Numerical Methods : Rate equations or MC approaches for modelling growth Write up for a better implementation

Léo BECHET, M2 CompuPhys 2024-2025

I. Introduction

In recent evaluations of our simulation, it became apparent that a more efficient implementation could be achieved. The current design relies on updating each cell of the grid, a process that requires a total of nine passes for each update cycle. However, by leveraging a list-based approach to track only the cells that need updates, we can significantly reduce computational overhead.

II. Principle

While we would still maintain the 2D grid structure to represent the simulation environment, the key optimization lies in tracking only the active cells (those that require updating) rather than iterating over the entire grid. This approach enables targeted updates, leading to performance gains, particularly in sparse simulations where only a fraction of the grid is active.

II.1 Single-Threaded Approach

In single-threaded systems, this optimization could bring substantial improvements. By using a linked list to manage the active cells dynamically, we eliminate the need for predefined arrays and reduce unnecessary cell checks. In large grids, this would greatly decrease the number of cells processed during each cycle.

The process would involve navigating through the list of active cells, updating them as needed, and adding newly activated neighboring cells to the list while marking them to avoid duplicate updates.

II.1.a Update Challenge

One challenge arises when updating a cell that neighbors an inactive or non-aggregated cell. This would require scanning the remaining portion of the list to identify and aggregate these neighboring cells for updates. Efficiently managing this while avoiding duplicate entries or redundant updates is crucial to maintaining performance.

II.2 Multithreaded Approach

For a multithreaded implementation, a queue-based system could replace the linked list, allowing updates to be dispatched to multiple threads in parallel. This approach would distribute the workload across cores, further increasing performance, especially for larger grids or simulations with a high number of active cells.

II.2.a Race Condition Challenge

However, the multithreaded approach introduces potential race condition issues. If two neighboring cells are updated simultaneously by different threads, their interactions could lead to unpredictable results. To mitigate this, the dispatcher would need to implement checks to prevent updates within a defined proximity (e.g., a Z-cell radius) from occurring in parallel. Ensuring thread safety in such a system would be essential to avoid data corruption or erroneous updates.

III. Conclusion

While this targeted-update approach could lead to significant performance improvements, especially in scenarios with relatively few active cells, it may also introduce overhead in simulations with a large number of active cells due to the added complexity of managing the linked list or thread queue. The performance benefits will depend on the nature of the simulation, with sparse grids benefiting the most.

We encourage others in the community to explore this approach, experiment with its implementation, and share their findings, challenges, and optimizations for handling larger-scale simulations.