanisms are particularly useful in the case of systems of free moving objects.

One of the most important features of both methods is that decomposition domains assigned to parallel processes are topologically connected. The amount of communication between adjacent processes is thereby minimized and most of the calculations can be performed independently. Data exchange regions, which are also borders of domain parts assigned to different processes, are illustrated in Figure 5.1.

Nearest neighbours searching and potential function

Computations of interactions between lattice-free cells are very complex and time consuming since for each cell all of its neighbour cells need to be determined. However there exist a number of methods developed especially for so called many- or N-body problems, where the physical phenomena are described by objects in 3-D space interacting with each other [93]. Mathematically, the N-body problem can be formulated as follows:

$$V(p_i) = \sum_j F(p_i, p_j) \tag{5.1}$$

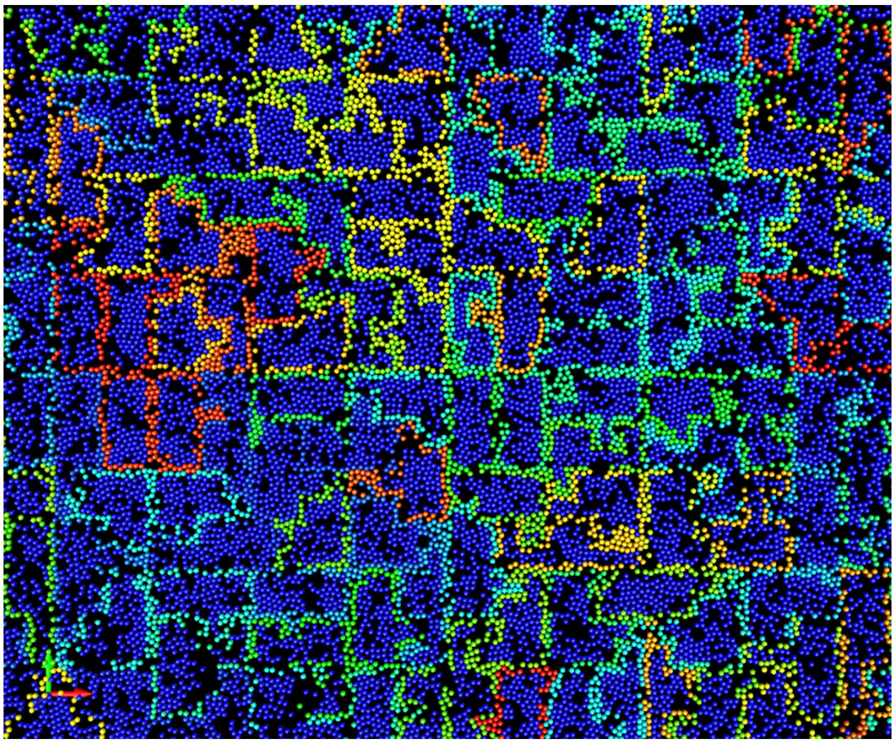


FIGURE 5.1: Halo exchange areas shown with colours in a 2-D simulation of a cellular colony on 256 parallel MPI processes. Local cells of each process are denoted by a navy blue colour. The colours of halo exchange cells has been randomly chosen for each process.

where $V(p_i)$ is a physical quantity at object (or cell) p_i which is obtained by summing the pairwise interactions $F(p_i, p_j)$. Examples include the gravitational forces and relations among the stars in a galaxy [94] [95] or the Coulomb forces exerted by the atoms in a molecule [96]. As can be seen, Equation 4.1 which is the formula for computing the potential function V_{c_i} for a given cell c_i by summing the pairwise bonding forces between cells is a special case of the above Equation 5.1.

In Appendix A the most popular and appropriate methods for solving the N-body problems are described. All of those methods were developed for physical phenomena where each of the objects interacts not only with its direct neighbours (short-range interactions) but also with particles that are distant (long-range interactions). As it was described in Subsection 4.1 our model assumes only a finite neighbourhood defined by a sphere around the center of each cell. In this formulation most of the computational methods developed for N-body problems can be modified to capture only short-range interactions between cells. Furthermore methods which deal mostly with the long-range forces approximation, like the Particle-Mesh algorithm [93], are inadequate. Here we describe the tree-based algorithm developed to compute the potential function on the considered domain. We present and discuss the performance comparison of the original algorithm with two other methods.

The Barnes-Hut algorithm (see Appendix A) is most often used for models which involve force computations defined by the laws of physics and dependent on the individual particles' masses. In the general case it is assumed that force calculations for a given physical object are the result of summarized interactions with all other objects in the system. There is a class of algorithms in which long-range forces are numerically approximated and not computed directly, for instance the Barnes-Hut algorithm belongs to this class. In the case of the N-body problem described in this work, long-range interactions can be ignored since the potential of each cell c_i depends only on interactions with neighbours contained in the cell's finite neighbourhood $B_{\epsilon_{ci}}(c_i)$. This observation is the basis of the Reduced Octree method that we propose in this section. In the following formulation of the algorithm we use the term cells instead of objects in order to refer to specific application of the method.

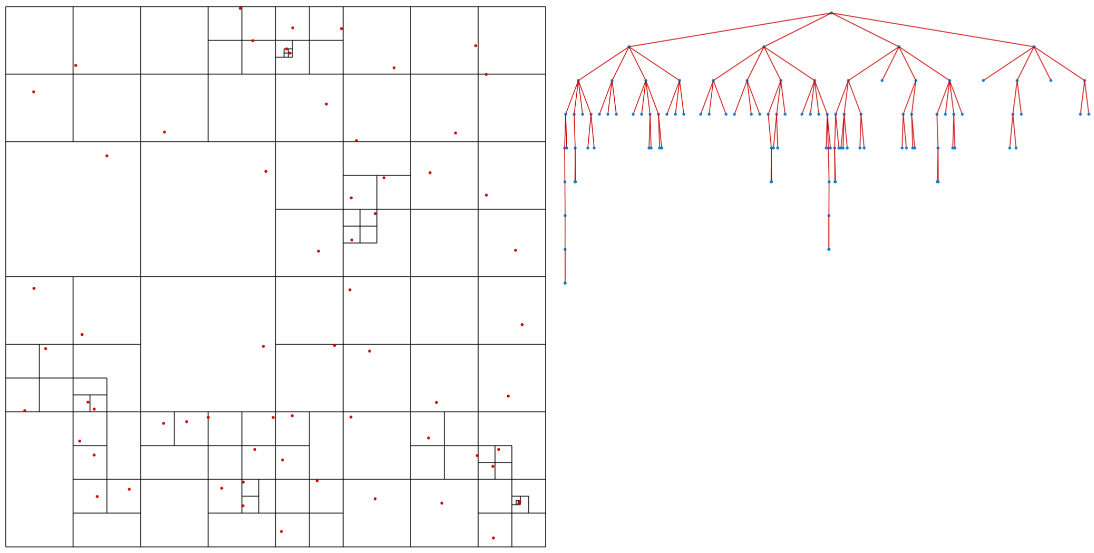


FIGURE 5.2: Example of a quadtree generated for 64 randomly distributed cells on 2-D plane. Tree structures generated during real 3-D simulations are much more complex.

The very first step of our method is the octree construction algorithm very similar to the one proposed in Barnes-Hut algorithm (described in Appendix A). Cells are organized in a tree structure with respect to their position in the 3-D box. The individual sizes of the cells are omitted in the tree construction step, the tree is constructed based on the Cartesian coordinates of the cells' centres. We begin with a cube in the 3-D space which is a root of the tree and encloses all cells in the simulated tissue. The cube is divided into eight smaller cubes of $\frac{1}{8}$ the area which are called octants. Each octant can be further broken into eight cubes and so on. Octants are divided repeatedly until they enclose more than one cell. Therefore, all leaves of the resulting octree enclose only a single cell. An example quadtree construction in the 2-D case is schematically presented in Figure 5.2. The main modification implemented in the tree construction step is that we are using an additional array of pointers in order to keep track of each leaf node in the tree and its corresponding cell number. This is easily achieved by modification to the InsertParticle procedure known from the formulation of Barnes-Hut algorithm. The resulting BuildOctree and InsertParticle procedures are presented in Scheme 2 and Scheme 3).

Scheme 2 BuildOctreeNew

```
1: $root = empty\ node$
2: for $i = 1 \to NC$ do
3:     InsertParticle(i,root)
4: end for
5: Eliminate empty leaves by traversing the tree
```

Scheme 3 InsertParticleNew(i,n)

```
1: if n is not a leaf then
2:     determine in which child c of node n the particle i resides
3:     InsertParticleNew(i,c)
4: else
5:     if n is a leaf and is full then
6:         add eight childern of node n
7:         move particles that were located at n into appropriate child nodes
8:         update leaf pointers for each of the moved particles
9:         determine in which child c of node n the particle i resides
10:        InsertParticleNew(i,c)
11:    else
12:        if n is a leaf and is not full then
13:            store particle i in node n
14:            leaf[i] points to the node n
15:        end if
16:    end if
17: end if
```

The new structure of the algorithm presented in Scheme 3 with modifications indicated in the pseudo code (red font) can be used for efficient potential function computations, as proposed next. The main advantage of having an additional array $leaf[i]$ with pointers to leaf nodes is that now the potential function can be computed by traversing the tree in a bottom-up fashion, in contrast to top-down fashion present in the classical formulation of the octree algorithms. The tree for each cell c_i is traversed starting from the leaf $n \to leaf[i]$ in which the cell resides. In each consecutive step we are examining the neighbourhood of a given cell c_i by computing the intersection of its neighbourhood $B_{\epsilon_i}(c_i))$ and the cubes $C(s)$ related to each of siblings s of node n of the tree. If the intersection is not empty we are opening a new path and performing a top-down tree traversing on the subtree rooted at the sibling s. This is achieved by calling the procedure $SubtreePotential(i, n)$. When each of the siblings was examined and all of its potential function contributions have been computed the bottom-up traversal is recursively continued by proceeding to the father f of node n. The whole procedure is then repeated for $n \to father$. We stop the traversal when we reach the root node or when the neighbourhood of cell c_i is fully a subset of cube $C(n)$. Described method for computing the potential of cells can be formulated in a form presented in Schemes 4 and 5.

Scheme 4 TreePotential

```
Procedure TreePotential
  for $i = 1 \to NC$ do
    $n = leaf[i]$
    while 1 do
      for each sibling s of node n do
        if $(C(s) \cap B_{\epsilon_i}(c_i))$ is not empty then
          SubtreePotential(i,s)
        end if
      end for
      $n = n \to father$
      if $n \to father == NULL$ then
        break
      end if
      if $B_{\epsilon_i}(c_i) \subset C(n)$ then
        break
      end if
    end while
  end for
```

Scheme 5 SubtreePotential(i,n)

```
  if n is a leaf then
    compute bonds F directly
    return
  end if
  for each nonempty child k of node n do
    if $B_{\epsilon_i}(c_i) \cap C(k) = 0$ then
      continue
    end if
    if k is a leaf then
      compute bonds F directly
    else
      SubtreePotential(i,k)
    end if
  end for
```

We have compared the performance of three algorithms for potential function computations: the naive Particle-Particle method, the classical Octree method and Reduced Octree method presented in this work. The measurements were made for two sizes of cellular colonies with number of cells $NC = 2^{20}$ and $NC = 2^{24}$ on a single processor machine. Cell positions were randomly generated. It is obvious that the initial distribution of cells has an impact on the tree structure and therefore might have an impact on the performance of tree-based algorithms. Therefore, we decided to evaluate the performance for two different random distributions:

- a uniform distribution generated with the use of the SPRNG library [97],
- and a Gaussian distribution generated with the use of the SPRNG library equipped with the Box-Muller transform [98].

For both test cases an appropriate neighbourhood radius ϵ has been chosen. The ϵ parameter has influence on the performance of the computational algorithms, values proposed for benchmarking came from real simulations.

As we can see in Table 5.1 the Particle-Particle method is extremely slow when compared with tree-based algorithms. Therefore we have skipped the performance measurements for the bigger test case. The main difference between the classical Octree algorithm and the Reduced Octree method is the direction of tree traversing. Thanks to bottom-up traversing introduced in the later one we are able to check whether the cell's neighbourhood is fully enclosed in the currently considered node. That enables us to reduce the search procedure and consequently the number of collision detections performed on the cells' neighbourhood and tree nodes. We have measured two orders of magnitude less collision detections for the Reduced Octree method for the bigger test case.

Algorithm	$NC = 2^{20}, \epsilon = 0.05$		$NC = 2^{24}, \epsilon = 0.001$	
	Uniform	Box-Muller	Uniform	Box-Muller
Reduced Octree	2.47s	9.03s	34.13s	39.17s
Classical Octree	5.81s	13.62s	132.42s	145.55s
Particle-Particle	9435.40s	8897.66s	N/A	N/A

TABLE 5.1: Performance comparison of three algorithms for different problem sizes and initial random distributions.

Communication and data exchange

One of the consequences of a parallel implementation of the model is the need for communication and data exchange, since neighbours of boundary cells might be stored on different processes. As can be seen from Scheme 1, potential and density function computations, which are based on a tree traversing algorithm, are divided into two stages. Data exchange between processes is initiated (Step 3a, Scheme 1) before the first stage of calculations which is completed with the use of locally available data (Steps 4a and 5a, Scheme 1). The second stage (Steps 4b and 5b, Scheme 1) is computed when the remote data packages, or messages, from other processes are received which is ensured in Step 3b, Scheme 1. This is achieved with the use of non-blocking communication mechanism available in the MPI library. With this approach, we are able to overlap computations and communication, which has an important impact on final scalability of the application.

Density function

As it was shown, the value of the potential function can be determined based on the information about the neighbours of each cell. At this stage of the iteration we are also able to calculate the approximate value of the density function for each cell. This is achieved by adopting the concept of Smooth Particle Hydrodynamics (SPH) [99]. The SPH algorithm was originally developed for fluid dynamics problems, however nowadays it is successfully used in many areas of computational sciences. More information on the SPH method can be found in Appendix B.

We assume that the value of the density function $\tilde{\rho}$ for given position $\vec{p}$ in three-dimensional space can be computed as a sum over all cells c_i in the system with the use of following formula:

$$\tilde{\rho}(\vec{p}) \simeq \sum_{i=1,N} m_{c_i} W(\vec{p} - \vec{p_{c_i}}, h),$$

where N denotes the number of cells in the system, m_{c_i} is the mass of cell c_i, $\vec{p_{c_i}}$ is the location of cell c_i in 3-D space and h is the parameter that defines the radius of the region of influence for the approximation of the density function, so-called smoothing length. The kernel function W used in our model is of the following form:

$$W(z,h) = \frac{8}{\pi h^3} \begin{cases} 1 - 6(\frac{z}{h})^2 + 6(\frac{z}{h})^3, & 0 \leq \frac{z}{h} \leq 0.5 \\ 2(1-\frac{z}{h})^3, & 0.5 \leq \frac{z}{h} \leq 1 \\ 0, & \frac{z}{h} > 1 \end{cases} .$$

Values of the kernel function for cells further away than a distance of h from $\vec{p}$ are equal to zero. In our simulations we usually assume that h is equal to the radius of the cells' neighbourhood r. With this assumption we are able to calculate the value of the density function at the location of each cell with the use of the previously determined neighbourhood. Therefore in order to increase the smoothing length h the radius of the cells' neighbourhood r needs to be increased. This can be easily achieved since r is one of the simulation's parameters. However, users should be aware of an important feature of the model: increasing the neighbourhood for each cell can also cause a significant increase in computation time. An example image presenting the density function computed for a given cellular colony is shown on Figure 5.3.

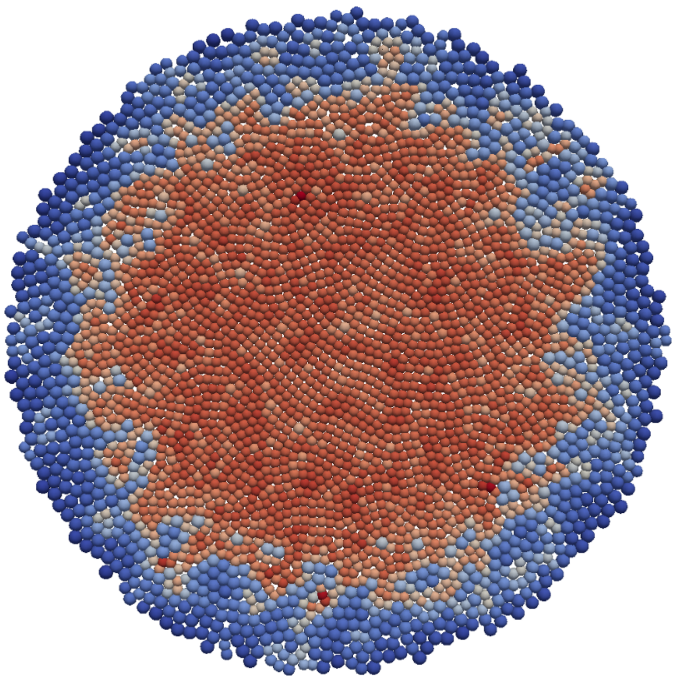


Figure 5.3: Density of cells in a colony shown for clarity in 2-D (red - higher density, blue - lower density). Proliferation occurs mainly in lower density areas.

Cell cycle update and movement of cells

The development and dynamics of cell colony in our model is driven by two previously described characteristics: potential and density functions. Each cell undergoes the cell cycle (Step 9, Scheme 1) which is controlled by the density function, i.e. cells do not divide or grow if there is no sufficient space. In the particular case when cell division occurs, the number of cells in the cellular system is increased accordingly. The next iteration of the simulation starts with a new number of cells. The movement of each cell is on the other hand driven by the gradient of the potential function, as it was described earlier. Cells move towards the direction computed from repulsive and adhesive interactions (Step 10, Scheme 1). In our model, we assume that all cells involved in the simulation reside in a 3-D finite computational box. If any cell moves out of the box, it is removed from the simulation and not considered further.

Another very important detail of the implementation is how to decide whether a cell has enough space to continue its cycle and growth. In our model this is controlled by a single parameter for each cell type, e.g. cancer cells may continue to grow in the environment far more dense than healthy cells. The parameter is related to the critical value of density at which a cell of a specific type will not be able to continue its growth. Specifying the critical density value directly is difficult and not very intuitive. In our model, the critical density value is calculated indirectly from the parameter specified by the user. This parameter specifies the minimum distance of a given cell from each of the surrounding neighbours (4 neighbours in the case of 2-D and 6 neighbours for 3-D) at which the cell can continue its growth and development. This is schematically presented in Figure 5.4 in 2-D case, four neighbour cells (orange) are equally distant from the cell placed in the middle (blue). The user can define the critical distance $d_{critical}$ apriori. The density is computed in the middle point of the blue coloured cell and the computed value is the resulting critical density value for cells of a given type.

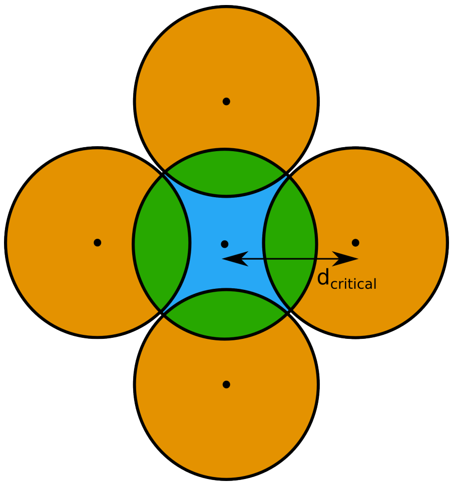


FIGURE 5.4: Image explains how the critical density value is computed. Shown in 2-D for clarity. In 3-D case middle cell (blue) has 6 instead of 4 neighbours.

5.1.2 Computational methods for global fields

For the global fields computations we employ the Hypre parallel library [53], which is a collection of high performance preconditioners and solvers for large sparse linear systems of equations on massively parallel computing systems. The discretization of the PDE is achieved with the use of an implicit in time finite difference scheme:

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - \lambda u + f - g \tag{5.2}$$

$$\begin{aligned}\frac{u^{t+1}_{x,y,z} - u^{t}_{x,y,z}}{\Delta t} = D_u\Big(&\frac{u^{t+1}_{x+1,y,z} - 2u^{t+1}_{x,y,z} + u^{t+1}_{x-1,y,z}}{h^2} + \frac{u^{t+1}_{x,y+1,z} - 2u^{t+1}_{x,y,z} + u^{t+1}_{x,y-1,z}}{h^2} \\ &+ \frac{u^{t+1}_{x,y,z+1} - 2u^{t+1}_{x,y,z} + u^{t+1}_{x,y,z-1}}{h^2}\Big) - \lambda u^{t+1}_{x,y,z} + f^{t+1}_{x,y,z} - g^{t+1}_{x,y,z}\end{aligned} \tag{5.3}$$

$$
\begin{aligned}
&(1 + 6\frac{D_u \Delta t}{h^2} + \lambda \Delta t) u^{t+1}_{x,y,z} \\
&\quad - \frac{D_u \Delta t}{h^2}(u^{t+1}_{x+1,y,z} + u^{t+1}_{x-1,y,z} + u^{t+1}_{x,y+1,z} + u^{t+1}_{x,y-1,z} + u^{t+1}_{x,y,z+1} + u^{t+1}_{x,y,z-1}) = u^{t}_{x,y,z} \\
&\qquad + \Delta t (f^{t+1}_{x,y,z} - g^{t+1}_{x,y,z})
\end{aligned}
\tag{5.4}
$$

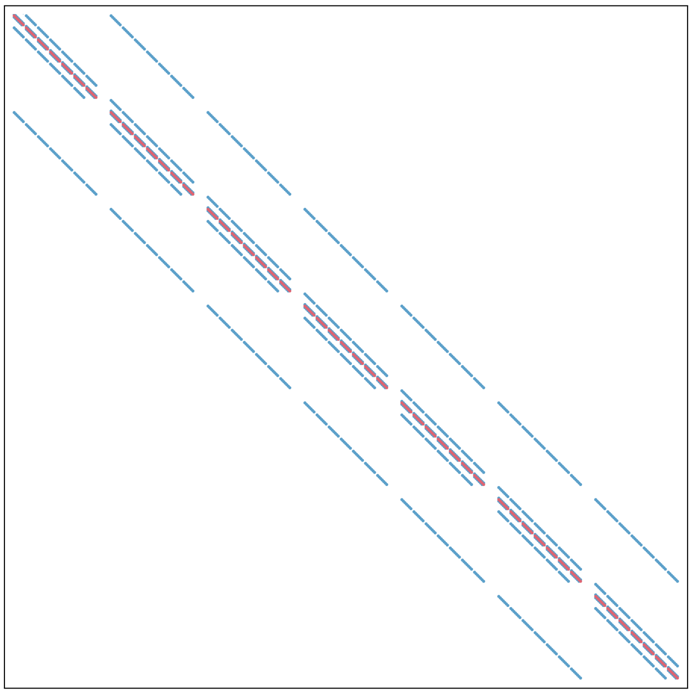


FIGURE 5.5: Sparse matrix resulting from the discretization of Equation 4.2 with finite difference method in 3-D. Non-zero elements are shown.

The discretization scheme presented in Equation 5.4 is equipped with boundary conditions appropriate to the considered global field type, as described in Section 4.2. The system is then defined in the ParCSR parallel format. The decomposition of data in Hypre is achieved by assigning computational grid blocks to different processes. We are using the Conjugate Gradient solver preconditioned by the BoomerAMG algebraic multigrid method to solve the system.

Coupling of discrete and continuous approaches

Solving global fields together with the large scale discrete cellular system is a challenging task in the context of parallel processing. An important problem lies in the block data decomposition required by the Hypre library, which is different than the HSFC and RCB methods ideal for the tree-based computations. In each iteration of the simulation the source and sink/uptake functions for the nutrient fields need to be calculated from cellular data. On the other hand individual cells need to be informed on the level of basic nutrients or temperature in their environment in order to incorporate this information into their cycle. To approach this problem we have decided that each process can determine by itself with which grid block its local cell data overlaps, see Figure 5.6. For each such patch, the local cell data is interpolated with the cloud-in-cell method (CIC). In the next step each patch is transmitted to the processor that holds the corresponding grid block. In this way the density field for each grid block is computed from the contributions of several processors. Density is then used to compute the source and sink/uptake functions. After global fields have been computed we can pull previously determined patches from the grid blocks and send them back to corresponding processors. Now the patches are carrying information about the level of nutrients or temperature in the environment. With this scenario each local cell is able to determine the concentration of nutrients in its surrounding, which is needed to proceed with the cycle scheme. The main advantage of our approach is that only the scalar values are transmitted between processors (i.e. those corresponding to density, nutrient fields and temperature). The implementation is more complicated but there is no need of transmitting the whole cell data between processors and therefore the amount of required communication is significantly reduced.

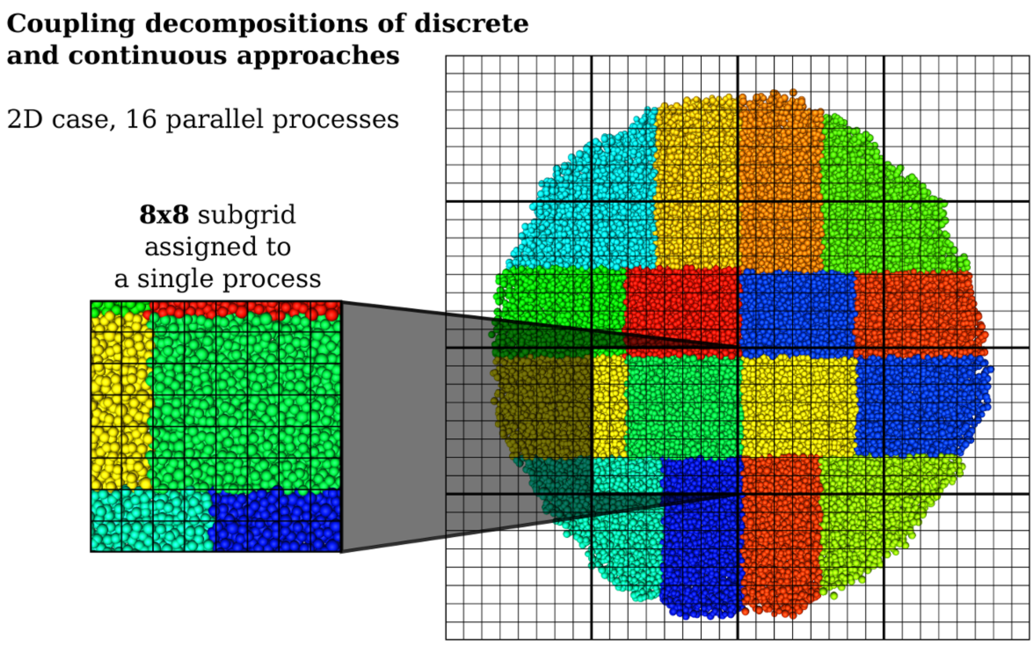

Coupling decompositions of discrete and continuous approaches
2D case, 16 parallel processes
8x8 subgrid assigned to a single process

FIGURE 5.6: Figure showing how two different decompositions of cell colony dynamics discrete method (depicted with colors) and global field continuous method (depicted with grid) are coupled together within one computational model. 2-D simulation data are used for clarity.

5.2 Highly scalable parallel implementation

5.2.1 Implementation details

Developing powerful applications for today's largest supercomputing systems, built with thousands or millions of computational cores, requires the use of advanced parallel programming techniques and tools. The most important requirements placed on the applications created for execution on current largest and future HPC systems are listed in Section 3.2. Here we describe the most important features of the implementation of our model, by which it can process large-scale biological systems efficiently and is highly optimized on specific parallel architectures. We present performance measurements of the application collected on two HPC systems available at ICM, University of Warsaw:

- **the Nostromo system (IBM Blue Gene/Q)**: 1024 compute nodes, 16384 processor cores (16 per node), 16 TB RAM memory, peak performance of approximately 210 TFlop/s,
- **the Boreasz system (IBM Power 775)**: 76 compute nodes, 2432 processor cores (32 per node), 9.7 TB RAM memory, peak performance of approximately 74 TFlop/s.

Technologically, both systems are based on the most modern computer technologies available for HPC applications nowadays.

Parallel processing techniques used in the implementation of our model are on one hand universal in use, therefore the computer code may be used on HPC systems of different architecture, e.g. it was already used on large x86 clusters. On the other hand, highly optimized versions for architectures such as the IBM Blue Gene/Q and the IBM Power 775 allows the simulations on much larger systems, for instance on one of the largest currently available European computational systems - JUQEEN which is based on the IBM Blue Gene/Q architecture and is accessible for European researchers through the PRACE project [100].

At the end of this section we present usage details and show how to run simulations on parallel systems.

Hybrid MPI/OpenMP implementation

Maintaining extremely large number of processes with the use of a flat MPI programming model very often results in a large memory overhead and significantly reduces the scalability of applications [73]. One of the most popular ways to overcome these issues is to use a multi-level or hybrid parallelization model, where MPI processes are additionally parallelized with the use of a thread based programming model. In our work we have decided to use a mixed MPI/OpenMP parallel programming model. Most of the time consuming stages of the simulation, especially those related to computing the potential function and its gradient, have been parallelized with the use of OpenMP. Additionally the Hypre library used for computing global fields has also been parallelized with the use of a mixed MPI/OpenMP model.

Depending on the size of the cellular colony in the simulation run we use up to 16 MPI processes per node on the IBM Blue Gene/Q and up to 32 MPI processes per node on the IBM Power 775. Each MPI process is then further parallelized with the use of OpenMP. In this way we can reduce the number of MPI processes used per node but also take the full advantage of SMT hardware support. Performance measurements, scalability and performance analysis of different SMT execution modes are presented in Section 5.2.2.

Non-blocking communication

One of the most important mechanisms for optimizing parallel computations based on the message-passing model is the use of non-blocking communication. This mechanism allows to overlap computations and communication, it is available in MPI standard.

The best example of the use of non-blocking communication in our application is the data transfer mechanism used for computing the potential function and its gradient, presented in Scheme 1. Part of the computations may be performed on data available locally while waiting for data transferred from neighbour processes. A second example of the use of non-blocking communication in the application is the exchange of information between the model of cellular colonies dynamics and the grid used for the discretization of continuous equations describing global fields.

Parallel I/O

Subsequent simulation states can be written to output files with the frequency specified by the user. Output files can then be further used for analysis of current results of the simulation. Typically, in order to trace all relevant stages of the simulation it is required to record tens or even hundreds of very large files. All output files generated during simulations are written in parallel by processes involved in the calculations based on the mechanism of parallel I/O available in the MPI standard. Such an implementation significantly speeds up the process of creating output files, and has a significant impact on the performance and scalability of the entire application. In the case of large-scale simulations of cellular colonies built with approximately 10^9 cells the sizes of output files are of the order of tens of giga bytes. The processing of such data can only be done by using parallel computing techniques.

Checkpoint/restart mechanism

The presented application has a functionality of saving the state of the simulation with the use of the checkpoint/restart mechanism. The method of generating restart files during the simulation is currently one of the most reliable fault tolerance mechanisms. Restart files can be used to resume the simulation in case of critical hardware or operational system faults without the need to repeat a large amount of calculations. In the case of our computational model, this mechanism has an additional advantage related to the specific scientific application. Users can generate a large cellular colony in a separate simulation prior to running many complex simulation runs to evaluate various scenarios e.g. such as those related to tumour growth in-vivo.

The checkpoint/restart mechanism available in our application is implemented with the use of a parallel I/O scheme and therefore saving and loading restart files can be done very efficiently. For instance loading a restart file consisting of more than 16 million cells (file size of approximately 2 GB) takes only tens of seconds with 32 parallel processes.

An additional improvement to the checkpoint/restart mechanism in our implementation is that the mechanism can be used between different processor architectures. In particular the application is able to perform automatic conversion between binary representation used on big-endian and little-endian systems. This makes it possible to e.g. read the restart file generated on x86 cluster (little-endian architecture) and use it on the IBM Power 775 system (big-endian architecture).

5.2.2 Performance measurements and optimization

We have performed a series of performance measurements on available HPC systems. Here we present the most important results showing that our implementation is very efficient, scalable and well prepared for running large scale simulations.

MPI/OpenMP performance

We have examined the scalability of the OpenMP performance within a single node. Testing was carried out on the IBM Blue Gene/Q architecture. We have used a starting field consisting of approximately 10^6 cells. Program was executed on 2 nodes with one MPI process per node. We have tested various numbers of OpenMP threads per process and gathered time measurements of the execution of three main stages of the simulation: potential function computation, gradient of the potential function approximation and numerical solving of global fields. We were able to investigate the impact of using different simultaneous multithreading (SMT) modes available on the IBM Blue Gene/Q architecture. It should be noted that the SMT mechanism does not increase the maximum theoretical peak performance of the system, however it might influence the performance of chosen algorithms and applications. This specific processor architecture feature is currently available in many petascale HPC systems available worldwide. Both IBM Power7 processors available in Power 775 systems and IBM Power A2 processors available in Blue Gene/Q systems are built upon 4-way simultaneous multithreaded cores.

Table 5.2: Measurements of OpenMP performance of three main stages of the simulation on the IBM Blue Gene/Q system. Speedup is shown for the best results.

Threads per node	**Potential**	**Gradient**	**Global fields**
1	57.75s	64.02s	27.38s
2	29.91s	37.73s	17.71s
4	16.27s	22.55s	9.40s
8	8.64s	12.11s	5.08s
16	4.85s	6.59s	2.89s
32 (SMT2)	2.90s	3.88s	**2.19s**
64 (SMT4)	**1.99s**	**2.78s**	2.55s
Speed up	29.02x	23.02x	12.50x

We were motivated by results presented by Abeles *et al.* [101] which show that the performance gain from using the SMT mechanism varies depending on the applications and algorithm type, program execution model, the threading mode being used on the processor, and the resource utilization of the program. One of the conclusions of the study was that throughput type workloads are best suited to achieve gains from using higher SMT modes. On the other hand, the performance of high memory traffic codes will most likely be decreased or will remain at a similar level when increasing the number of hardware threads per core.

Applications can use the SMT mechanism by running more processes and/or threads than the physical number of cores available in the system. This may be achieved by:

- execution with 2x or 4x more MPI processes than physical cores,
- execution with with 2x or 4x more OpenMP or POSIX threads than physical cores,
- execution with mixed MPI/multithread mode with the total number of threads exceeding the number of physical cores.

In the case of our application a natural candidates for efficient use of SMT mechanism were tree-based algorithms used for computing potential function and its gradient. Indeed results presented in Table 5.2 show that both algorithms can be speed up by using SMT4 mode (4 threads per core). On the other hand the algorithms used for solving global fields does not present a significant speed up when using SMT. The overall scalability of OpenMP implementation is satisfactory and very promising.

Scalability - cellular dynamics simulations

The first scalability tests were performed on the IBM Blue Gene/Q system and covered only the simulation of cellular dynamics without coupling with the continuous cellular environment. Here we describe the whole optimization process focused on achieving the best possible speed-up with increasing number of computational cores.

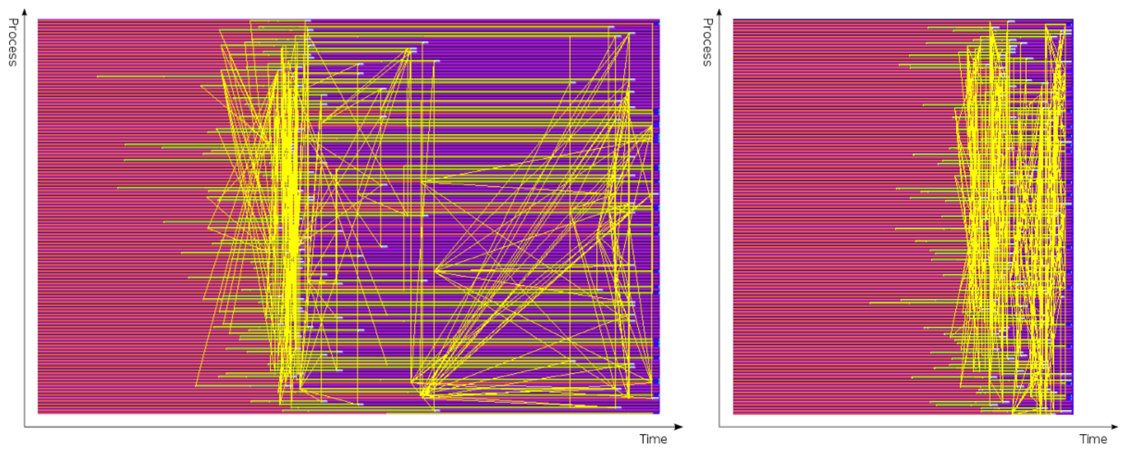


Figure 5.7: Visualization of the trace data collected during one simulation step. The imbalance between working processes in the case of non-optimal domain partitioning (left) is confronted with optimized domain partitioning (right). Measurements and plots were obtained with the use of the TAU performance analysis tool [102].

Performance optimization on the IBM Blue Gene/Q system was carried out iteratively. We have designed and executed a large scale benchmark run. We have collected MPI trace data with the use of the TAU performance analysis tool [102]. Results of the analysis showed an unexpected imbalance between parallel processes which was identified as the main bottleneck limiting the code's scalability. Figure 5.7 shows a detailed look at the computer program activities in a single iteration step of the simulation, before and after optimization. Computational activities are depicted in pink whereas the standby mode when processes are idle is depicted in purple. At the end of each iteration step there is a synchronization barrier which is preceded by an exchange of data between processes (depicted with yellow lines). As can be seen on the non-optimal version (left) some of the processes are waiting on the barrier for others to finish their work.

We have identified that non-optimal load balancing between working processes was the main performance and scalability bottleneck. The main observation was that computing potential function values is more time consuming for cells which reside in high-density areas (e.g. inside of a developing solid tumour). Such cells have more interactions with their neighbours and therefore the tree traversal is more complex. To solve this problem we have used weighted HSFC partitioning. Now each cell has a weight assigned to it, which is equal to the value of the density function computed in the cell's center. The trace analysis of the optimized code is shown in Figure 5.7 (right). As can be seen the imbalance between processes was significantly reduced.

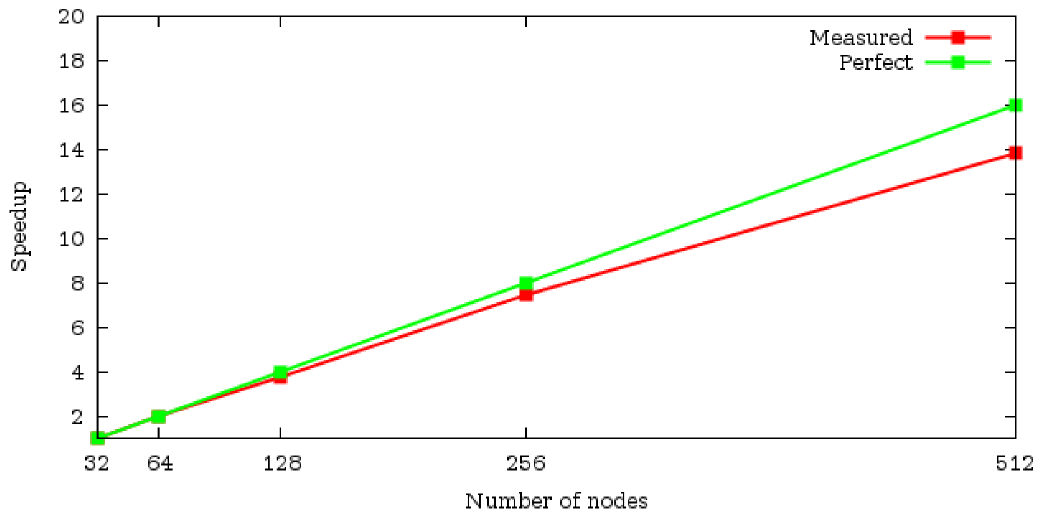


Figure 5.8: Scalability results: simulation of a large scale cellular colony on the IBM Blue Gene/Q.

The performance improvement was confirmed by plotting scalability curves, the scalability curve for the optimized version is presented in Figure 5.8. The maximal computational partition used in this measurement consisted of 512 nodes (which is equivalent to approximately 100 TFlop/s). As can be seen we achieved nearly linear speedup and we expect that this behaviour will continue on larger computational partitions for bigger benchmark sizes.

Scalability - cellular dynamics simulations with global fields

The second scalability test was performed after extending the application with implementation of the numerical solution of equations describing global fields. One of the most important tasks in the implementation of global fields was to maintain high scalability on HPC systems. To measure the scalability of the application we have used a starting field of more than 15 million cells. The benchmark consisted of 16 iterations preceded by loading a restart file of size larger than 1.5 GB. The oxygen level was computed as a continuous field in each step of the simulation. Benchmark runs were executed on the IBM Power 775 system. Results presented in Table 5.3 show good scalability of the application.

Table 5.3: Scalability results on IBM Power 775 system

Number of cores	Walltime	Timestep
16	806s	49.02s
32	455s	27.06s
64	251s	14.34s
128	159s	7.45s
256	102s	4.30s
512	71s	2.97s

5.2.3 Usage details

Standard usage of the program is achieved by defining necessary simulation parameters in the parameter file. An example parameter file is presented in Figure 5.9. Here, we describe the meaning of all important parameters, allowing better understanding of the functionality of the simulator.

The first group of parameters defines the most important settings of the simulation. The meaning, expected values and important features of these parameters are listed below.

- `NC`: sets the initial number of cells in the simulation,
  - expected value: positive integer number,
  - simulation can start with a positive integer number of cells,
  - cells are disturbed randomly in 3-D space,
- `NSTEPS`: sets the number of iterations in the simulation,

```
# Parameter file for 3DPCS

# Main simulation parameters
NC 1
NSTEPS 128
DIM 3
MITRAND 1
NHS 1000
RD 0.5
TGS 1

# Computational box
SIZEX 1024
SIZEY 1024
SIZEZ 1024

# Time steps and grid resolution
SECPERSTEP 1200
GFDT 600
GFH 4.0

# Cell cycle parameters
V 0.5
# Healthy tissue
G1 11.0
S 8.0
G2 4.0
M 1.0
# Cancer cells
CG1 6.0
CS 8.0
CG2 4.0
CM 1.0

# Output parameters
OUTDIR results
GFLOGDIR log
VISOUTSTEP 16
STATOUTSTEP 1
PROFOUTSTEP 1
RSTOUTSTEP 100
```

FIGURE 5.9: Example parameter file for simulation starting from a single cell placed in the center of computational box.

  - expected value: positive integer number,

- `DIM`: sets the dimensionality of the simulation (2-D or 3-D),
  - expected value: 2 or 3,

- `MITRAND`: controls the direction in which daughter cells are shifted from the center of mother cell during mitosis,
  - expected value: 0 or 1,
  - 1 indicates random placement,
  - 0 indicates placement consistent with the direction of movement of the mother cell in last iteration of the simulation;

- `NHS`: sets the number of cells needed to activate apoptosis i.e. programmed cell death,
  - expected value: positive integer number,
  - apoptosis is activated when the number of cells in the simulation is greater than `NHS`,

- `RD`: sets the probability for each cell of being marked for apoptosis,
  - expected value: real value,
  - assumption: $0 \leq \texttt{RD} \leq 1$,

- `TGS`: indicates that the simulation is a tumour growth simulation,
  - expected value: 0 or 1,

The second group of parameters is used to define the size of the computational box. This can be achieved by setting three positive integer numbers (two in the case of 2-D simulations) corresponding to the size of the computational box in X, Y and Z axes, parameters `SIZEX`, `SIZEY` and `SIZEZ` respectively.

The third group of parameters is associated with setting of time steps and the mesh size used for the discretization of equations describing global fields. Below is the explanation of those three parameters.

- `SECPERSTEP`: sets the number of seconds per each iteration step,
  - expected value: positive real number,

- we recommend that the value of this parameter is not greater than 3600 seconds (1 hour) since the time of cell cycle phases are usually defined in hours,

- `GFDT`: sets the time step Δt (in seconds) for the discretization of equations describing global fields,
  - expected value: positive real number,
  - important assumptions: `GFDT` $\leq$ `SECPERSTEP` and `SECPERSTEP` $= k \cdot$ `GFDT` where k is a positive integer
- `GFH`: sets the mesh size h for the discretization of equations describing global fields,
  - expected value: positive real number,
  - unit is a multiple of the maximum size of a biological cell, e.g. for $h = 2$ the mesh size will be equal to the maximum size of a biological cell multiplied by 2,

The next group of parameters is used to define cell cycle phases lengths. There is a distinction between healthy and cancer cells. Mean cell cycle phases lengths for healthy cells are set with the use of `G1`, `S`, `G2` and `M` parameters. In the case of cancer cells `CG1`, `CS`, `CG2` and `CM` parameters are used. The expected values are positive real numbers defining the lengths of each of the phases in hours. There is an additional parameter V, which is the variance used to define the exact cell cycle phases lengths for each cell individually, as described in Section 4.1.

The last group of parameters is used to control how results of the simulation should be stored. There is an assumption that all output files are located in a single directory. The name of this directory is defined with the `OUTDIR` parameter. In order to control the numerical results of global fields computations user can specify the name of the directory in which the Hypre library log files will be placed, this is defined with `GFLOGDIR`. The rest of the parameters specify how often the output informations are written, i.e.: the `VISOUTSTEP` parameter defines how often the visualization files will be written, the `STATOUTSTEP` parameter defines how often the statistics of the simulation will be written to the standard output stream, the `PROFOUTSTEP` parameter defines how often the profiling information (showing timings of different procedures and functions) should be written to the file called `prof.out`. All these parameters are defined by specifying the number of iterations, which separate successive writing out of the results.

The application allows users to use the checkpoint/restart mechanism. In order to write checkpoint files the user has to specify how often this should be done. This is done by

```
# Restart parameters
RSTFILE step00007000.rst
RSTRESET 0
```

FIGURE 5.10: Parameters required to start the simulation from a restart file.

setting the parameter RSTOUTSTEP to the number of iterations of the simulation, which separate successive checkpoints. In order to restart the simulation from the checkpoint file the parameter file should be modified to include two additional lines presented in Figure 5.10.

The parameter RSTFILE should be set to point to the file containing the restart information. In such a case, by default all simulation parameters are set to values stored in the restart file. However sometimes it is useful to use a starting field containing a large scale model of a healthy tissue to start a completely new simulation with some of the parameters overwritten. This can be achieved by setting the parameter RSTRESET to 1.

In order to run the simulations it is required to define the number of parallel processes and number of OpenMP threads assigned to each process. The application should be executed with the use of a special command used for running MPI programs. This command is system dependent, e.g. `mpiexec` is usually available on Linux clusters but on the IBM Blue Gene/Q architecture `runjob` or `srun` commands should be used. Here we present an example of execution commands used on the IBM Power 775 system:

```
setenv OMP_NUM_THREADS 2
mpiexec -n 32 ./3dpcs -p parameters.txt
```

At the beginning of the simulation the application reports the most important parameters by writing the output header to standard output stream. Users should verify that all parameters including the number of processes and threads were set properly. An example output header is presented in Figure 5.11.

During the execution the current status of the simulation is reported and written to the standard output stream. An example status can be seen in Figure 5.12.

5.3 Visualization of results

Simulation results can be analysed with the use of the visual analysis and visualization package VisNow [103]. For large scale simulations we have also developed an alternative method based on the ray-tracing visualization package Pov-Ray [104]. Pov-Ray input

```
Tue Jun 17 19:15:57 CEST 2014

    _/_/_/          _/
         _/    _/_/_/  _/_/_/      _/_/_/    _/_/_/
    _/_/    _/    _/  _/    _/  _/        _/_/
       _/  _/    _/  _/    _/  _/            _/_/
_/_/_/      _/_/_/  _/_/_/      _/_/_/  _/_/_/
                   _/
                  _/

3DPCS - 3 Dimensional Parallel simulator for Cell Systems

Author: Maciej Cytowski
Interdisciplinary Centre for Mathematical and Computational Modelling,
University of Warsaw
http://www.icm.edu.pl

Running on 32 processors.
Each process working with 2 OpenMP threads.
This is a big endian system.

Reading parameters file: parameters.txt
Box size: 1024x1024x1024
Output directory: results
Log directory: log

Particle allocation finished (117 MB allocated).
Zoltan Version 3.600. Initialized.
ZOLTAN Load balancing method = 3 (RCB)
Grid size:104x104x102
```

FIGURE 5.11: Output header containing most important parameters of the simulation.

files containing 3-D scenes descriptions are created during simulation. Those input files are processed on the HPC system afterwards to produce high quality images. One of such images is presented in Figure 5.13.

```
------------------------------------------------------------------------
 Step     8000,   Time step = 1200.0000  Number of cells =       1093663
------------------------------------------------------------------------


 Time: 2666.2822

 Tree build...done
 Potential computations...done
 Potential gradient...done
 Solving field: oxygen...done

                   Min       Max       Avg       Dev
     Density    0.7622    8.9038    6.9236    0.8129
    Distance    0.0778    3.1616         -         -
    Velocity    0.00000    0.0035         -         -
        Size    0.1984    0.2528         -         -

  Cell phase          G0          G1           S          G2           M
 N. of cells      187446      349737       22603       13219        3846

   Healthy cells           0
    Cancer cells     1093663
  Necrotic cells      516812
 Tumour diameter     1.79 mm
 Cells movement...done
```

Figure 5.12: Output header containing most important statistics of the simulation.

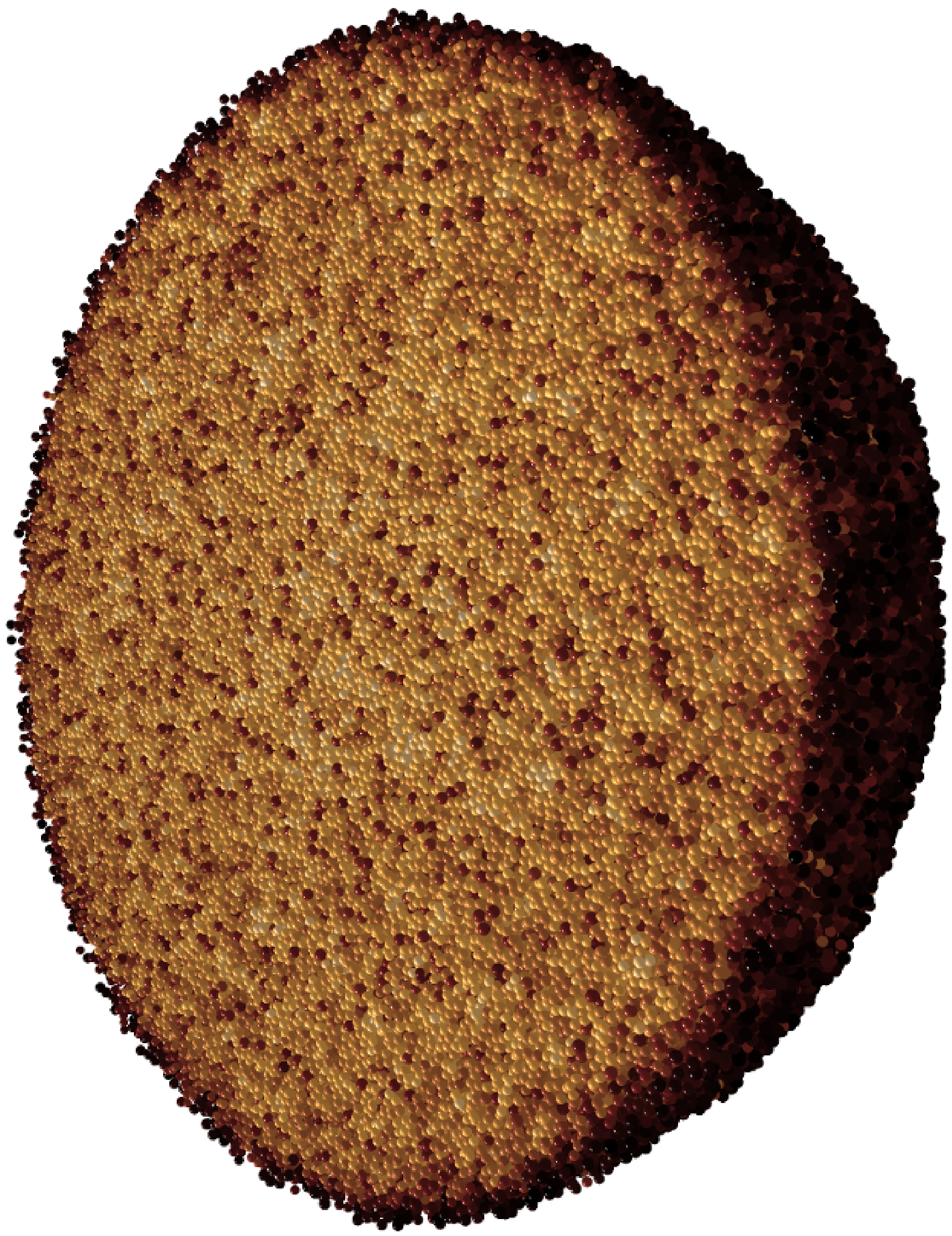


FIGURE 5.13: Example high quality image created with the use of Pov-Ray module implemented in the simulation package.

Chapter 6

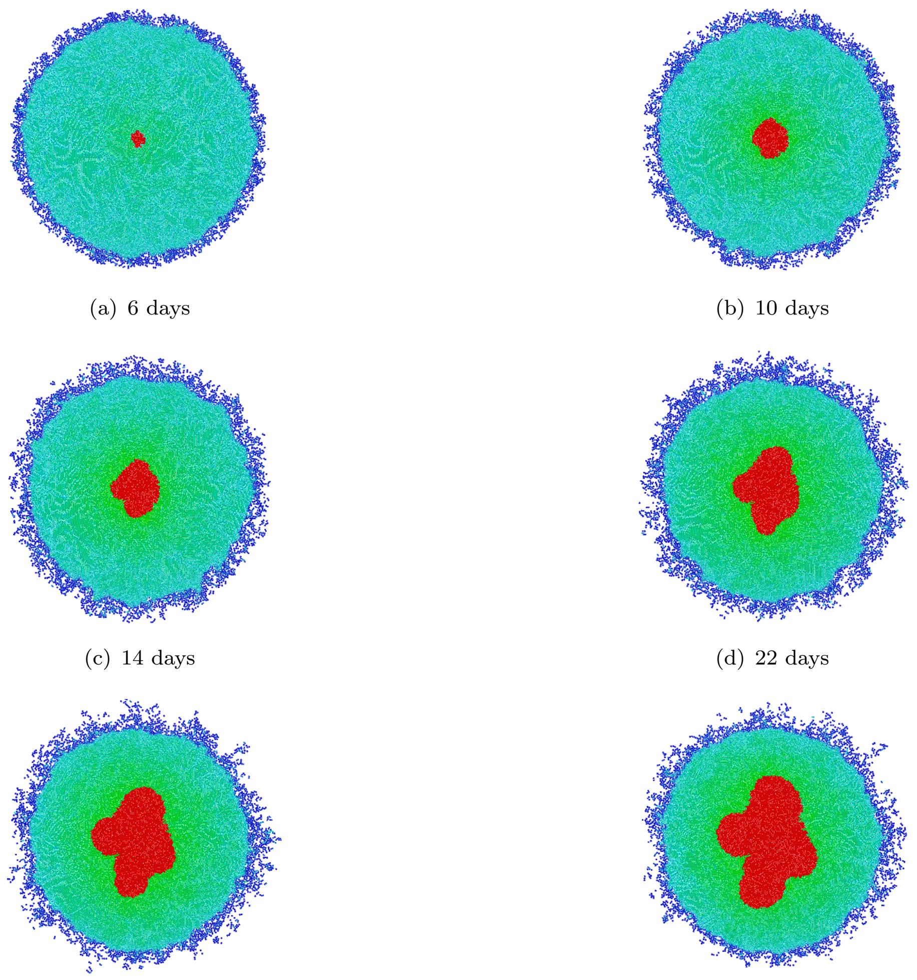
Reference applications

In the developed world, cancer is second only to heart-disease as the major cause of death. In a natural way, the question arises as to how to construct mathematical models of complex biological processes such as carcinogenesis. Since most cancer treatments refer to specific forms of activity of tumours (e.g., invasion or angiogenesis) it appears reasonable to build models based on specific biological questions. Creating such simple, partial models can be helpful in understanding complex phenomena. This approach, however, carries the risk of omission of important aspects of the process. Therefore models constructed in this way must still confront the results of experience, looking to distinguishing the effects of the test mechanism from the others, giving similar results, and paying special attention to the cases (cf. parameter space) where the model results are not in agreement with the results of experiments.

6.1 Solid tumour growth *in-vivo*

6.1.1 Two-dimensional case

We simulated the phenomenon of *in vivo* growing solid tumour in its initial avascular stage. We start the simulation when the cell colony is in the stage of homeostasis and contains about 25000 cells corresponding to healthy tissue. Average cell cycle phases lengths are equal to: 9, 11, 1, 1 hours for G_1, S, G_2 and M phases, respectively. The exact length of each cell cycle phase for respective cells is taken randomly from the interval $[T \cdot (1-V), T \cdot (1+V)]$, where T is the average length of the phase, and V is the volatility. In the case of the described simulation $V = 0.5$. Moreover we assume that the probability P_{c_i} for healthy cells to die is equal to 0.5.



(a) 6 days (b) 10 days

(c) 14 days (d) 22 days

(e) 31 days (f) 39 days

FIGURE 6.1: Plots showing the simulation of growing solid tumour inside a healthy tissue. Red color indicates tumour cells, whereas colors from navy blue to green indicates the growing density of cells. Chosen plots correspond to the growing tumour after 6 (24998 total number of cells including 70 tumour cells), 10 (24991 total number of cells including 622 tumour cells), 14 (24996 total number of cells including 1181 tumour cells), 22 (24991 total number of cells including 2374 tumour cells), 31 (24990 total number of cells including 3656 tumour cells) and 39 (24997 total number of cells including 5102 tumour cells) days of growth, respectively. Average cell cycle phases length equal to: 9, 11, 1, 1 hours for G_1, S, G_2 and M phases, respectively and the volatility V is equal to 0.5

At the beginning of the simulation a single tumour cell in the G_1-phase is placed in the middle of those uniformly distributed cells. Tumour cells divide repeatedly following a set of rules and produce a cluster of cells and in contrast to the healthy cells they have down regulated apoptotic pathways. This scenario corresponds to a growth of cancer in the middle of healthy tissue. Figure 6.1 presents the simulation results of this computational experiment. Plot 6.1(a) shows the tumour growing for 6 days, plot 6.1(b) 10 days, plot 6.1(c) 14 days, plot 6.1(d) 22 days, plot 6.1(e) 31 days and finally plot 6.1(f) 39 days after the appearance of the first tumour cell. The red color indicates tumour cells, whereas colors from navy blue to green indicates the growing density of cells.

The results of the simulations show how the cancer cells in the tissue replaces healthy cells gradually leading to the impaired functioning and thus inactivation. We can also see how small differences in the density of healthy tissue leads to a breakdown of symmetry and the consequent emergence of clinically observed tumours of different shapes.

6.1.2 Three-dimensional case

In this section we present an example 3-D simulation of early tumour growth obtained with our model. Solid tumours are believed to start from a single mutated cell that, due to subsequent divisions, becomes a multi-cellular spheroid (MCS) [105]. In the subsequent development the tumour adapts more complex structure and looses its spherical symmetry. Here, the simulation size was reduced to 10^6. Thanks to this we were able to present visualizations of the whole underlying phenomena. Visualizations of 10^9 size simulations are actually limited to the analysis of specified smaller subregions of the tissue.

Prior to starting the simulation we develop a large colony consisting of 10^6 cells that represents a healthy tissue. At the beginning of the simulation healthy cells start to undergo the G_1/S and G_2/M checkpoints. We have chosen the probability P_{c_i} of being marked for programmed cell death (apoptosis) to be equal to 0.5. This ensures the homeostasis of the whole cell population, which basically means that we obtain the stabilization of the cell number in the colony. Also, at this point of the simulation we place a single tumour cell in G_1 phase in the middle of the tissue.

This initial tumour cell divides repeatedly following a set of rules and produces a cluster of cells. The system is updated repeatedly as the program runs through a loop. During one time step, for each cell its phase is checked and, if necessary, updated. A cell divides if it has sufficient space around it to place its daughter cell. If there is no free place nearby, the cell can “push” up some neighbouring cells in order to create an empty space for its daughter cell. The exact spatial location for the new cell is computed using a potential function.

There is a significant differentiation between cancer and healthy cells in the presented simulation. Firstly, for healthy cells average cell cycle phases lengths are: 11, 8, 4 and 1 hour for G_1, S, G_2 and M phases respectively. Cancer cells have a shorter G_1 phase which is only 6 hours in average. Secondly, cancer cells have a greater ability to grow and divide in a high-density neighbourhood than healthy cells. Finally, the diseased

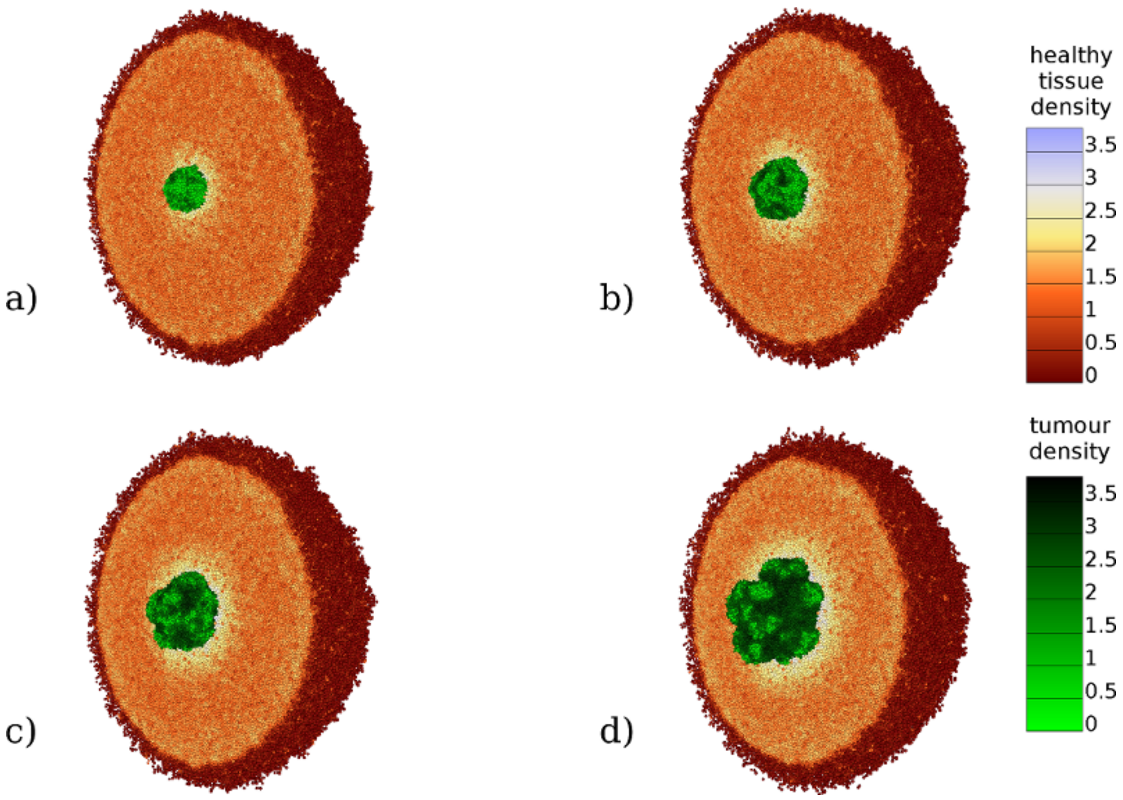


FIGURE 6.2: Plots showing the simulation of growing solid tumour inside a healthy tissue. Plot a) corresponds to the growing tumour after 80 days, with 2742 cancer cells. Plot b) corresponds to the growing tumour after 82.5 days, with 8091 cancer cells. Plot c) corresponds to the growing tumour after 85 days, with 15626 cancer cells. Finally, plot d) corresponds to the growing tumour after 90 days, with 37681 cancer cells. The whole cell population in all time steps consisted of approximately 10^6 cells.

cells produce slightly weaker adhesion forces. We have been able to incorporate all those differences to the life cycles of single cells thanks to the individual-based approach of our model. The chosen steps of the simulation together with the description of their main characteristics are presented in Figure 6.2.

6.1.3 Extreme scale three-dimensional simulation

In this Section we present an example of an extreme scale tumour growth simulation. The whole simulation consists of two main steps. The first step of the simulation produces a healthy tissue consisting of approximately $2.5 * 10^8$ cells. The result is stored in a restart file. The size of the restart file is approximately 26 GB. The second step is started by placing a single tumour cell inside the healthy tissue. We stopped the simulation when the tumour consisted of approximately $75,000$ cells. The cross-section of the tissue is presented in Figure 6.3.

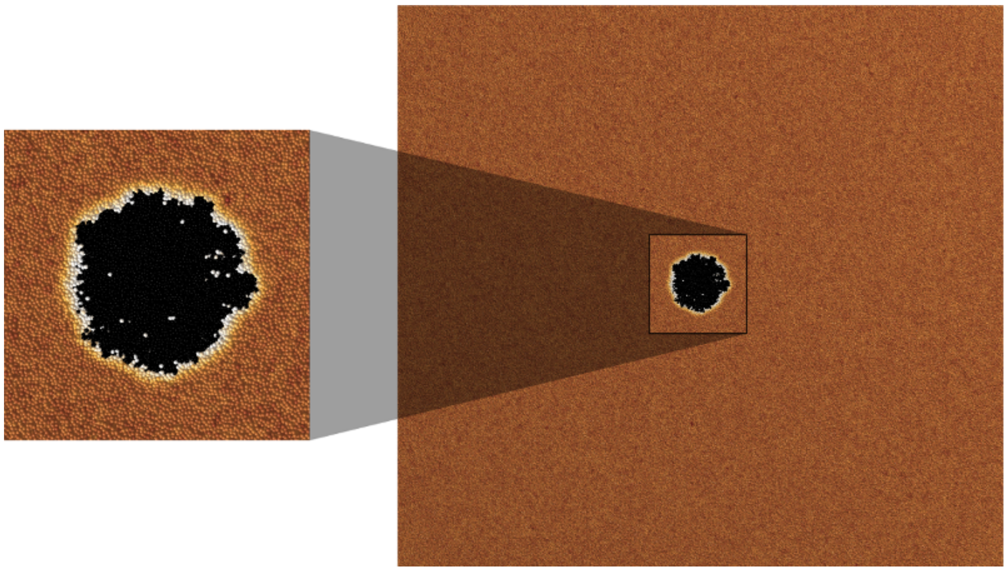


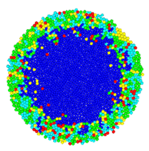
FIGURE 6.3: The cross-section of the tissue consisting of approximately $2.5 * 10^8$ healthy cells and 75,000 tumour cells. Tumour cells are shown with black color. On the left-hand side the structure of the tumour is shown in a close-up view.

Simulations of this size require the usage of HPC resources. A single time step is computed in approximately 100 seconds on 128 IBM Power7 cores. As it was noted in Section 5.2.3, we recommend that a single time step should correspond to at most 3600 seconds of a real biological process development. In order to achieve a reasonable tumour growth simulation at least 3 months of real biological process development should be simulated. Therefore, on the given computational partition (128 IBM Power7 cores) it takes more than 60 days to compute the final result.

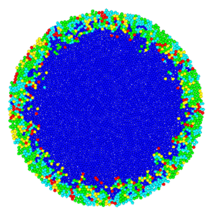
6.2 Solid tumour growth *in-vitro*

6.2.1 Two-dimensional case

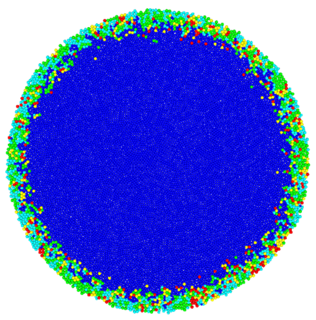
We simulate the phenomenon of an *in vitro* growing solid tumour. At the beginning of the simulation there is a single cell located at the domain centre in the G_1-phase of the cell-cycle. This initial cell divides repeatedly following a set of rules and produces a cluster of cells. Figure 6.4 shows the plots of a fast growing solid tumour in the case when average cell cycle phases lengths are: 6, 7, 3, 2 hours for G_1, S, G_2 and M phases, respectively and cancer cells down-regulate their apoptotic pathways. Different colors denote corresponding cell cycle phases, i.e. navy blue, light blue, green, yellow and red corresponds to G_0, G_1, S, G_2 and M phases respectively. Plots 6.4(a), 6.4(b), 6.4(c), 6.4(d) present the solid tumour after 10, 13, 17 and 21 days respectively. Figure 6.5 shows the corresponding plots of density inside the growing solid tumour. The red color indicates higher and blue shades lower cell density. The overall dynamic looks similar to those known from experimental *in vitro* studies, i.e. there is an inner core of non-proliferating cells surrounded by a ring of proliferating cells.



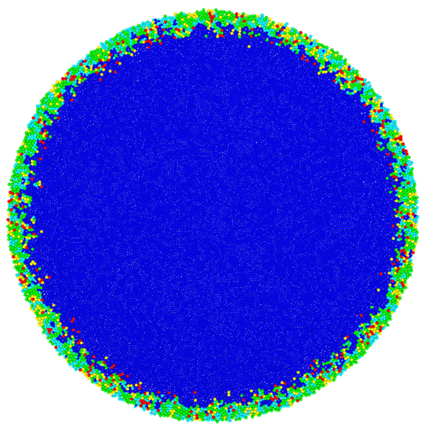
(a) 10 days



(b) 13 days



(c) 17 days

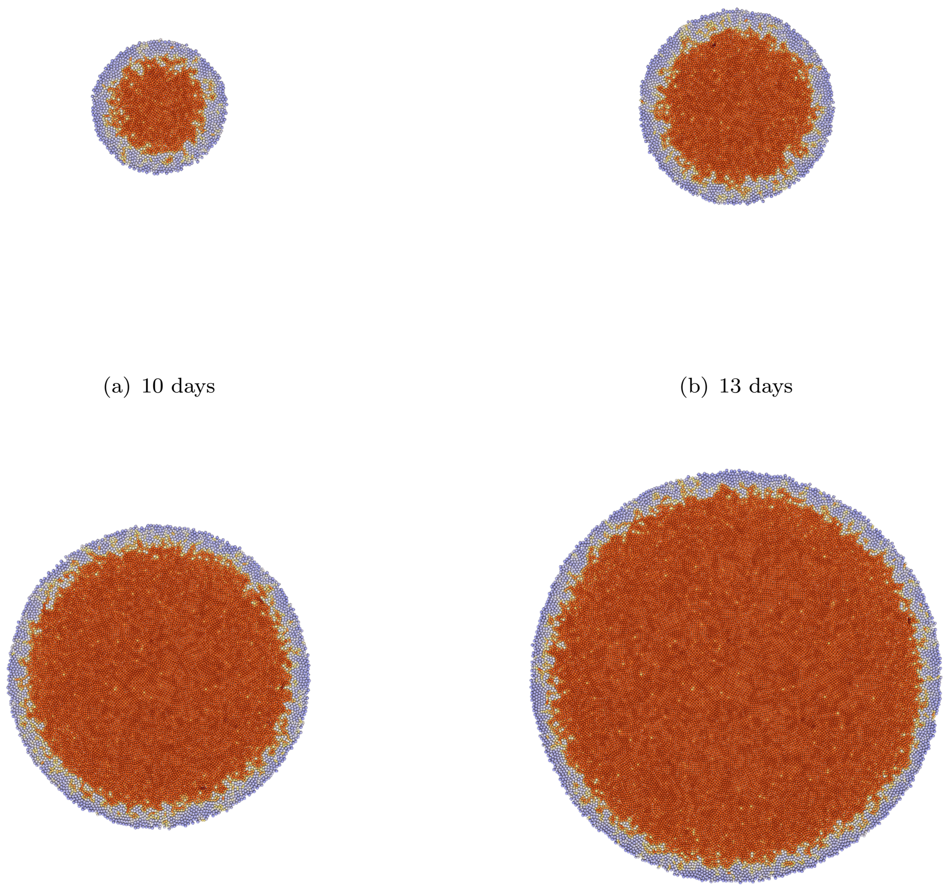


(d) 21 days

FIGURE 6.4: Plots showing the simulation of growing solid tumour cell cycle phases when average G_1 phase length is equal to 6 hours and tumour cells down-regulate their apoptotic pathways. Chosen plots correspond to the growing tumour after 4, 13, 17 and 21 days of cultivation respectively. Navy blue, light blue, green, yellow and red corresponds to G_0, G_1, S, G_2 and M phases respectively.

6.2.2 Three-dimensional case with oxygen field

Here we will show the results of simulations in which, in addition to the cancer cell colony we also consider the spatial distribution of oxygen concentration.



(a) 10 days (b) 13 days

(c) 17 days (d) 21 days

Figure 6.5: Plots showing the simulation of growing solid tumour density when average G_1 phase length is equal to 6 hours and tumours cells down-regulate their apoptotic pathways. Chosen plots correspond to the growing tumour after 10, 13, 17 and 21 days of cultivation respectively.

It is believed that a solid tumour starts from the occurrence of abnormal mutations within a single cell. Due to subsequent divisions the tumour becomes a multi-cellular spheroid (MCS). During this initial avascular growth phase, the MCS obtain nutrients from external growth factors, such as glucose or oxygen, which diffuse into the interior. That is when the tumour adopts a multi-layered internal structure. Cells towards the centre, being devoid of vital nutrients, stop to proliferate and become quiescent. Subsequent growth of the tumour results in a decrease in internal nutrient concentration, which leads to the death of quiescent cells in the centre of the MCS and gives rise to a necrotic core. This means that a mature MCS possess a well-defined internal structure, comprising a central core of necrotic (dead) cells, surrounded by a layer of

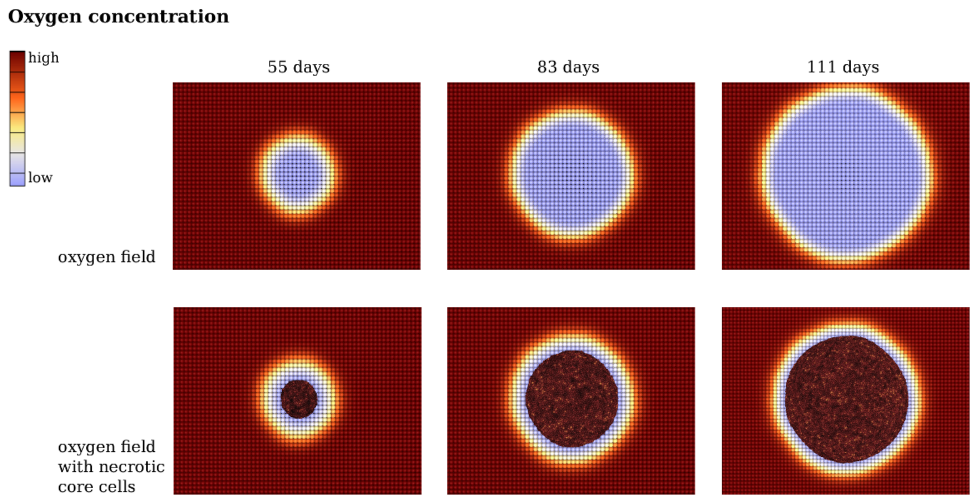


FIGURE 6.6: Plots showing the oxygen concentrations corresponding to growing multicellular spheroid (MCS) presented in Figure 6.7. Upper row shows the oxygen profile for simulations after 83, 55 and 111 days respectively. Lower row shows the same oxygen profiles with superimposed necrotic cores.

non-proliferating, quiescent cells, with proliferating cells in the outer, nutrient-rich layer of the tumour (see [106] Fig. 4a). Typically, these avascular spheroids reach a size of a few millimetres in diameter [17, 107].

In Figures 6.6 and 6.7 we present a simulation of the development of the MCS. In addition to a growing colony of cells, we take into account the dynamics of the oxygen concentration and interaction between cells and the field of oxygen. It can be seen that the growing *in silico* MCS adopts a multilayer structure known from the experiments.

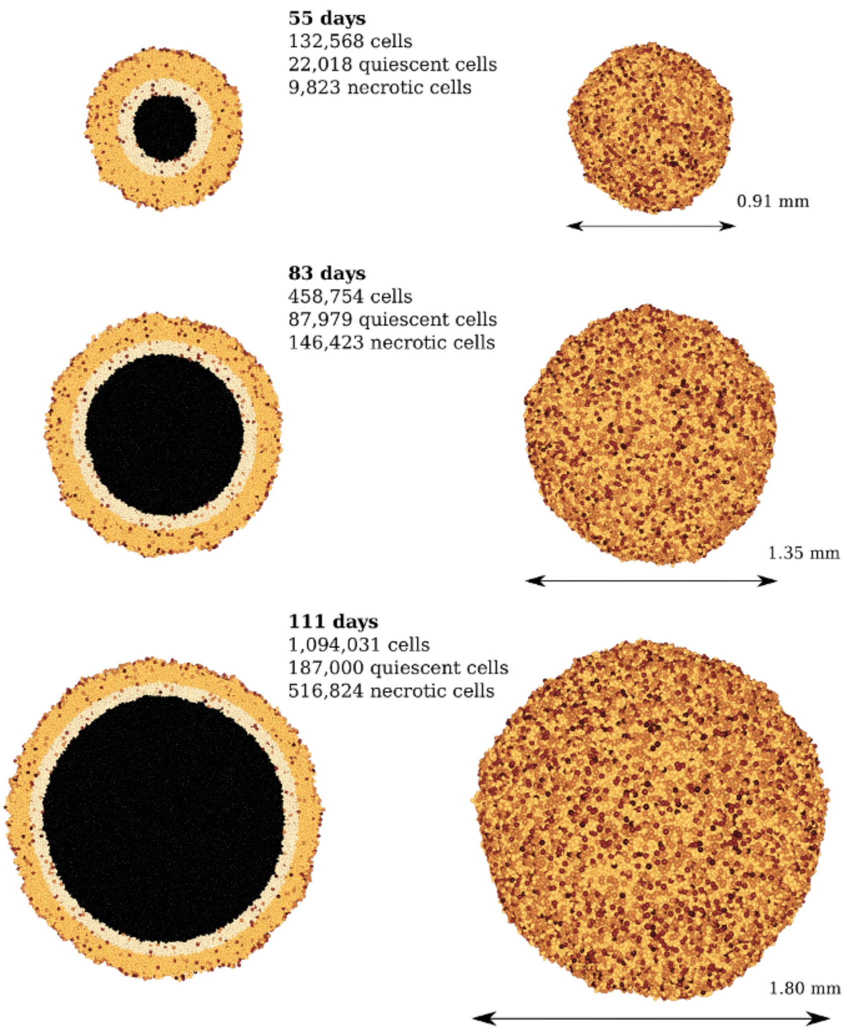


FIGURE 6.7: Plots showing the simulation of *vitro* multicellular spheroid (MCS) growing in an oxygen-containing environment. The cells in the G_1 phase are marked in yellow (●), cells in S phase are marked in orange (●), cells in G_2 phase are marked in red(●), cells in M phase are marked in claret (●). Quiescent cells in G_0 phase are marked in ecru (●) whereas necrotic (dead) cells are marked in black (●). Upper plot corresponds to the growing MCS after 55 days, with 132568 cells. Middle plot corresponds to the growing MCS after 83 days, with 458754 cells. Lower plot corresponds to the growing MCS after 111 days, with 1094031 cells. Figure 6.6 presents the corresponding oxygen concentrations.

Chapter 7

Conclusions and future work

In this work we have presented an extremely powerful tool for simulating complex biological systems consisting of a cell colony composed of different cell types growing and interacting with the environment. The cell colony is represented in a discrete manner as an individual-based model and is coupled with the environment represented by a PDE continuous description. What makes our model absolutely unique is its large scale, as it enables the simulation of systems consisting of 10^9 cells. This means, for example, being able to run a multi-scale simulation of structures such as solid tumours observed clinically, which we believe is a genuine breakthrough in terms of mathematical modelling. The model can be used to model a wide range of biological phenomena. In particular, they may be different processes associated with the development of tumours, such as solid tumour growth, both in the avascular and vascular phases.

In this work we have extensively used modern methods of parallel processing, one of the main objectives of the study was to obtain high computation optimization and scalability for emerging HPC systems. Addressing scientific challenges in such a large scale required the selection of appropriate algorithms, data structures and decomposition methods. In particular, the use of tree data structures and tree-based algorithms and their optimization for a given application significantly reduced the computational time and thus allowed for the examination of large cellular colonies in a 3-D discrete lattice-free model. Developing computational models for parallel systems requires the use of appropriate decomposition methods. The selection of the appropriate method is important for achieving high scalability. In this work we have presented the efficient application of two classes of decomposition methods. The first class, based on geometrical decomposition methods like HSFC and RCB, is used in the computations of the discrete model describing the dynamics of cellular colonies. The second class, based on the block decomposition method of the computational grid, is used to numerically solve continuous global fields' equations introduced in the model. We have also presented an efficient method of combining both types of decompositions, which is required when coupling cellular and tissue scales.

Furthermore, it was necessary to apply a number of advanced programming methods and optimization techniques. Among others, the most important features of the application are: the use of the hybrid MPI/OpenMP computing model, the non-blocking communication mechanism, the parallel I/O implementation and the design of an efficient checkpoint/restart mechanism. All aforementioned improvements of the computational model enabled efficient simulations on massively parallel systems, high scalability on the other hand for the first time allowed to undertake the scientific challenges in the field of modelling of cellular biosystems in clinically detectable scale.

Our main objective was to propose a model that enriched with various models of signalling pathways will examine the reciprocal influence of the phenomena occurring at the intracellular, cellular and in the end the tissue level. The proposed tool allows to simulate a number of biological phenomena, but especially those occurring in the tissues in which the extracellular matrix is poor such as in epithelial tissue. By this our solution has the opportunity to contribute to the formulation of new hypotheses and to optimization of therapeutic protocols in cancer. Such a model has also the potential to be extremely useful in modelling the effects of intracellular events and their impact on higher scales i.e. the cellular and tissue scales. Without this multiscale approach, we will not be able to investigate the various effects of intracellular pathways on a growing cell colony and how different external stimuli influence global dynamics.

Planned future work can be divided into those relating to the development of the model itself and those relating to the use of the existing model. As regards the first of the planned lines of research, we plan to enrich the model by a description of the extracellular matrix which takes into account the fibrous proteins such as collagen, which is very important for modelling the connective tissue. A further step will be to supplement the model with acquisition of data from, for example, 3-D medical imaging of various modalities, the conversion of the data into the model input data. This will allow to simulate specific cases, allowing, for example, predictions of the directions of tumour invasion. When it comes to the use of the existing model, we plan to investigate the impact of cell cycle length on the effectiveness of hyperthermia. To this end we will equip the proposed model with the intracellular Hsp70 protein synthesis model [108] [109] and we will test efficacy of different heat application scenarios on different cell population types. Another application will be to study the effects of the microenvironment on the dynamics of a solid tumour. Using numerical simulations we want to show for example how harsh conditions, i.e. availability of oxygen and glucose affect the phenotype of tumour invasion. Finally, in collaboration with researchers from Moffitt Cancer Center, we would like to test the efficacy of various protocols of radiotherapy.

In addition to the benefits for the life sciences, mathematical modelling has a chance to contribute to the development of mathematics, in particular in areas such as numerical analysis (e.g., the numerical solution of parabolic-hyperbolic systems, free and moving boundary problems), qualitative analysis of partial differential equations, asymptotic and multi-scale analysis (e.g. homogenisation). Often the design and analysis of a new model requires the use of new mathematical methods.

Mathematical modelling in the biomedical sciences is a challenge not only scientifically but also socially, since research is often rigidly divided into academic disciplines. The benefits of the use of mathematical models are now indisputable, but one has to remember that every success requires an interdisciplinary approach. The design and validation of models based on quantitative experimental data, designing new experiences, finding new mathematical methods allowing for an appropriate description of the process and its analysis and, finally, numerical simulation and optimization of treatment requires the cooperation of specialists in various fields. Progress in overcoming these barriers is already noticeable, the use of mathematical models in oncology is slowly becoming recognized as a research tool.

Appendix A

N-body problems - computational methods

In this section we describe the most popular and appropriate methods for solving N-body problems. All of these methods were developed for physical phenomena where each object in 3-D space interacts not only with its neighbours (short-range interactions) but also with other objects that are distant (long-range interactions). We present appropriate references for further readings.

Particle-Particle Method

The simplest approach to solve Equation 5.1 is reffered to as the Particle-Particle method. It is a direct solving method where each pair of objects is examined and corresponding potential is computed. The Particle-Particle method is not feasible for large number NC of objects due to its $O((NC)^2)$ complexity. The pseudo code of this method is presented in Algorithm 6.

Scheme 6 Particle-Particle

```
1: for $i = 1 \rightarrow NC$ do
2:     for $j = i + 1 \rightarrow NC$ do
3:         $F = F(c_i, c_j)$
4:         $V(c_i) += F$
5:         $V(c_j) += F$
6:     end for
7: end for
```

In our model, for each cell c_i we consider only interactions with those cells which are enclosed by the finite neighbourhood $B_{\epsilon_i}(c_i)$. Therefore, the pseudo code of the Particle-Particle method can be rewriteen as shown in Algorithm 7.

Scheme 7 Particle-Particle-ϵ

```
1: for $i = 1 \to NC$ do
2:   for $j = i + 1 \to NC$ do
3:     if $c_j \in B_{\epsilon_i}(c_i)$ then
4:       $F = F(c_i, c_j)$
5:       $V(c_i) += F$
6:       $V(c_j) += F$
7:     end if
8:   end for
9: end for
```

In the above formulation we omit force calculations for cells that are distant enough. However we introduce an additional operation to check the distanse between cells.

Barnes-Hut Algorithm

One of the most extensively used approaches for solving N-body problems are those based on tree representation of objects. The classical Barnes-Hut algorithm [110] is a very good example on how to use tree structures in N-body codes. The algorithm is widely used in astrophysics [94] [111] and was successfully parallelized [112]. The very first step of the Barnes-Hut algorithm in 3-D is the octree construction which proceeds as follows. We begin with a cube in the 3-D space which is the root of the tree and encloses all objects of the system. The cube is divided into eight smaller cubes of $\frac{1}{8}$ the area which are called octants. Each octant can be further broken into eight cubes and so on. In most of the cases, objects do not fall uniformly into the consecutive octants and we are using a technique called adaptive octrees wherein we divide an octant if it encloses more than a certain number of objects. The pseudo code of the sequential algorithm for octree initialization can be formulated with the use of two procedures as shown in Algorithm 8 and 9. Throughout this work we will refer to the cube of node n of the tree by $C(n)$. The lenght of the edge of the cube $C(n)$ (size of the cube) will be indicated by $D(C(n))$.

Scheme 8 BuildOctree

```
1: $root = empty\ node$
2: for $i = 1 \to NC$ do
3:   InsertObject(i,root)
4: end for
5: Eliminate empty leaves by traversing the tree
```

The BuildOctree procedure is a driving routine which inserts consecutive objects into the tree. We start with an empty node and each call to the InsertObject procedure reconfigures the tree and determines the appropriate location of the object based on

Scheme 9 InsertObject(i,n)

```
1: if n is not a leaf then
2:    determine in which child c of node n the object i resides
3:    InsertObject(i,c)
4: else
5:    if n is a leaf and is full then
6:       add eight childern of node n
7:       move objects that were located at n into appropriate child nodes
8:       determine in which child c of node n the object i resides
9:       InsertObject(i,c)
10:   else
11:      if n is a leaf and is not full then
12:         store object i in node n
13:      end if
14:   end if
15: end if
```

its cartesian coordinates. The InsertObject procedure is called recursively however, as with other tree based algorithms, it is easy to implement its non-recursive version which usually presents better performance on modern processor architectures. Each leaf node of the tree constructed in such a way consist of a number of objects less than a maximum fixed number of particles in a node. There exist tree algorithms that permit a maximum number of objects per tree leaf greater than one [113]. In our case we will assume that a leaf node is full if it consists a single object.

The second step of the algorithm is calculating the potential function for all the objects in the tree. This is the main core of the Barnes-Hut algorithm. For each object the tree is traversed starting from the root and corresponding potential function contributions are gathered from interacting objects. The Barnes-Hut algorithm is based on the idea of approximating the long-range forces by aggregating objects distant enough into one object, with the position equal to the center of mass of corresponding objects, and the mass equal to the sum of masses. The decision is made based on the size of the cube ($D(C(n))$) coressponding to the specified tree node n and the distance r from the object to the center of mass of the cube. The algorithm for computing the potential function over all objects can be formulated with the use of two procedures schematically shown in Algorithm 10 and 11.

Scheme 10 TreePotential

```
1: $n = root$
2: for $i = 1 \to NC$ do
3:    ObjectPotential(i,n)
4: end for
```

Scheme 11 ObjectPotential(i,n)

```
1: $F = 0$
2: if n is a leaf then
3:     F = potential computed directly
4: else
5:     r = distance from object i to the center of mass of objects in node n
6:     if $\frac{D(C(n))}{r} < \theta$ then
7:         compute F directly using the approximation
8:     else
9:         for all children c of n do
10:            F = F + ObjectPotential(i,c)
11:        end for
12:    end if
13: end if
14: $V(c_i) = F$
```

The TreePotential procedure is a driving routine which computes the potential for every object in the system. As it was described before the tree is traversed starting from the root for each object by calling the ObjectPotential procedure. In the case of the cellular colonies dynamics model described in this work the conditional statement in line 7 of the ObjectPotential procedure should be modified since we are considering only the finite neighbourhood for each cell. The condition should be replaced by checking whether the correspoding cube intersects with the cell neighbourhood. If this condition is not satisfied the execution of the ObjectPotential procedure should be stopped by calling a return statement.

Appendix B

SPH method

The SPH method is based on the idea that the physical quantity of a given discrete object in the system can be computed by summing appropriate quantities of all objects that lie in the neighbourhood. The contributions of each object are weighted according to the distance and density. Mathematically this is done by introducing the so called kernel function which defines a region of influence for each object. The kernel is usually defined as a function $W(\vec{p} - \vec{p}_{c_i}, h)$, where $\vec{p}_{c_i}$ is the location of object c_i and h is the value that definies the diameter of the region of influence for this object. Kernel function W has its maximum for $\vec{p} = \vec{p}_{c_i}$ and falls off smoothly to zero with distance from $\vec{p}_{c_i}$. Additionally the kernel function has the following two properties:

$$\int_V W(\vec{p} - \vec{p_{c_i}}, h) d\vec{p_{c_i}} = 1$$

$$\lim_{h \to 0} W(\vec{p} - \vec{p_{c_i}}, h) = \delta(\vec{p} - \vec{p_{c_i}})$$

Suppose that we have a function $V(\vec{p})$ defined at every position in space. We can define the convolution of $A(\vec{p})$ with the kernel $W(\vec{p} - \vec{p}_{c_i}, h)$ as:

$$\tilde{V}(\vec{p}) = \int_V V(\vec{t}) W(\vec{p} - \vec{t}, h) d\vec{t}$$

The convolution operation, defined in this way, has an effect of smoothing or blurring of the $V(\vec{p})$ function. In the SPH method the right hand side integral is replaced by a discrete expansion over objects:

$$\tilde{V}(\vec{p}) \simeq \sum_{i=1,NC} \frac{m_{c_i}}{\rho_{c_i}} V(\vec{p_{c_i}}) W(\vec{p} - \vec{p_{c_i}}, h) \tag{B.1}$$

Here we assume that each of the objects has its own density ρ_{c_i} and mass m_{c_i}. $V(\vec{p_{c_i}})$ is the value of the V function for object c_i. Moreover if we set $V(p) = \rho(p)$ then we obtain a formula which approximates the density function $\rho(p)$for each position $\vec{p}$ in space:

$$\tilde{\rho}(\vec{p}) \simeq \sum_{i=1,NC} m_{c_i} W(\vec{p} - \vec{p_{c_i}}, h)$$

Please note, that this density can be efficiently computed simultaneously with the potential function.

The two most often used kernel functions are the Gaussian function and cubic splines. Unlike the Gaussian function the spline based kernels have the property of finite region of influence since the values of the kernel for objects further away than a distance of h are exactly zero. The usage of the cubic spline kernels increases the efficiency of the SPH method significantly. In this work we propose to use two different spline based kernels formulated below.

The kernel function used in our model was originally proposed in [94].

$$W(r,h) = \frac{8}{\pi h^3} \begin{cases} 1 - 6(\frac{r}{h})^2 + 6(\frac{r}{h})^3, & 0 \le \frac{r}{h} \le 0.5 \\ 2(1 - \frac{r}{h})^3, & 0.5 \le \frac{r}{h} \le 1 \\ 0, & \frac{r}{h} > 1 \end{cases}$$

In our model cell motion is driven by the gradient of the potential function defined in Equation B.1. Lets define the derivative of the smooth function $\tilde{V}(\vec{p})$:

$$\nabla \tilde{V}(\vec{p}) \simeq \sum_{i=1,NC} \frac{m_{c_i}}{\rho_{c_i}} V(\vec{p_{c_i}}) \nabla W(\vec{p} - \vec{p_{c_i}}, h) \tag{B.2}$$

The gradient of the kernel function can easilly be computed. For $r = \vec{p} - \vec{p_{c_i}}$ the derivative can be formulated as follows:

$$\nabla W(r,h) = \frac{48}{\pi h^4} \hat{r} \begin{cases} -2\frac{r}{h} + 3(\frac{r}{h})^2, & 0 \le \frac{r}{h} \le 0.5 \\ -(1 - \frac{r}{h})^2, & 0.5 \le \frac{r}{h} \le 1 \\ 0, & \frac{r}{h} > 1 \end{cases}$$

where

$$\hat{r} = \frac{\vec{p} - \vec{p}_{c_i}}{|\vec{p} - \vec{p}_{c_i}|}.$$

Bibliography

[1] M. Cytowski, Z. Szymańska, and B. Borucki. A 2-d large-scale individual-based model of solid tumour growth. *GAKUTO International Series Mathematical Sciences and Applications, Nonlinear Analysis in Interdisciplinary Sciences*, 36:43–56, 2013.

[2] M. Cytowski and Z. Szymańska. Large scale parallel simulations of 3-d cell colony dynamics. *IEEE Computing in Science and Engineering*, 2014.

[3] A.R.A. Anderson, M.A.J. Chaplain, and K.A. Rejniak, editors. *Single-Cell-Based Models in Biology and Medicine*. Birkhauser, Basel, 2007.

[4] J. S. Lowengrub, H. B. Frieboes, F. Jin, Y.-L. Chuang, X. Li, P. Macklin, S. M. Wise, and V. Cristini. Nonlinear modelling of cancer: bridging the gap between cells and tumours. *Nonlinearity*, 23:R1–R91, 2010.

[5] H. P. Greenspan. Models for the growth of the solid tumor by diffusion. *Stud. Appl. Math.*, 51:317–340, 1972.

[6] H. P. Greenspan. On the growth and stability of cell cultures and solid tumors. *J. Theor. Biol.*, 56:229–242, 1976.

[7] D. Ambrosi and L. Preziosi. On the closure of mass balance models for tumor growth. *Mathematical Models and Methods in Applied Sciences*, 12(05):737–754, 2002. doi: 10.1142/S0218202502001878.

[8] J. P. Ward and J. R. King. Mathematical modelling of avascular-tumour growth II: Modelling growth saturation. *Math Med Biol*, 16(2):171–211, June 1999. doi: 10.1093/imammb/16.2.171.

[9] D. M. Brizel, G. S. Sibley, L. R. Prosnitz, R. L. Scher, and M. W. Dewhirst. Tumor hypoxia adversely affects the prognosis of carcinoma of the head and neck. *International Journal of Radiation Oncology*Biology*Physics*, 38(2):285 – 289, 1997. ISSN 0360-3016.

[10] M. A. J. Chaplain and G. Lolas. Mathematical modelling of cancer cell invasion of tissue: The role of the urokinase plasminogen activation system. *Math. Models Methods Appl. Sci.*, 15:1685–1734, 2005.

[11] T. L. Jackson and H. M. Byrne. A mechanical model of tumor encapsulation and transcapsular spread. *Mathematical Biosciences*, 180(1–2):307 – 328, 2002. ISSN 0025-5564.

[12] M. Cytowski, A. Ito, and M. Niezgodka. Uniqueness and local existence of solutions to an approximate system of a 1D simplified tumor invasion model. *Banach Center Publ.*, 86:45–58, 2009.

[13] P. Macklin and J. S. Lowengrub. A new ghost cell/level set method for moving boundary problems: Application to tumor growth. *J. Sci. Comput.*, 35(2-3):266–299, June 2008. ISSN 0885-7474.

[14] D. Ambrosi and L. Preziosi. On The Closure of Mass Balance Models For Tumor Growth. *Mathematical Models and Methods in Applied Sciences*, 12(5):737–754, 2002.

[15] H. M. Byrne and M. A. J. Chaplain. Growth of non-necrotic tumours in the presence and absence of inhibitors. *Math. Biosci.*, 130:151–181, 1995.

[16] H. M. Byrne and M. A. J. Chaplain. Growth of necrotic tumours in the presence and absence of inhibitors. *Math. Biosci.*, 135:187–216, 1996.

[17] H. M. Byrne and M. A. J. Chaplain. Free boundary value problem associated with the growth and development of multicellular spheroids. *Eur. J. Appl. Math.*, 8: 639–658, 1997.

[18] S. A. Enam, M. L. Rosenblum, and K. Edvardsen. Role of extracellular matrix in tumor invasion: migration of glioma cells along fibronectin-positive mesenchymal cell processes. *Neurosurgery*, 42(3):599–607; discussion 607–8, 1998. ISSN 0148-396X.

[19] R. A. Gatenby, E. T. Gawlinski, A. F. Gmitro, B. Kaylor, and R. J. Gillies. Acid-mediated tumor invasion: a multidisciplinary study. *Cancer Research*, 66(10): 5216–5223, 2006. doi: 10.1158/0008-5472.CAN-05-4193.

[20] A.L. Garner, Y.Y. Lau, Trachette L. Jackson, M.D. Uhler, D.W. Jordan, and R.M. Gilgenbach. Incorporating spatial dependence into a multicellular tumor spheroid growth model. *Journal of Applied Physics*, 98(12):124701–124701–8, 2005. ISSN 0021-8979.

[21] D. Balding and D. L. S. McElwain. A mathematical model of tumor-induced capillary growth. *J. Theor. Biol.*, 114:53–73, 1985.

[22] H. M. Byrne and M. A. J. Chaplain. Mathematical models for tumour angiogenesis: Numerical simulations and nonlinear wave solutions. *Bull. Math. Biol.*, 57:461–486, 1995.

[23] M. E. Orme and M. A. J. Chaplain. Two-dimensional models of tumour angiogenesis and anti-angiogenesis strategies. *IMA J. Math. Appl. Med. Biol.*, 14:189–205, 1997.

[24] A.R.A. Anderson and M.A.J. Chaplain. Continuous and discrete mathematical models of tumor-induced angiogenesis. *Bulletin of Mathematical Biology*, 60(5): 857–899, 1998. ISSN 0092-8240. doi: 10.1006/bulm.1998.0042. URL http://dx.doi.org/10.1006/bulm.1998.0042.

[25] D. Drasdo. On Selected Individual-based Approaches to the Dynamics in Multicellular Systems. In W. Alt, M. Chaplain, M. Griebel, and J. Lenz, editors, *Polymer and Cell Dynamics - Multiscale Modeling and Numerical Simulations*, pages 169–203. Birkhäuser, Basel, 2003.

[26] D. Drasdo and S. Höhme. A single-cell-based model of tumor growth in vitro: monolayers and spheroids. *Phys. Biol.*, 2:133–147, 2005.

[27] A. A. Patel, E. T. Gawlinski, S. K. Lemieux, and R. A. Gatenby. A cellular automaton model of early tumor growth and invasion: The effects of native tissue vascularity and increased anaerobic tumor metabolism. *Journal of Theoretical Biology*, 213(3):315 – 331, 2001. ISSN 0022-5193. doi: http://dx.doi.org/10.1006/jtbi.2001.2385. URL http://www.sciencedirect.com/science/article/pii/S0022519301923859.

[28] A.R Kansal, S Torquato, G.R Harsh IV, E.A Chiocca, and T.S Deisboeck. Cellular automaton of idealized brain tumor growth dynamics. *Biosystems*, 55(1–3):119 – 127, 2000. ISSN 0303-2647. doi: http://dx.doi.org/10.1016/S0303-2647(99)00089-1. URL http://www.sciencedirect.com/science/article/pii/S0303264799000891.

[29] A. R. A. Anderson, M. A. J. Chaplain, E. L. Newman, R. J. C. Steele, and A. M. Thompson. Mathematical modelling of tumour invasion and metastasis. *J. Theor. Med.*, 2:129–154, 2000.

[30] A. R. A. Anderson and M. A. J. Chaplain. Continous and and discrete mathematical models of tumour-induced angiogenesis and metastasis. *Bull. Math. Biol.*, 60:857–900, 1998.

[31] A. Deutsch and S. Dormann, editors. *Cellular Automaton Modeling of Biological Pattern Formation*. Birkhauser, Boston, 2005.

[32] H. Hatzikirou, L. Brusch, C. Schaller, M. Simon, and A. Deutsch. Prediction of traveling front behavior in a lattice-gas cellular automaton model for tumor invasion. *Computers and Mathematics with Applications*, 59(7):2326 – 2339, 2010. ISSN 0898-1221. doi: http://dx.doi.org/10.1016/j.camwa.2009.08.041. URL http://www.sciencedirect.com/science/article/pii/S0898122109006294. Mesoscopic Methods in Engineering and Science, International Conferences on Mesoscopic Methods in Engineering and Science.

[33] H. Hatzikirou, L. Brusch, and A. Deutsch. From cellular automaton rules to an effective macroscopic mean-field description. *Acta Physica Polonica B Proceedings Supplement*, 3:399–416, 2010.

[34] H. Hatzikirou and A. Deutsch. Lattice-gas cellular automaton modeling of emergent behavior in interacting cell populations. In Jiri Kroc, Peter M.A. Sloot, and Alfons G. Hoekstra, editors, *Simulating Complex Systems by Cellular Automata*, volume 0 of *Understanding Complex Systems*, pages 301–331. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12202-6. doi: 10.1007/978-3-642-12203-3_13. URL http://dx.doi.org/10.1007/978-3-642-12203-3_13.

[35] J. A. Glazier and F. Graner. Simulation of the differential adhesion driven rearrangement of biological cells. *Phys. Rev. E*, 47:2128–2154, Mar 1993. doi: 10.1103/PhysRevE.47.2128. URL http://link.aps.org/doi/10.1103/PhysRevE.47.2128.

[36] E. L. Stott, N. F. Britton, J. A. Glazier, and M. Zajac. Stochastic simulation of benign avascular tumour growth using the potts model. *Mathematical and Computer Modelling*, 30(5–6):183 – 198, 1999. ISSN 0895-7177. doi: http://dx.doi.org/10.1016/S0895-7177(99)00156-9. URL http://www.sciencedirect.com/science/article/pii/S0895717799001569.

[37] N. J. Poplawski, U. Agero, J. S. Gens, M. Swat, J. A. Glazier, and A. R. A. Anderson. Front instabilities and invasiveness of simulated avascular tumors. *Bulletin of Mathematical Biology*, 71(5):1189–1227, 2009. ISSN 0092-8240. doi: 10.1007/s11538-009-9399-5. URL http://dx.doi.org/10.1007/s11538-009-9399-5.

[38] I. Ramis-Conde, M. A. J. Chaplain, and A. R. A. Anderson. Mathematical modelling of cancer cell invasion of tissue. *Math. Comput. Modelling*, 47:533–545, 2008.

[39] K. A. Rejniak. A single cell approach in modeling the dynamics of tumor microregions. *Math. Biosci. Eng.*, 2:643–655, 2005.

[40] K. A. Rejniak. An immersed boundary framework for modelling the growth of individual cells: an application to the early tumour development. *J. Theor. Biol.*, 247:186–204, 2007.

[41] K. A. Rejniak and R. H. Dillon. A single cell-based model of the ductal tumour microarchitecture. *Computational and Mathematical Methods in Medicine*, 8(1): 51–69, 2007.

[42] K. A. Rejniak and A. R. A. Anderson. A computational study of the development of epithelial acini. sufficient conditions for the formation of a hollow structure. *Bulletin of Mathematical Biology*, 70(3):677–712, 2008. ISSN 0092-8240.

[43] R. Dillon, M. Owen, and K. Painter. A single-cell based model of multicellular growth using the immersed interface method. *Khoo, BC., et al., editors. Contemporary Mathematics: Moving Interface Problems and Applications in Fluid Dynamics, AMS, Providence*, 466:1–16, 2008.

[44] Y. Kim, M. Stolarska, and H. G. Othmer. A hybrid model for tumor spheroid growth in vitro I: Theoretical development and early results. *Math. Models Method Appl. Sci.*, 17:1773–1798, 2007.

[45] B. Ribba, T. Colin, and S. Schnell. A multiscale mathematical model of avascular tumor growth to investigate the therapeutic benefit of anti-invasive agents. *Theor. Biol. Med. Model.*, 3:7, 2006.

[46] A. R. A. Anderson. A hybrid mathematical model of solid tumour invasion: The importance of cell adhesion. *Math. Med. Biol.*, 22:163–186, 2005.

[47] P. Gerlee and A. R. A. Anderson. A Hybrid Cellular Automaton Model of Clonal Evolution in Cancer: The Emergence of the Glycolytic Phenotype. *J. Theor. Biol.*, 250:705–722, 2008.

[48] S. R. McDougall, A. R. A. Anderson, M. A. J. Chaplain, and J. A. Sherratt. Mathematical Modelling of flow through vascular networks: Implications for tumour-indeuced angiogenesis and chemotherapy strategies. *Bull. Math. Biol.*, 64:673–702, 2002.

[49] S. R. McDougall, A. R. A. Anderson, and M. A. J. Chaplain. Mathematical Modelling of dynamic adaptive tumour-induced angiogenesis: Clinical applications and therapeutic targeting strategies. *J. Theor. Biol.*, 241:564–589, 2006.

[50] A.R.A. Anderson. A Hybrid Multiscale Model of Solid Tumour Growth and Invasion: Evolution and the Microenvironment. In A.R.A. Anderson, M.A.J. Chaplain, and K.A. Rejniak, editors, *Single-Cell-Based Models in Biology and Medicine*, pages 3–28. Birkhäuser, Basel, 2007.

[51] I. Ramis-Conde, M. A. J. Chaplain, A. R. A. Anderson, and D. Drasdo. Multiscale modelling of cancer cell intravasation: the role of cadherins in metastasis. *Phys. Biol.*, 6:016008, 2009.

[52] D. K. Schlueter, I. Ramis-Conde, and M. A. J. Chaplain. Computational modeling of single-cell migration: The leading role of extracellular matrix fibers. 2012. URL http://dx.doi.org/10.1016/j.bpj.2012.07.048.

[53] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang. Scaling hypre's multigrid solvers to 100,000 cores. In M. W. Berry, K. A. Gallivan, E. Gallopoulos, A. Grama, B. Philippe, Y. Saad, and F. Saied, editors, *High-Performance Scientific Computing*, pages 261–279. Springer London, 2012. ISBN 978-1-4471-2436-8. doi: 10.1007/978-1-4471-2437-5_13. URL http://dx.doi.org/10.1007/978-1-4471-2437-5_13.

[54] URL http://awards.acm.org/bell.

[55] Jack Dongarra et al. The international exascale software project roadmap. *International Journal of High Performance Computing Applications*, 25(I):3–60, 2011.

[56] Top500 supercomputer sites list. URL http://www.top500.org.

[57] A. Petitet, R.C. Whaley, J.J. Dongarra, and A. Cleary. HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. Technical report, Innovative Computing Laboratory, Sep 2000.

[58] P.M. Kogge et al. ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems. *Washington, DC: DARPA Information Processing Techniques Office*, 278, 2008.

[59] Wikipedia. Moore's law — Wikipedia, the free encyclopedia, 2012. URL http://en.wikipedia.org/wiki/Moore%27s_law. [Online; accessed 02-February-2012].

[60] R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, and A.R. LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of*, 9(5):256–268, Oct 1974. ISSN 0018-9200. doi: 10.1109/JSSC.1974.1050511.

[61] M. Cytowski, M. Filocha, J. Katarzyński, and M. Szpindler. Performance Analysis of Parallel Applications on Modern Multithreaded Processor Architectures. *PRACE Whitepaper, available on-line at www.prace-ri.eu*, 2013.

[62] J. Katarzyński and M. Cytowski. Towards Autotuning of OpenMP Applications on Multicore Architectures. *CoRR, abs/1401.4063*, 2014. URL http://arxiv.org/abs/1401.4063.

[63] M. Gschwind, H.P. Hofstee, B. Flachs, M. Hopkin, Y. Watanabe, and T. Yamazaki. Synergistic processing in cell's multicore architecture. *Micro, IEEE*, 26(2):10–24, March 2006. ISSN 0272-1732. doi: 10.1109/MM.2006.41.

[64] Green500 list. URL http://www.green500.org.

[65] M. Cytowski, B. Borucki, and M. Remiszewski. Nautilus - a testbed for green scientific computing. *ERCIM News*, 2009(79), 2009.

[66] M. Cytowski, M. Remiszewski, and I. Soszyński. Astronomical period searching on the Cell Broadband Engine. *Parallel Processing and Applied Mathematics 2009 (book chapter), Lecture Notes in Computer Science*, 6067:507–516, 2010.

[67] M. Cytowski. Analysis of gravitational wave signals on heterogeneous architecture. *PARA 2010, Part I, Lecture Notes in Computer Science, Springer, Heidelberg*, 7134:347–357, 2012.

[68] M. Cytowski and M. Niezgódka. Increasing the Efficiency of the DaCS Programming Model for Heterogeneous Systems. In *Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics - Volume Part I*, PPAM'11, pages 710–719, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-31463-6. doi: 10.1007/978-3-642-31464-3_72. URL http://dx.doi.org/10.1007/978-3-642-31464-3_72.

[69] Ł. Bieniasz-Krzywiec, M. Cytowski, L. Rychlewski, and D. Plewczynski. 3d-hit: fast structural comparison of proteins on multicore architectures. *Optimization Letters*, pages 1–12, 2013. ISSN 1862-4472. doi: 10.1007/s11590-013-0697-3. URL http://dx.doi.org/10.1007/s11590-013-0697-3.

[70] P. W. Coteus, S. A. Hall, T. Takken, R. A. Rand, S. Tian, G. V. Kopcsay, R. Bickford, F. P. Giordano, C. M. Marroquin, and M. J. Jeanson. Packaging the ibm blue gene/q supercomputer. *IBM Journal of Research and Development*, 57(1/2): 2:1–2:13, 2013. ISSN 0018-8646. doi: 10.1147/JRD.2012.2225922.

[71] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, editors. *MPI The Complete Reference*. The MIT Press, Scientific and Engineering Computation, 1995.

[72] B. Chapman, G. Jost, and R. van der Pas, editors. *Using OpenMP, Portable Shared Memory Parallel Programming*. The MIT Press, Scientific and Engineering Computation, 2007.

[73] P. Balaji, D. Buntinas, D. Goodell, W. Gropp, T. Hoefler, S. Kumar, E. L. Lusk, R. Thakur, and J. Larsson Träff. MPI on Millions of Cores. *Parallel Processing Letters*, 21(1):45–60, 2011.

[74] M. Cytowski and M. Bernardini. Scalability analysis and openmp hybridization of a code for direct numerical simulation of a real wing. *PRACE Whitepaper available on-line at www.prace-ri.eu*, 2013. URL `http://prace-ri.eu/IMG/pdf/wp122.pdf`.

[75] OpenCL - the open standard for parallel programming of heterogeneous systems, . URL `https://www.khronos.org/opencl`.

[76] OpenACC - directives for accelerators, . URL `http://www.openacc-standard.org`.

[77] V. Sarkar, W. Harrod, and A. E. Snavely. Software challenges in extreme scale systems. In *In 2009 Conference on Scientific Discovery through Advanced Computing Program (SciDAC)*, June 2009.

[78] V.Sarkar, S.Amarasinghe, D.Campbell, et al. ExaScale Software Study: Software Challenges in Extreme Scale Systems. *Washington, DC: DARPA Information Processing Techniques Office*, 159, 2009.

[79] J. Demmel, L. Grigori, M.F. Hoemmen, and J. Langou. Communication-optimal parallel and sequential qr and lu factorizations. Technical Report UCB/EECS-2008-89, EECS Department, University of California, Berkeley, Aug 2008.

[80] J. Demmel, L. Grigori, and H. Xiang. Communication-Avoiding Gaussian Elimination. In *Proc. of Supercomputing 2008*, 2008.

[81] M. Mohiyuddin, M. Hoemmen, J. Demmel, and K. Yelick. Minimizing Communication in Sparse Matrix Solvers. In *Proc. of Supercomputing 2009*, 2009.

[82] U.S. Defense Advanced Research Projects Agency. High Productivity Computing Systems Website. *http://www.highproductivity.org*, 2002.

[83] M. Weiland. Chapel, fortress and x10: novel languages for hpc. Technical report, Technical Report from the HPCx Consortium, EPCC, The University of Edinburgh, Oct 2007.

[84] M. D. Garrett. Cell cycle control and cancer. *Curr. Sci.*, 81:515–522, 2001.

[85] J. Galle, M. Loeffer, and D. Drasdo. Modelling the Effect of Deregulated Proliferation and Apoptosis on the Growth Dynamics of Epithelial Cell Populations In Vitro. *Biophysical Journal*, 88:62–75, 2005.

[86] R. E. Mahaffy, C. K. Shih, F. C. MacKintosh, and J. Kas. Scanning Probe-Based Frequency-Dependent Microrheology of Polymer Gels and Biological Cells. *Physical Review Letters*, 85(4):880–883, 2000.

[87] A.J. Maniotis, C.S. Chen, and D.E. Ingber. Demonstration of mechanical connections between integrins, cytoskeletal filaments, and nucleoplasm that stabilize nuclear structure. *Cell Biology*, 94:849–854, 1997.

[88] I. Ramis-Conde, D. Drasdo, A. R. A. Anderson, and M. A. J. Chaplain. Modelling the the influence of the E-Cadherin-β-catenin pathway in cancer cell invasion: A multi-scale approach. *Biophys.*, 95:155–165, 2008.

[89] I. Ramis-Conde, M. Chaplain, and A. Anderson. Mathematical modelling of cancer cell invasion of tissue. *Mathematical and Computer Modelling*, 47:533–545, 2008.

[90] P. Gerlee and A. R. A. Anderson. Stability analysis of a hybrid cellular automaton model of cell colony growth. *Phys. Rev. E*, 75:051911, 2007.

[91] Z. Szymańska. *Mathematical modelling of the heat shock response and the involvement of heat shock proteins in cancer development.* PhD thesis, Institute of Biochemistry and Biophysics, Polish Academy of Sciences, 2010.

[92] M. Rybinski, Z. Szymanska, and A. Gambin. On modeling of protein denaturation during heat shock and mechanism of Hsp70 response. *In preparation.*

[93] R.W. Hockney and J.W. Eastwood, editors. *Computer Simulation Using Particles.* McGraw-Hill, New York, 1981.

[94] V. Springel. The cosmological simulation code GADGET-2. *Mon. Not. R. Astron. Soc.*, 364:1105–1134, 2005.

[95] M.S. Warren and J.K. Salmon. A Fast Tree Code for Many-Body Problems. *Los Alamos Science*, 22:88–97, 1994.

[96] K. J. Oh and Y. Deng. An efficient parallel implementation of the smooth particle mesh Ewald method for molecular dynamics simulations. *Computer Physics Communications*, 177:426–431, 2007.

[97] M. Mascagni and A. Srinivasan. Algorithm 806: SPRNG: A Scalable Library for Pseudorandom Number Generation. *ACM Transactions on Mathematical Software*, 26:436–461, 2000.

[98] G. E. P. Box and Mervin E. Muller. A Note on the Generation of Random Normal Deviates. *Ann. Math. Statist.*, 29:610–611, 1958.

[99] J.J. Monaghan. Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543–574, 1992.

[100] Partnership For Advanced Computing in Europe. URL http://www.prace-ri.eu.

[101] J. Abeles et al. Performance Guide for HPC Applications on IBM Power 775 Systems. *IBM Systems and Technology Group Publication*, 2012.

[102] S. Shende and A. D. Malony. The TAU Parallel Performance System. *International Journal of High Performance Computing Applications, SAGE Publications*, 20(2): 287–331, 2006.

[103] VisNow, a generic visualization framework. URL http://visnow.icm.edu.pl.

[104] POV-Ray - The Persistence of Vision Raytracer. URL http://www.povray.org.

[105] S. Kruś, editor. *Anatomia patologiczna*. PZWL, Warszawa, 2000.

[106] J. Folkman and M. Hochberg. Self-Regulation of Growth in Three Dimensions. *J Exp Med.*, 138:745–753, 1973.

[107] P. Macklin, S. McDougall, A.R.A. Anderson, M.A.J. Chaplain, V. Cristini, and J. Lowengrub. Multiscale modelling and nonlinear simulation of vascular tumour growth. *J. Math. Biol.*, 58:765–798, 2009.

[108] Z. Szymańska and M. Zylicz. Mathematical modelling of heat shock protein synthesis in response to temperature change. *J. Theor. Biol.*, 259:562–569, 2009.

[109] M. Rybiński, Z. Szymańska, S. Lasota, and A. Gambin. Modelling the efficacy of hyperthermia treatment. *Journal of The Royal Society Interface*, 10(88), 2013. doi: 10.1098/rsif.2013.0527. URL http://rsif.royalsocietypublishing.org/content/10/88/20130527.abstract.

[110] J. Barnes and P. Hut. A hierarchical O(NlogN) force-calculation algorithm. *Nature*, 324:446–449, 1986.

[111] J. Dubinski, J. Kim, C. Park, and R. Humble. GOTPM: a parallel hybrid particle-mesh treecode. *New Astronomy*, 9:111–126, 2003.

[112] A. Grama, V. Kumar, and A. Sameh. Scalable parallel formulations of the Barnes–Hut method for n-body simulations. *Parallel Computing*, 24:797–822, 1998.

[113] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM J Sci Stat Comput*, 9:669–686, 1988.