cellular_raza - Novel Flexibility in Design of Agent-Based Models in Cellular Systems*

Jonas Pleyer, Christian Fleck

Freiburg Center for Data Analysis, Modeling and AI
University of Freiburg

21.11.2024

Abstract

This paper uses `cellular_raza` to develop a model with cell-type specific interactions whereby cells self-assemble into regions of similar species which is also known as cell-sorting. We use this model to asses the parallelization performance of the numerical backend at the core of `cellular_raza` and show that values of up to $p = 97.78 \pm 0.14\%$ parallelizable code can be achieved, which indicates a high level of parallelizability.

1 Introduction

`cellular_raza` is a cellular agent-based modeling framework [1] which allows researchers to construct models from a clean slate. In contrast to other agent-based modelling toolkits, it is free of assumptions about the underlying cellular representation. This enables researchers to build up complex models while retaining full control over every parameter and behaviour introduced.
At the 9th bwHPC Symposium 2023, we presented a very compacted variety of applications and analytics of `cellular_raza` [2]. Here, we focus on the cell-sorting example which was also presented but has been refined since then. We refer the interested reader to the documentation website cellular-raza.com where one can find the full documentation, remaining examples and information about the underlying assumptions and methods which we leveraged in the development of `cellular_raza`.

2 Cell Sorting

Cell Sorting is a naturally occuring phenomenon which drives many biological processes [3, 4]. While the biological reality can be quite complex, it is rather simple to describe such a system in its most basic form. The underlying principle is that interactions between cells are specific with respect to their type.

Mathematical Description We assume that our cells are spherical objects which interact via force potentials. The values R_i, R_j are the radii of the cells $(i \neq j)$ interacting with each other. For simplification, we can assume that they are identical $R_i = R_j = R$. The two positions of cells are x_i, x_j and their distance is $r_{i,j} = ||x_i - x_j||$. Furthermore, we assume that the equation of motion is given by

$$\partial_t^2 \vec{x}_i = \vec{F}_i - \lambda \partial_t \vec{x}_i \tag{1}$$

where the first term is the usual force term $\vec{F}_i = -\sum_j \vec{\nabla} V_{i,j}$ obtained by differentiating the given potential and the second term is a damping term which arises due to the cells being immersed inside a viscuous fluid.

*Proceedings of the 9th bwHPC-Symposium 2023. Mannheim Conference Series (MaConf). 2025. [CC BY 4.0]

Parameter	Symbol	Value
Cell Radius	R_i	$6\,\mu\mathrm{m}$
Potential Strength	V_0	$2\,\mu\mathrm{m}^2/\mathrm{min}^2$
Damping Constant	λ	$2\,\mathrm{min}^{-1}$
Interaction Range	ξ	$1.5(R_i+R_j)=3R_i$
Time Stepsize	Δt	$0.2\,\mathrm{min}$
Time Steps	N_t	$10'000$
Domain Size	L	$110\ \mu\mathrm{m}$
Cells Species 1	$N_{C,1}$	800
Cells Species 2	$N_{C,2}$	800

Table 1: This table shows two sections containing parameters and other variables respectively which to fully specify and initialize the system. In total, 1600 cells with random initial positions and zero velocity were placed inside the domain.

Note that we rescaled the units of our parameters in order to normalize our mass to $m = 1$. This means, every parameter can be expressed in units of length and time.
We can assume that interactions between cells are restricted to close ranges and thus enforce a cutoff $\xi \geq R_i + R_j$ for the interaction where the resulting force is identical to zero. We further assume that cells of different species do not attract each other but do repel. To describe this behaviour, we set the potential to zero when $r > R_i + R_j$ (i.e., $\kappa_{i,j} > 1$) and both cells have distinct species type s_i.

$$\kappa_{i,j} = \frac{r_{i,j}}{R_i + R_j} \tag{2}$$

$$U_{i,j} = V_0 \left(\frac{1}{3\kappa_{i,j}^3} - \frac{1}{\kappa_{i,j}} \right) \tag{3}$$

$$V_{i,j} = \begin{cases} 0 & \text{if } \kappa_{i,j} \geq \xi/(R_i + R_j) \\ 0 & \text{if } s_i \neq s_j \text{ and } \kappa_{i,j} \geq 1 \\ U(\kappa_{i,j}) & \text{else} \end{cases} . \tag{4}$$

Parameters In total, we are left with only 4 cellular parameters to describe our system. In order to fully specify the system, we also need to define the time for which we are solving, the overall physical size and the initial number of agents which we put into the simulation. Their values are given in Table 1.

Results Figure 1 shows the inital placement of the cells and their final state. We can clearly see that the cells have assembled into connected regions of the same species. The size of these regions depends on the interaction range and its strength. In this example, we did not assume any stochastic motion of the cells. It is to be noted that this assumption is not required by `cellular_raza` but rather a free choice of the model. Depending on the desired complexity, users can substantially modify the cellular representation, which is the main reason for the development of `cellular_raza`. We provide two additional examples which implement mechanics with Brownian [5] and Langevin [5, 6] dynamics but even more complex cellular representations are possible as well (see eg. bacterial-rods).

3 Multithreading Performance (Amdahl's Law)

Theory One measure of multithreaded performance is to calculate the possible theoretical speedup given by Amdahl's law [7]. It provides an estimate for the speedup and assumes that the workload can be split into a parallelizable and non-parallelizable part which is quantified by $0 \leq p \leq 1$. A higher value means that the contribution coming from non-parallelizable algorithms is lower. The theoretical maximum $p = 1$ means that all of the executed code is parallelizable. Amdahl's law is given by

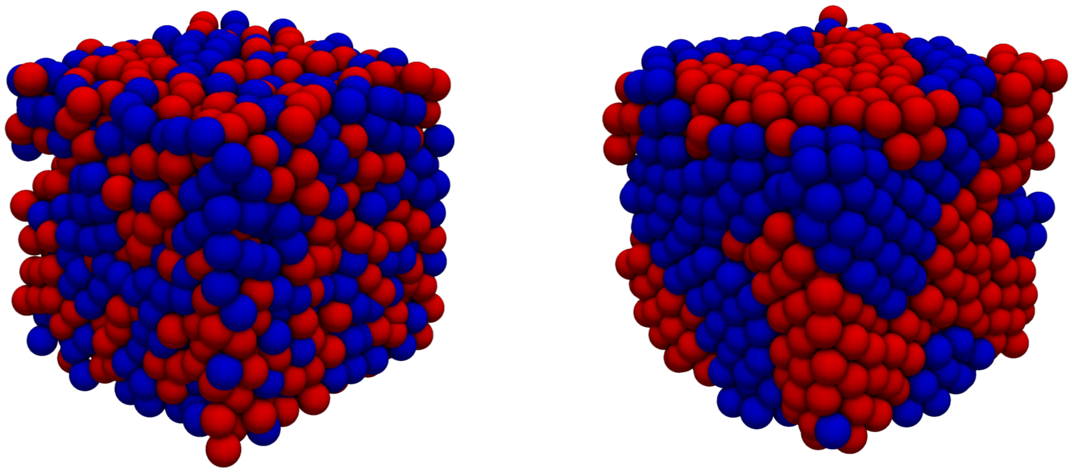

Figure 1: Cells are initially placed randomly inside the cuboid simulation domain. After the simulation has finished, the cells have assembled into connected regions of the same species.

$$T(n) = T_0 \frac{1}{(1-p) + \frac{p}{n}} \tag{5}$$

where $T(n)$ describes the throughput which can be achieved given n parallel threads and the variable p is the relative proportion of execution time which benefits from parallelization. The total latency of a program can be determined via the inverse of the throughput.

Simulation Setup Measuring the performance of any simulation will be highly dependent on the specific cellular properties and complexity. For this comparison, we chose the previously explained cell-sorting example which contains minimal complexity compared to other examples (see cellular-raza.com/showcase). Any computational overhead which is intrinsic to `cellular_raza` and not related to the chosen example would thus be more likely to manifest itself in performance results.
In order to produce reproducible results and simplify this overall process, we provide the cellular_raza-benchmarks crate. It is a command-line utility which can be used to run benchmarks with various configurations. Its arguments are displayed in Listing 1.

CPU	Fixed Clockspeed	Memory Frequency	TDP
AMD Ryzen 3700X [8]	$2200\,\mathrm{MHz}$	$3200\,\mathrm{MT/s}$	$65\,\mathrm{W}$
AMD Ryzen Threadripper 3960X [8]	$2000\,\mathrm{MHz}$	$3200\,\mathrm{MT/s}$	$280\,\mathrm{W}$
Intel Core i7-12700H [9]	$2000\,\mathrm{MHz}$	$4800\,\mathrm{MT/s}$	$45\,\mathrm{W}$

Table 2: List of tested hardware configurations.

```
# cd cellular_raza-benchmarks
# cargo run -- -h
cellular_raza benchmarks

Usage: cell_sorting [OPTIONS] <NAME> [COMMAND]

Commands:
  threads   Thread scaling benchmark
  sim-size  Simulation Size scaling benchmark
  help      Print this message or the help of the given subcommand(s)

Arguments:
  <NAME>  Name of the current runs such as name of the device to be benchmarked

Options:
  -o, --output-directory <OUTPUT_DIRECTORY>
          Output directory of benchmark results [default: benchmark_results]
  -s, --sample-size <SAMPLE_SIZE>
          Number of samples to be generated for each measurement [default: 5]
      --no-save
          Do not save results. This takes priority against the overwrite settings
      --overwrite
          Overwrite existing results
      --no-output
          Disables output
  -h, --help
          Print help
  -V, --version
          Print version
```

Listing 1: Usage of the benchmark CLI tool. We provide two benchmarks, one for increasing the number of agents and another for increasing the number of threads. The subcommands can be further customized and will automatically run the given simulation multiple times for the specified configurations.

Results generated in this way are stored inside the `benchmark_results` folder. In addition, we provide a python script `plotting/cell_sorting.py` to quickly visualize the obtained results.

Hardware This benchmark was run on three distinct hardware configurations. There exists a wide range of variables which could influence our measured runtime results. However, we expect that the biggest effects are due to power-limits and variable frequency of the central processing unit (CPU) (see Figure 2). Both of these effects can be circumvented by choosing an artificially fixed frequency which is low enough such that the total power limit of the CPU is never reached even when multiple cores are under load. While it is well known that other aspects such as cache-size and memory latency can have an impact on absolute performance, they should however not introduce any significant deviations in terms of relative performance scaling.

Results In figure 2, we fit Amdahl's law of equation 5 to our measured datapoints and obtain the parameter p from which the theoretical maximal speedup S can be calculated via

$$\lim_{n\to\infty} T(n) = S = \frac{1}{1-p} \tag{6}$$

The values for the maximum theoretical speedup are $S_{\text{3700X}} = 13.64 \pm 1.73$, $S_{\text{3960X}} = 44.99 \pm 2.80$ and $S_{\text{12700H}} = 34.74 \pm 5.05$. Their uncertainty $\sigma(S)$ can be calculated via the standard gaussian propagation

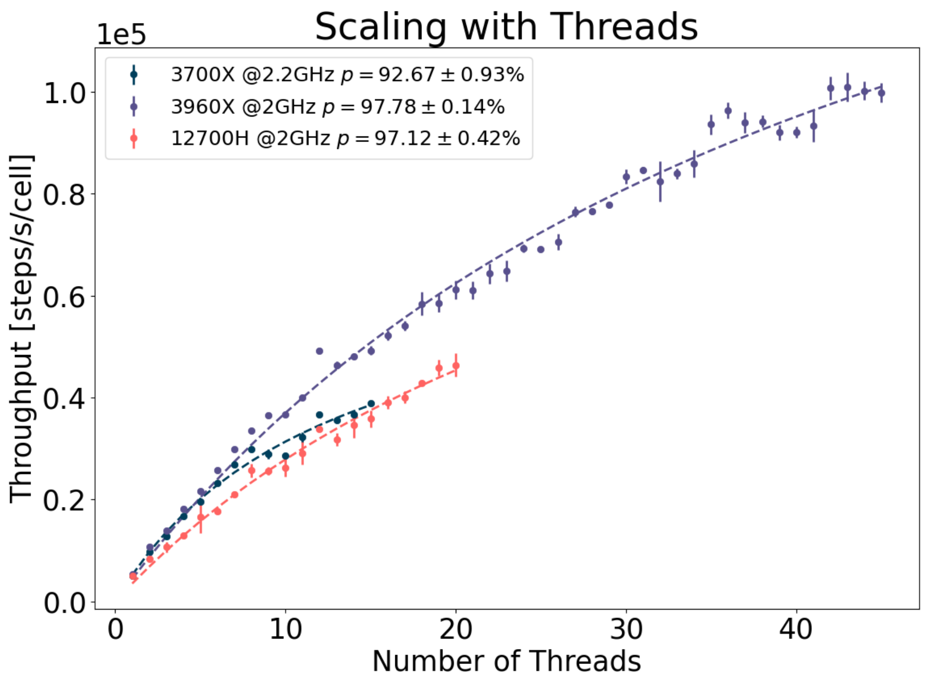

Figure 2: Performance of the throughput $T(n)$ for increasing number of utilized threads n.

$$\sigma(S) = \frac{\sigma(p)}{(1-p)^2} \tag{7}$$

where $\sigma(p)$ is the uncertainty of the parameter p obtained via the fit in figure 2.

Discussion The perfect score of a fully parallelizable system with $p = 1$ is considered almost unobtainable in a real-world scenario due to effects such as the workload of the underlying operating system and physical constraints. Our results showed that the measured value p does also depend on the respective hardware.
In addition to hardware-related effects, we also expect a portion of $1 - p$ of our simulation code to be fundamentally not parallelizable. This fraction can be made up of the initial setup of the simulation which necessarily has to start single-threaded and can only extend to multiple workers once all respective subdomains are generated. Furthermore, ending the simulation not only frees resources which requires computation but also locks the main routine until every individual thread has finished. Even more importantly, all threads are currently using a shared barrier [10] to sync with each other which means that even a single worker can block all others. This limitation could be fixed in the future with improved versions `cellular_raza` and is only an implementation detail and not a fundamental drawback.
However, the total speedup S is still very good for all configurations which can be directly attributed to a good implementation and the core assumption of `cellular_raza` that all interactions are strictly local and subdomains are only interacting along their borders without the need to construct long-ranging synchronization algorithms.

4 Conclusions

We showed along the example of cell-sorting how `cellular_raza` can be used to model cellular biological systems. The modeling process is flexible due to the variety of cellular representations which are supported. We chose to represent our cells with only 4 parameters, which specify the physical representation and interaction

of the cells. The numerical results showed the expected behaviour of cells assembling into connected regions of identical species.
Utilizing the cell-sorting simulation, we benchmarked the performance of the numerical backend. We picked a simulation which is large enough to fully saturate all processors and gradually increased the number of threads utilized. The fractional amount of parallelizable code was determined by fitting the values of runtime to Amdahl's law. It reached values up to $p = 97.78\%$ for a workstation setup which indicates a well-parallelizable implementation.

5 Acknowledgements

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. JP and CF received funding from FET-Open research and innovation actions grant under the European Union's Horizon 2020 (CyGenTiG; grant agreement 801041).

References

[1] J. Pleyer and C. Fleck, "Agent-based models in cellular systems," *Frontiers in Physics*, vol. 10, Jan. 2023. [Online]. Available: http://dx.doi.org/10.3389/fphy.2022.968409

[2] Jonas Pleyer. (2023) bwhpc symposium 2023. [Online]. Available: https://jonaspleyer.github.io/peace-of-posters/showcase/2023-10-23-bwhpc-symposium/

[3] M. S. Steinberg, "Reconstruction of tissues by dissociated cells: Some morphogenetic tissue movements and the sorting out of embryonic cells may have a common explanation." *Science*, vol. 141, no. 3579, p. 401–408, Aug. 1963. [Online]. Available: http://dx.doi.org/10.1126/science.141.3579.401

[4] F. Graner and J. A. Glazier, "Simulation of biological cell sorting using a two-dimensional extended potts model," *Physical Review Letters*, vol. 69, no. 13, p. 2013–2016, Sep. 1992. [Online]. Available: http://dx.doi.org/10.1103/physrevlett.69.2013

[5] T. Schlick, *Molecular Modeling and Simulation*. Springer New York, 2002. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-22464-0

[6] R. W. Pastor, *Techniques and Applications of Langevin Dynamics Simulations*. Springer Netherlands, 1994, p. 85–138. [Online]. Available: http://dx.doi.org/10.1007/978-94-011-1168-3_5

[7] D. P. Rodgers, "Improvements in multiprocessor system design," *ACM SIGARCH Computer Architecture News*, vol. 13, no. 3, p. 225–231, Jun. 1985. [Online]. Available: http://dx.doi.org/10.1145/327070.327215

[8] AMD Ryzen™ 7 3700X. (2024) Amd product specifications. [Online]. Available: https://www.amd.com/en/products/specifications.html

[9] Intel Corporation. (2024) Intel i7 12700h. [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/132228/intel-core-i7-12700h-processor-24m-cache-up-to-4-70-ghz.html

[10] Jon Gjengset, "Hurdles." [Online]. Available: https://crates.io/crates/hurdles