

Multiscale Parallel Simulation of Malignant Pleural Mesothelioma via Adaptive Domain Partitioning – an Efficiency Analysis Study

Anton Dolganov^{1,2}[0009–0001–6014–1812], Valeria Krzhizhanovskaya¹[0000–0002–8247–129X], Stefano Trebeschi^{2,3}[0000–0002–5714–289X], and Vivek M. Sheraton^{1*}[0000–0002–6577–6016]

¹ Computational Science Lab, Informatics Institute, University of Amsterdam, The Netherlands

² Department of Radiology, The Netherlands Cancer Institute – Antoni van Leeuwenhoek Hospital, Amsterdam, The Netherlands

³ GROW – Research Institute for Oncology & Reproduction, Maastricht University, Maastricht, the Netherlands

Corresponding author: v.s.muniraj@uva.nl

Abstract. A novel parallel efficiency analysis on a framework for simulating the growth of Malignant Pleural Mesothelioma (MPM) tumours is presented. Proliferation of MPM tumours in the pleural space is simulated using a Cellular Potts Model (CPM) coupled with partial differential equations (PDEs). Using segmented lung data from CT scans, an environment is set up with artificial tumour data in the pleural space, representing the simulation domain, onto which a dynamic bounding box is applied to restrict computations to the region of interest, dramatically reducing memory and CPU overhead. This adaptive partitioning of the domain enables efficient use of computational resources by reducing the three-dimensional(3D) domain over which the PDEs are to be solved. The PDEs, representing oxygen, nutrients, and cytokines, are solved using the finite-volume method with a first-order implicit Euler scheme. Parallelization is realized using the public Python library `mpi4py` in combination with `LinearGMRESSolver` and `PETSc` for efficient convergence. Performance analyses have shown that parallelization achieves a reduced solving time compared to serial computation. Also, optimizations enable efficient use of the available memory and improved load balancing amongst the cores.

Keywords: Malignant Pleural Mesothelioma · Parallel Simulation · Computational Oncology · Multiscale Modelling

1 Introduction

Despite the rapid progression of technology and advancements in the medical field, cancer remains one of the most complex challenges in modern medicine. The complexity of cancer arises from both its internal heterogeneity and its dynamic interactions with the microenvironment on molecular, cellular and tissue-level, hindering a comprehensive understanding of its intricate nature. Tumour growth is an emerging phenomenon that arises from intercellular signalling dynamics between cancer cells and their microenvironments, consisting of stromal fibroblasts, immune cells, a complex vascular network and a heterogeneous environment of extracellular matrix components (ECM) [20, 18]. Traditional experimental approaches struggle to capture the spatiotemporal complexity of multiscale interactions. This emphasises the importance of computational models that connect molecular mechanisms to tissue-scale behaviours.

Accurate simulations and predictions of cancer progression must incorporate various phenomena on scales from nanometres to millimetres. Intracellular signalling pathways govern cell cycle regulation and adhesion molecule expression, whereas tissue-level forces and nutrient gradients shape collective migration and metastatic potential [1, 12]. For example, hypoxia-driven upregulation of the vascular endothelial growth factor (VEGF), which besides playing a key role in the promotion of angiogenesis also induces epithelial-mesenchymal transition (EMT), allowing individual cells to detach from the primary tumour cell clusters and invade the surrounding tissues [20, 1]. This type of intracellular to tissue transition highlights the necessity for frameworks that bridge low-level molecular signalling with high-level migration patterns. Similarly, metabolic symbiosis in tumours involves a cooperative interaction between hypoxic cancer cells that produce lactate via glycolysis, which in turn is absorbed by oxygenated cancer cells to refuel oxidative phosphorylation, which is essential for tumour cell survival and growth [7]. Capturing these complex interdependencies requires multiscale frameworks that can couple discrete cellular behaviours with continuum-level ODEs/PDEs, a computational challenge that demands unprecedented level and resolution [20, 18].

Traditional serial computing algorithms struggle to handle the computational demands of realistic tumour models. Even 2D multiscale cancer models require extensive computational resources to simulate chemotherapy treatment, which exhausts memory resources, and incurs runtimes that span days or weeks [5]. The Cellular Potts Model (CPM) implemented in platforms like CompuCell3D showcases these exact challenges: At each Monte Carlo Step (MCS), energy functions for millions of lattice sites are iteratively evaluated while tracking chemical diffusion and cell-state interactions [1, 18], whilst at tissue level, reaction-diffusion PDEs are solved for oxygen, nutrients, and growth factors, increasing the computational loads even more. All these intermediate computations of spatial gradients and cell contact histories at each MCS require the data to be stored between the steps, creating serious memory bottlenecks.

Recent studies, which investigated spatial-temporal variations in drug efficacy within 3D tumour spheroids [8] or explored the effects of combined chemother-

apy drugs on tumour dynamics [15], highlight the growing complexity and associated computational intensity of modelling and simulating complicated tumour behaviour. The combination of the heavy computational load with the memory bottleneck makes the serial approach for predictive oncology impractical, especially for more complex computational problems or large 3D environments.

To reduce computational time and resolve the memory bottlenecks, parallel algorithms are commonly applied on multi-core CPUs [9, 11], GPUs and distributed memory clusters [20, 18], to decompose the computational workload across parallel threads or processes [1]. For Agent Based Models (ABMs), parallelization enables simultaneous updates of cell states and (microenvironmental) variables, resulting in a drastic reduction of simulation runtime [17]. Particularly interesting is the distributed-memory framework that uses Message Passing Interface (MPI), which enhances the scalability in high performance computing (HPC) clusters by allowing the spatial domains to be partitioned via inter-process communication. A recent study explored such challenges of parallelizing bacterial biofilm simulations, pointing out how load imbalance and communication overhead can seriously affect simulation efficiency in these frameworks [16]. For further load balancing optimization hybrid approaches that allow the integration of MPI with shared memory can be used, in particular for heterogeneous workloads [1, 10], like discrete cell agents and continuum solvers as mentioned previously [13, 3].

In this paper we propose an adaptive domain partitioning framework with the goal of minimizing the computational load of our 3D computational model for simulating the growth of MPM. This is achieved by (1) focusing only on the regions of interest, namely the region where the MPM is located, in our CPM with continuum PDE solvers; (2) using parallel computing and (3) applying an iterative GMRES solver. We perform a detailed parallel performance analysis with a focus on speed-up, load imbalance and time taken to solve PDEs in combination with the dynamic bounding box strategy. We demonstrate that adaptive partitioning in combination with robust parallelization can significantly enhance simulation efficiency in complex MPM models.

2 Methodology

2.1 Multiscale Model and Computational Domains

Our framework integrates cell-level interactions with tissue-scale dynamics. This multiscale modelling covers a lower-scale process occurring over seconds, namely solute diffusion and reaction processes that shape the nutrient and chemical gradient, while also modelling cellular proliferation that unfolds over hours, resulting in growth and motility over days, linking molecular processes to tumour progression. Building on these capabilities, the goal of this study is to simulate the proliferation of MPM within the pleural space, the region between the parietal and visceral pleura of the lungs. This multiscale workflow is depicted in Fig. 1.

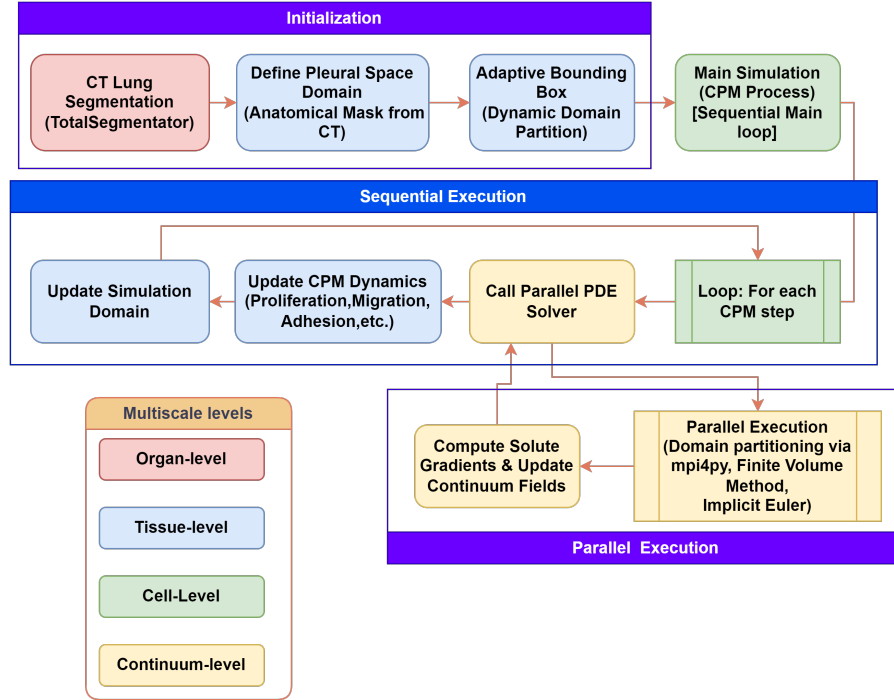


Fig. 1. Flowchart depicting the multiscale modelling workflow, starting from organ-level CT segmentation and progressing through tissue-level domain definition, cell-level CPM dynamics and parallel continuum-level PDE computations. Everything takes place in the CompuCell3D Steppables with exception of the Parallel execution.

The lung data are acquired from segmented CT scans using TotalSegmentator [19, 2] and used in the computational model, which is embedded in the agent-based CompuCell3D (see Fig. 2B). In the pleural space, a cluster of epithelioid cancer cells is initialized to resemble an early stage of MPM (see Fig. 2C).

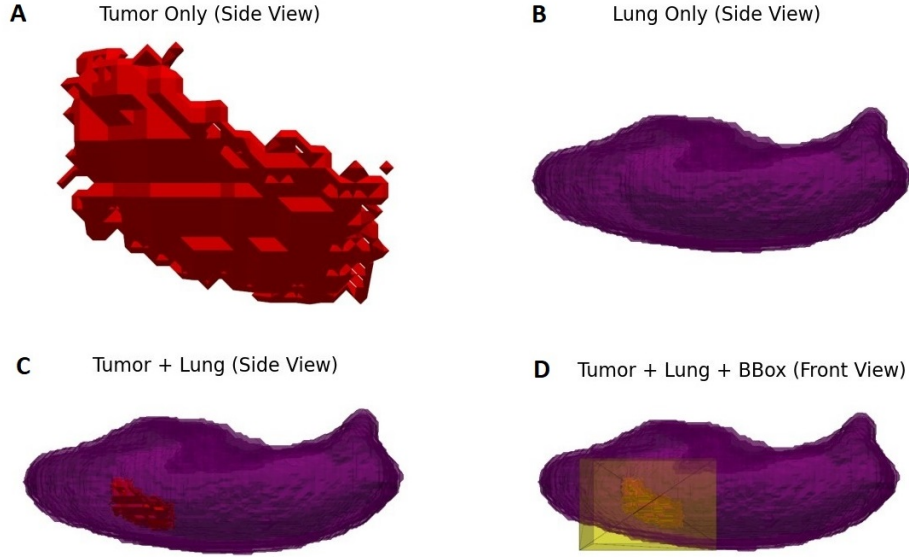


Fig. 2. Computational domain generated from CT scans. (A) Zoomed in MPM tumour. (B) Visceral pleura extracted from CT scan segmentation. (C) Tumour in the pleural space. For visualization purposes the parietal pleura is not shown. (D) Bounding box around the tumour, used to reduce the computational demand.

2.2 Mathematical Models

For the simulation of tumour growth and cell proliferation we implement a CPM, as visualized in Fig.1. The dynamics of oxygen (C_O), nutrients (C_n) and cytokines (IL6 and IL8) is described by a system of partial differential equations:

$$\frac{dC_O}{dt} = D_O \nabla^2 C_O - S_O \quad (1)$$

$$\frac{dC_n}{dt} = D_n \nabla^2 C_n - S_n \quad (2)$$

$$\frac{dC_{IL6}}{dt} = D_{IL6} \nabla^2 C_{IL6} - \mu_{IL6} C_{IL6} \quad (3)$$

$$\frac{dC_{IL8}}{dt} = D_{IL8} \nabla^2 C_{IL8} - \mu_{IL8} C_{IL8} \quad (4)$$

In these equations, C_i represents concentrations, D_i diffusion coefficients, S the consumption rate and μ the degradation rate for oxygen (O), nutrients (n) and the cytokines (IL6 and IL8).

2.3 Bounding Box Mesh

To model the proliferation of tumour cells in the pleural space, a regular 3D mesh was initialized for the spatial oxygen, nutrient and cytokine concentrations. Due to the large size of the lung and consequently the large simulation domain with around 9 million cells (CPM agents), it is unfeasible to perform multiple simulations and parametric variation analyses over the entire environment. To reduce computational complexity, we dynamically track the region of interest, namely the pleural space where the MPM tumour is growing, with an additional margin (applied to the computed bounding coordinates of all tumour cells) to account for diffusion. Dynamical tracking is based on the motility of the tumour and its expanding size, but occurs every 50 MCS to minimize overhead and keep the computational domain as small as possible, while capturing the relevant tumour interactions (see Fig. 2D).

2.4 Implementation

The sequential model is implemented in CompuCell3D, where each cell is represented as an individual agent, and uses the underlying Steppables functionality. Steppables are Python modules that execute specific functions at pre-defined simulation intervals (MCS). Examples of such Steppables that we use are `CellInitializationSteppable` (used to initialize the pleural space and tumour cells), `MitosisSteppable` (used to simulate mitosis) and `FiPySteppable`, the Steppable which performs dynamic domain partitioning and calls the parallel PDE solver.

In the parallel PDE solver we use FiPy version 3.4.5 [6], a finite-volume PDE solver parallelized via `mpi4py` version 4.0.3. [4] over n cores, to solve the PDEs (eq. 1–4) governing the oxygen, nutrient and cytokine diffusion. The spatial domain within the dynamical bounding box is discretized by a structured 3D mesh. Time integration is handled using the first-order implicit Euler scheme and a linear iterative GMRES solver based on the Generalized Minimal Residual method [14]. The resulting output is mapped on the pleural mask, representing the respective oxygen, nutrients, and cytokines concentration distributions.

2.5 Parallelization Strategy

Due to the heavy computational load of the PDE solver, a separate script is developed outside of the sequential Steppables used by CompuCell3D, as can be seen in Fig. 1. This script is called in parallel using `mpi4py` after the `FiPySteppable` initializes the bounding box around the initial tumour in the pleural space and wrote the initial data (mesh, oxygen, nutrients, pleural mask and cytokines) to a shared file.

Within the parallel solver, each process reads the shared file and takes an automatically partitioned domain using FiPy’s internal parallel structure. Once each process acquires a domain, it starts solving for that domain and sends back the updated concentrations upon completion. Once the PDEs solutions are

obtained, the main process will verify the completion and distribute it to the CompuCell3D Steppables for computing cell-level processes such as proliferation, cell death, etc. This approach ensures that the PDE solver can scale to large 3D domains.

To evaluate the efficiency of the parallel implementation, we conducted a detailed performance analysis measuring the following key metrics:

1. The computational time required per timestep for both the parallel ($T_{Parallel}$) and serial (T_{Serial}) implementations;
2. Parallel speedup S_p :

$$S_p = \frac{T_{Serial}}{T_{Parallel}}; \quad (5)$$

3. Parallel efficiency E_p on p cores:

$$E_p = \frac{S_p}{p}; \quad (6)$$

4. Load imbalance $f_{l,i}$, an important metric indicating how evenly the computational work is distributed among the cores:

$$f_{l,i} = \frac{t_i^m - \left(\frac{T_{i,s}}{p}\right)}{\langle t_i^m \rangle} = \frac{t_i^m}{\langle t_i \rangle} - 1, \quad (7)$$

where t_i^m is the maximum time for all p cores and t_i the average time it took for all p cores.

3 Results

In this section we present the performance analysis of the PDE solver under serial and parallel implementations. To compare the two implementations, we ran our simulation on three domain sizes (100^3 , 150^3 and 200^3) with the number of cores used ranging from 1 to 24.

3.1 Computational Time

We start by evaluating our main metric, the computational time required per time step for both the parallel and serial implementations. In Fig. 3 we see how the computational time changes with increasing core counts for the three domain sizes. For all domain sizes, we notice a drastic change in computational time as we switch from a serial to parallel implementation, with the computational time decreasing as we increase the number of cores up to 4.

This effect is stronger for larger domain sizes, with 4 cores proving to be the optimal number for these three domains. Beyond 8 cores, computational time is increasing, which can be explained by the growing communication overhead as the number of subdomains grows.

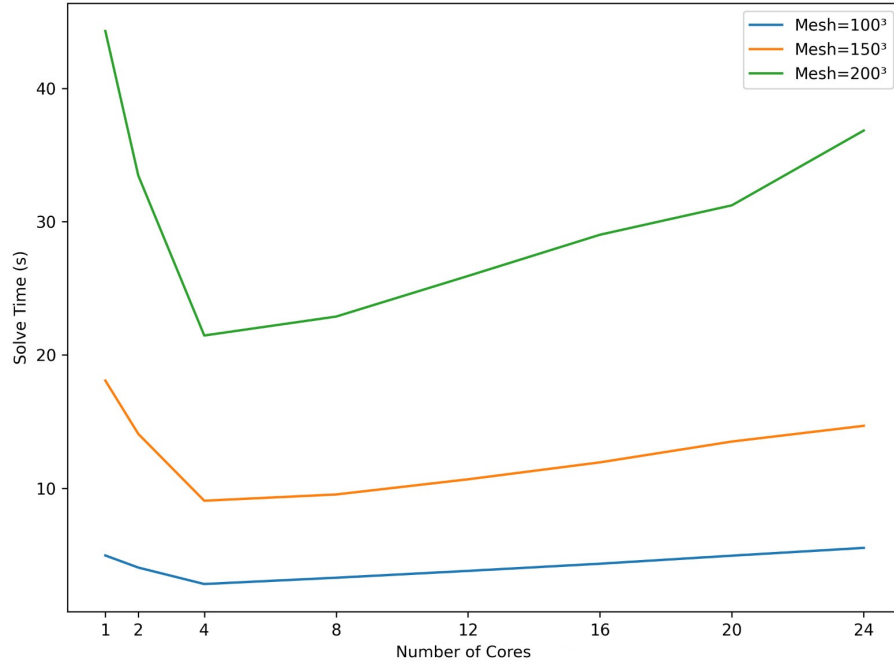


Fig. 3. PDE solver computational time per time step for three different domain sizes using different number of cores

3.2 Parallel Speedup

In Fig. 4 we notice that for smaller number of cores, namely in the range of 2-4, the speedup increases sharply, reaching its peak for 4 cores with a speedup of 1.8 for the mesh of size 100^3 and 1.95 for the mesh of size 200^3 . As the core count increases, the curves start to decline almost linearly. Despite this decline, only for the smallest domain at 24 cores the speedup is below 1, the larger domains maintain a slightly higher speedup due to having more compute work per core before the overhead becomes a dominant factor.

3.3 Parallel Efficiency

In Fig. 5 we see that for all domain sizes, parallel efficiency is close to 60% on 2 cores, slightly decreasing to 50% on 4 cores. With 8 or more cores, parallel efficiency is less than 20%. This decline can be attributed to the ratio of communication time to computation time, which increases as more cores are added. The difference between the three domain sizes is not significant, yet the larger domain size tends to maintain slightly higher efficiency compared to the smaller domains.

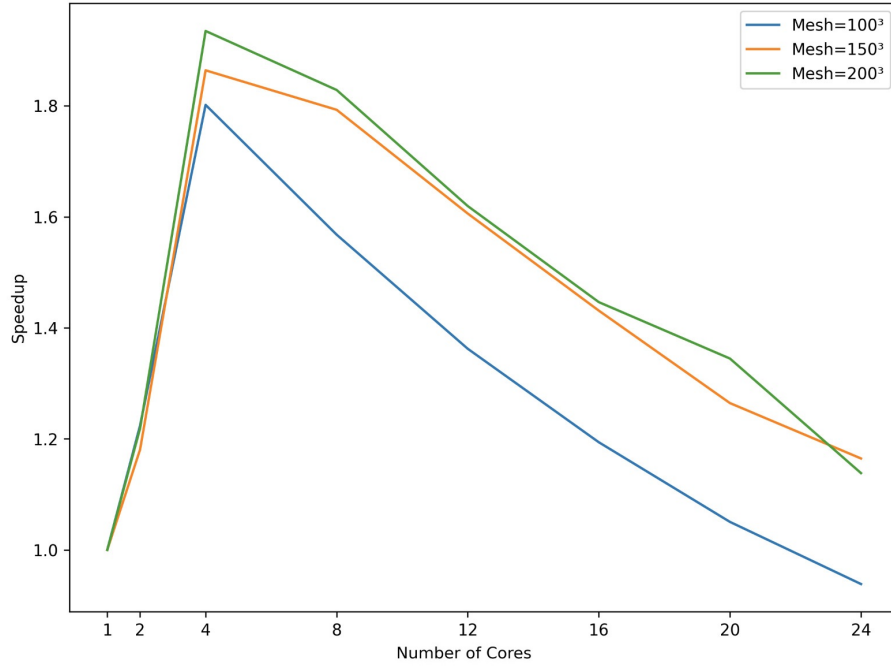


Fig. 4. Parallel speedup for various numbers of cores for the three respective domain sizes.

3.4 Load Imbalance

In Fig. 6 we notice that the simulations using fewer cores, namely 2 and 4, have a steep increase in the fractional load imbalance compared to the initial single-core simulation. This can be attributed to the fact that the assigned sub-domains per core vary in computational effort due to imperfect partitioning or varying boundary effects, resulting in some processes sitting idle whilst waiting for the other process to finish.

As the number of cores increases, the average fractional imbalance starts to approach 1, showing a more uneven distribution of workload among the cores. This waiting time, in turn, drives down the efficiency as can be seen in Fig. 5.

3.5 Impact of Dynamic Bounding Box

The dynamic bounding box plays a key role in reducing computational demand. By prioritizing the region of interest, it limits the number of cells that need to be processed. Solely due to the bounding box, the simulation can be run on a personal computer. Without it, the program cannot handle the computational demand of the whole pleural space and will result in severe performance issues or outright failure. This adaptive domain sizing decreases the memory overhead

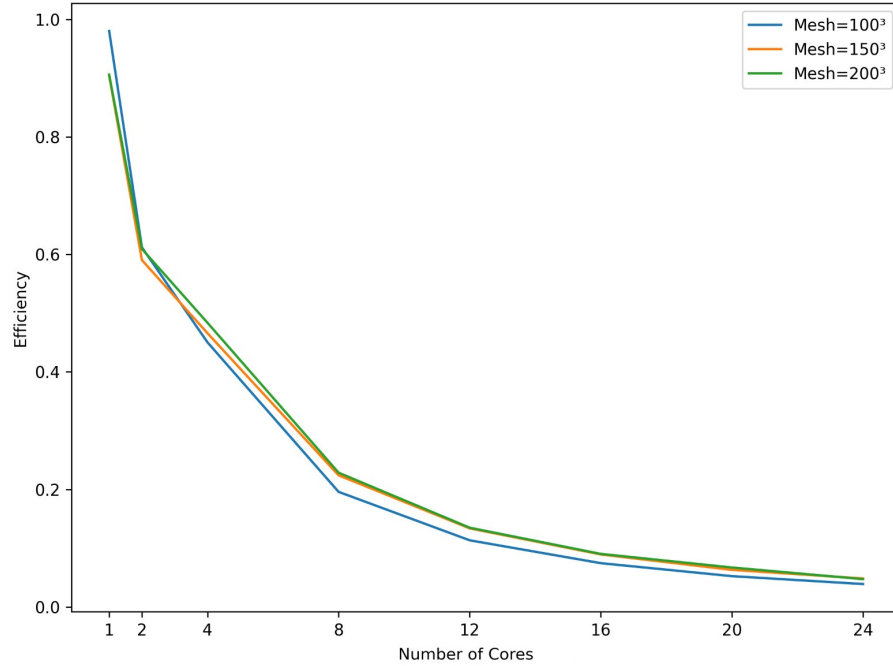


Fig. 5. Parallel efficiency for various numbers of cores for the three respective domain sizes.

and reduces the number of cores required to run the simulation efficiently, thus reducing the overall computational time. Fig. 7 depicts the part of the pleural space inside the bounding box, for which the FiPy calculations are solved.

4 Discussion

Parallel computing can offer many advantages in the acceleration of complex simulations, whether it is working on a complex biofilm simulation or a predictive tumour growth model. Our results demonstrate that parallel implementation can significantly reduce computational time, yet it also comes with limitations.

From our results we can conclude that switching from serial implementation to parallel implementation can reduce the computational time required for solving the PDEs. This general condition holds true for model conditions with cell counts in the range of 1 million to 8 million, with a linear speedup of 2 when using 4 cores.

At the same time, larger parallelization does not generalize to faster solving times as shown in our results (Fig. 4). Use of multiple cores results in larger communication overhead and lower efficiency and speedup (Fig. 5 and 6). The selection procedure for the number of cores to use depends on the domain size,

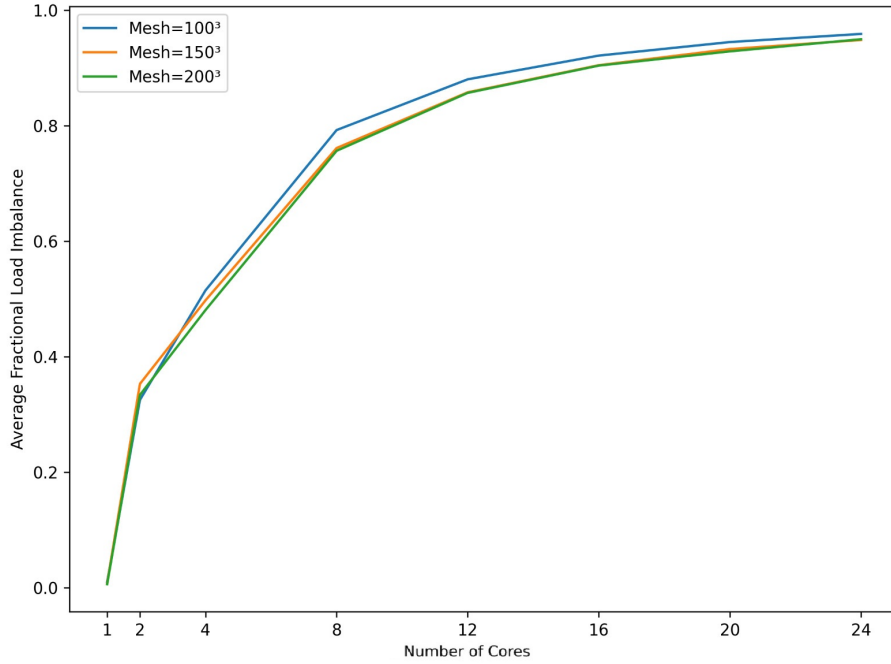


Fig. 6. Load imbalance for various numbers of cores for the three respective domain sizes.

and on the number of variables that need to be solved in that domain, as more variables result in more computational work.

The bounding box strategy has proved to be of key importance for this experiment, as it assisted in keeping the overall domain size manageable. It is important to note that due to its adaptive properties it greatly influences the domain size, and in a simulation where the tumour would grow aggressively and rapidly, the domain size could expand substantially, resulting in larger mesh sizes and consequently larger memory requirements. Thus, for future work it would be of interest to pair the bounding box mechanism with an automated core allocator based on domain size and solute components, to optimize solving speed and domain partitioning, thus acquiring optimal solving speed and memory usage throughout the whole simulation.

While multiple cores can reduce computation time, fine-grained load balancing, minimal interprocess communication, and efficient domain management are necessary to fully optimize parallel efficiency and avoid unwanted overhead.

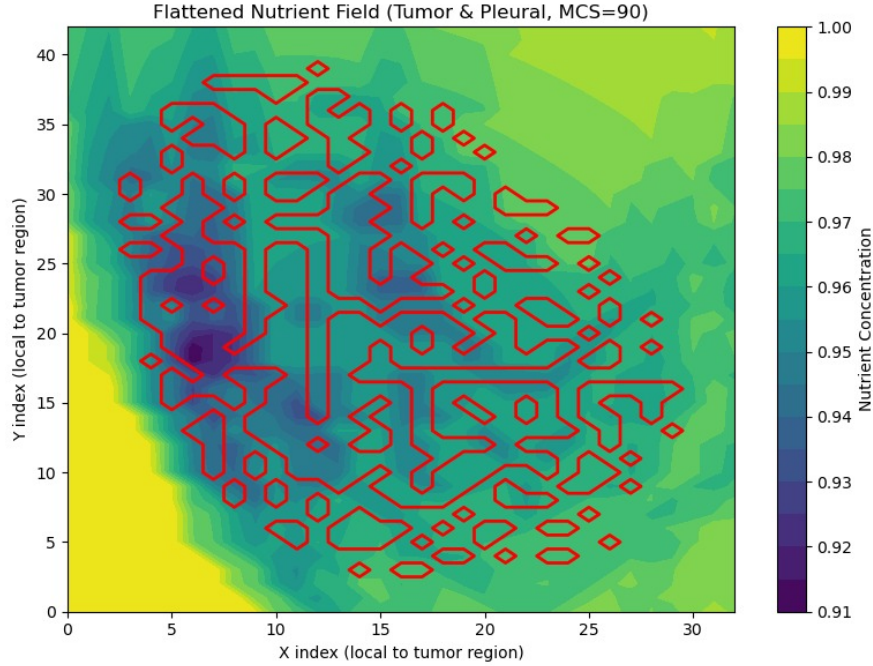


Fig. 7. A contour plot of the normalized nutrient concentrations in the pleural space (a 2D slice is shown), confined by the dynamically set bounding box. The perimeter lines of the tumour cells are highlighted in red.

5 Conclusion

We showed that the utilization of a parallel FiPy solver in a CPM based tumour growth model can accelerate the PDE computational time, with a speedup of 1.8-1.9 on 4 cores on mesh sizes 100^3 - 200^3 . However, it is crucial that attention should be paid to communication overheads, mesh-level processes including domain partitioning and data exchange. The bounding box plays a key role in controlling the computational footprint of complex large-scale 3D computational domains by focusing the required computations on the region of interest within the pleural space.

In the future, it is important to perform a scalability test for larger domains and a higher number of cores, improve load balancing, use adaptive meshing and automatically calculate the optimal number of cores before each execution of an MPI execute call. These improvements will greatly benefit the simulation of large-scale MPM tumours, by allowing more detailed and complex simulations at a lower computational cost.

References

1. Andasari, V., Roper, R.T., Swat, M.H., Chaplain, M.A.: Integrating intracellular dynamics using compucell3d and bionetsolver: applications to multiscale modelling of cancer cell growth and invasion. *PloS one* **7**(3), e33726 (2012)
2. Armato III, S.G., McLennan, G., Bidaut, L., et al.: Data from lidc-idri. The Cancer Imaging Archive (2015). <https://doi.org/10.7937/K9/TCIA.2015.LO9QL9SX>, <https://doi.org/10.7937/K9/TCIA.2015.LO9QL9SX>
3. Chen, N., Glazier, J.A., Izaguirre, J.A., Alber, M.S.: A parallel implementation of the cellular potts model for simulation of cell-based morphogenesis. *Computer physics communications* **176**(11-12), 670–681 (2007)
4. Dalcin, L., Fang, Y.L.L.: mpi4py: Status update after 12 years of development. *Computing in Science & Engineering* **23**(4), 47–54 (2021)
5. Deisboeck, T.S., Wang, Z., Macklin, P., Cristini, V.: Multiscale cancer modeling. *Annual review of biomedical engineering* **13**(1), 127–155 (2011)
6. Guyer, J.E., Wheeler, D., Warren, J.A.: Fipy: Partial differential equations with python. *Computing in Science & Engineering* **11**(3), 6–15 (2009)
7. Jayathilake, P.G., Victori, P., Pavillet, C.E., Lee, C.H., Voukantsis, D., Miar, A., Arora, A., Harris, A.L., Morten, K.J., Buffa, F.M.: Metabolic symbiosis between oxygenated and hypoxic tumour cells: An agent-based modelling study. *PLOS Computational Biology* **20**(3), e1011944 (2024)
8. Kaura, P., Mishra, T., Verma, N., Dalal, I.S., Sheraton, V.: Effects of combined chemotherapeutic drugs on the growth and survival of cancerous tumours—an in-silico study. *Journal of Computational Science* **54**, 101421 (2021)
9. Korkhov, V.V., Krzhizhanovskaya, V.V., Sloot, P.M.: A grid-based virtual reactor: Parallel performance and adaptive load balancing. *Journal of Parallel and Distributed Computing* **68**(5), 596–608 (2008)
10. Krzhizhanovskaya, V.V., Korkhov, V.V.: Dynamic load balancing of black-box applications with a resource selection mechanism on heterogeneous resources of the grid. In: *International Conference on Parallel Computing Technologies*. pp. 245–260. Springer (2007)
11. Krzhizhanovskaya, V.V., Zatevakhin, M.A., Ignatiev, A., Gorbachev, Y.E., Sloot, P.M.: Distributed simulation of silicon-based film growth. In: *International Conference on Parallel Processing and Applied Mathematics*. pp. 879–887. Springer (2001)
12. Ma, C., Gurkan-Cavusoglu, E.: A comprehensive review of computational cell cycle models in guiding cancer treatment strategies. *NPJ Systems Biology and Applications* **10**(1), 71 (2024)
13. Macklin, P., McDougall, S., Anderson, A.R., Chaplain, M.A., Cristini, V., Lowengrub, J.: Multiscale modelling and nonlinear simulation of vascular tumour growth. *Journal of mathematical biology* **58**, 765–798 (2009)
14. Saad, Y., Schultz, M.H.: Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing* **7**(3), 856–869 (1986)
15. Sheraton, M.V., Chiew, G.G.Y., Melnikov, V., Tan, E., Luo, K.Q., Verma, N., Sloot, P.: Emergence of spatio-temporal variations in chemotherapeutic drug efficacy: in-vitro and in-silico 3d tumour spheroid studies. *BMC cancer* **20**, 1–16 (2020)

16. Sheraton, M.V., Sloot, P.M.: Parallel performance analysis of bacterial biofilm simulation models. In: Computational Science–ICCS 2018: 18th International Conference, Wuxi, China, June 11–13, 2018, Proceedings, Part I 18. pp. 496–505. Springer (2018)
17. Stack, M., Macklin, P., Searles, R., Chandrasekaran, S.: Openacc acceleration of an agent-based biological simulation framework. *Computing in Science & Engineering* **24**(5), 53–63 (2022)
18. Swat, M.H., Thomas, G.L., Belmonte, J.M., Shirinifard, A., Hmeljak, D., Glazier, J.A.: Multi-scale modeling of tissues using compucell3d. In: *Methods in cell biology*, vol. 110, pp. 325–366. Elsevier (2012)
19. Wasserthal, J., Breit, H.C., Meyer, M.T., Pradella, M., Hinck, D., Sauter, A.W., Heye, T., Boll, D.T., Cyriac, J., Yang, S., et al.: Totalsegmentator: robust segmentation of 104 anatomic structures in ct images. *Radiology: Artificial Intelligence* **5**(5), e230024 (2023)
20. Wu, M., Swartz, M.A.: Modeling tumor microenvironments in vitro. *Journal of biomechanical engineering* **136**(2), 021011 (2014)