Start&Stop - a PhysiCell and PhysiBoSS 2.0 add-on for interactive simulation control

Riccardo Smeriglio[1], Roberta Bardini[1*], Alessandro Savino[1], Stefano Di Carlo[1*]

[1]Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Turin, 10129, TO, Italy.

*Corresponding author(s). E-mail(s): roberta.bardini@polito.it; stefano.dicarlo@polito.it;

Abstract

In computational biology, *in silico* simulators are vital for exploring and understanding the behavior of complex biological systems. Hybrid multi-level simulators, such as `PhysiCell` and `PhysiBoSS 2.0`, integrate multiple layers of biological complexity, providing deeper insights into emergent patterns. However, one key limitation of these simulators is the inability to adjust simulation parameters once the simulation has started, which hinders the interactive exploration and adaptation of dynamic protocols ranging from biofabrication to *in vitro* pharmacological testing. To address this challenge, we introduce the `Start & Stop` add-on for `PhysiCell`, and `PhysiBoSS 2.0`. This add-on offers multi-level state preservation and multi-modal stop control, triggered by simulation time or cell conditions, enabling users to pause a simulation, adjust parameters, and then resume from the exact halted state. We validate `Start & Stop` using two well-established `PhysiBoSS 2.0` use cases, a tumor spheroid 3T3 mouse fibroblasts use case under tumor necrosis factor (TNF) stimulation, and a lung cancer invasion simulation, demonstrating that it preserves the simulator's original behavior while enabling interactive configuration changes that facilitate the exploration of diverse and adaptive treatment strategies. By enhancing flexibility and user interaction, `Start & Stop` makes `PhysiCell` and `PhysiBoSS 2.0` more akin to real *in vitro* scenarios, thus expanding the range of potential simulations and advancing more effective protocol development in a variety of applications.

Keywords: Computational Biology, Modeling, In Silico, Simulation, Multi-Scale

1 Background

In computational biology, *in silico* simulators are vital for exploring and understanding the behavior of complex biological systems [1]. Hybrid multi-level simulators integrate multiple layers of biological complexity, providing deeper insights into emergent patterns comprising biological properties and behaviors at different scales and allowing the examination of population dynamics in various environments and under different conditions [3][11]. Among the plethora of available multi-level simulators, `PhysiCell` and `PhysiBoSS 2.0` are two relevant examples [13][4]. In *in silico* pharmacological research, these simulators help users understand and test drug administration schemas more quickly and cost-effectively than traditional trial-and-error methods [14].
`PhysiCell` is a multicellular simulation framework utilizing various solvers, such as a diffusion and transport solver for simulating chemical microenvironment interactions with cells, a mechanical solver for simulating intercellular mechanical interactions, and additional components to simulate cellular processes like growth, division, cell differentiation, and death [9]. To efficiently simulate secretion, diffusion, uptake, and decay of multiple substrates in large three-dimensional (3D) microenvironments, `PhysiCell` employs the `BioFVM` solver [8]. `PhysiBoSS 2.0` extends `PhysiCell` by integrating it with `MaBoSS`, a continuous-time Markovian simulator that models intracellular signaling networks and regulatory processes within cells [17]. It provides a broad range of use cases, from drug treatment analysis of tumors to cell differentiation [13].

In the context of multi-scale simulators like `PhysiCell` and `PhysiBoSS 2.0`, the possibility of monitoring and controlling the simulation over cell conditions is becoming crucial in several applications. Relevant examples include the development of efficient biofabrication protocols for regenerative medicine [6][10][2] and *in silico* pharmacological testing [18]. In the latter case, the ability to monitor and control the simulation to adjust the protocol settings after the simulation has started is crucial. For example, researchers could simulate the adjustment of treatments in response to observed cellular behaviors, making simulations more comparable to real *in vitro* experiments. Additionally, users could simulate biological processes such as tumor progression by dynamically introducing mutations, such as knocking out a critical gene involved in cell proliferation or migration.

Both `PhysiCell` and `PhysiBoSS 2.0` support easy customization of various parameters, such as simulation duration, set of administered stimuli, and initial position of cells. However, monitoring and controlling the simulation once it has started is impossible. To overcome this limitation, this paper introduces the `Start & Stop` add-on for `PhysiCell` and `PhysiBoSS 2.0`, providing:

- *Simulation breakpoints*: enabling to pause the simulation based on the simulation time or predefined cell conditions.
- *Simulation state save and restore*: enabling to save snapshots of the system state after pausing a simulation and to restore them to restart the simulation from a previously saved state.

These new features enhance user control during the simulation and enable a more interactive and adaptive exploration of stimulation protocols.

2 Implementation

The `Start & Stop` add-on allows users to pause a simulation, modify the configuration parameters, and then restart it, preserving the simulation state. The add-on operates on all relevant model layers:

- **Cellular aggregate conformation**: an agent-based model of all cells, their states, and their positions at the core of `PhysiCell`,
- **Intracellular states**: Only available if `PhysiBoSS 2.0` is used. Intercellular states are computed using boolean models of the intracellular regulation network. The `MaBoSS` add-on embedded in `PhysiBoSS 2.0` handles this layer.
- **Microenvironment**: reaction-diffusion Physical Differential Equations (PDEs) that represent the 3D environment divided into 3D voxels. Each voxel is characterized by the concentration of all the relevant diffusing substances.

Figure 1 shows a schema of the integration into `PhysiCell` and `PhysiBoSS 2.0` of the `Start & Stop` add-on. It relies on three primary operations detailed in the following sections: *auto-stop* (Section 2.1), *current state saving* (Section 2.2), and *state restoring* (Section 2.3).

2.1 Auto-stop

Currently, in `PhysiCell` and `PhysiBoSS 2.0`, it is only possible to simulate a use case for a predefined amount of time, without the ability to monitor other parameters, unless the core simulator (i.e., `main.cpp` file) is modified.

Each use case in `PhysiCell` and `PhysiBoSS 2.0` natively includes the `custom.cpp` file, enabling `PhysiCell` and `PhysiBoSS 2.0` users to define custom functions. The `Start & Stop` add-on leverages this file to enable users to define a new function named `auto_stop` implementing a custom *monitoring system* and a set of *stop conditions*. An example of this function is provided in the `PhysiBoSS 2.0` sample use case `spheroid_tnf_model`, located in the file `sample_project_intracellular/boolean/spheroid_tnf_model /custom_modules/custom.cpp`.

With this functionality, we provide the possibility to stop the simulation by monitoring specific parameters instead of setting a predefined simulation.

2.2 Current state saving

To save the current simulation state, the `Start & Stop` add-on implements output streaming operators in the simulator's core. Streaming operators are mechanisms offered by several programming languages that serialize and write complex data objects to external files, enabling the efficient storage of all simulation parameters. This ensures that essential information is preserved and can be reloaded when the simulation is resumed. These operators, have been added in the `PhysiCell` classes such as `cells`, `cell container`, `custom`, `phenotype`, and `microenvironment`, as well as in `MaBoSS`. The `Start & Stop` add-on saves several files containing:

- Information about the *microenvironment*, with each voxel position and the concentration of all substances along with metadata;
- The state of the *cells*;
- The *position* (x, y, and z coordinates) of all the cells in the 3D environment;
- The `PhysiCell` global parameters, such as the current time, the next saving time, and other parameters;
- If using `PhysiBoSS 2.0`, each cell's intracellular state is represented as the state of a Boolean model, if not, the intracellular state is not saved.

Importantly, all generated files are *human-readable*. Files are saved each time the simulation stops. Since the simulations can contain a huge number of cells and agents, in the case of multiple stops, the files are overwritten, thereby reducing storage requirements. The storage occupied by the saved files can vary as a function of the complexity of the simulation; complex simulations generate larger files. Examples of the storage requirements for the use cases presented in this paper are reported in section 3.3

2.3 State restoration

When a simulation is resumed using the `Start & Stop` add-on, the simulator reads all the state files saved at the last stop of the simulation. It restores all objects to the exact state they had when the simulation was interrupted, utilizing input streaming operators. This ensures that the multi-scale state of the entire simulator is fully restored before re-entering the main simulation loop. As a result, the simulation resumes in the exact state it was in when it was stopped. The integration in `PhysiCell` and `PhysiBoSS` is showed in figure 1

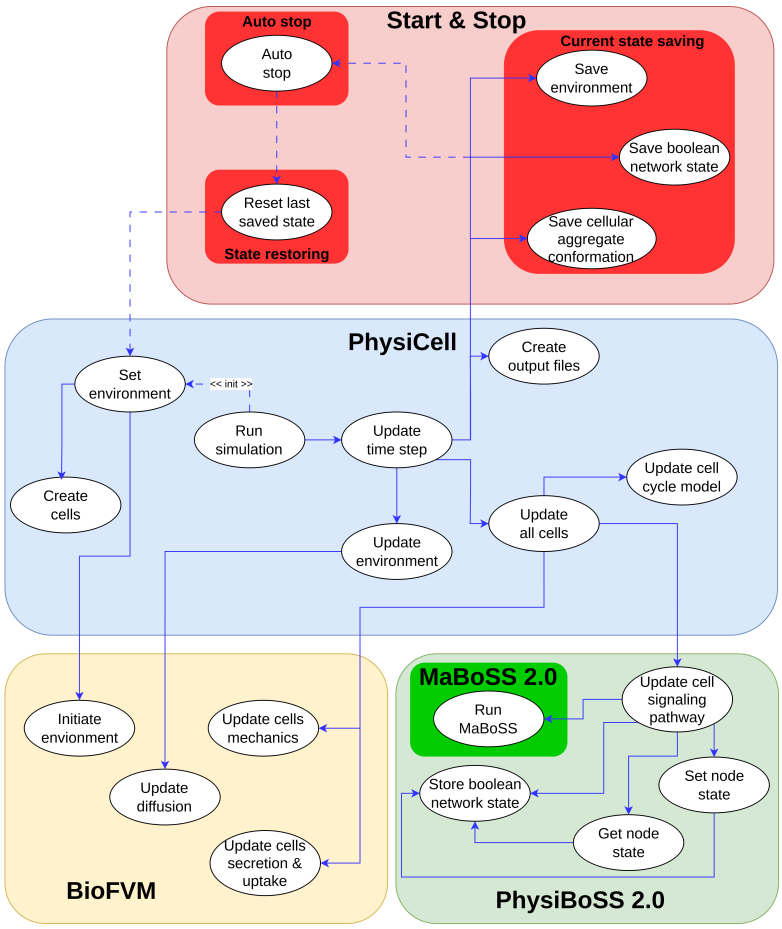

Fig. 1 *Integration of the `Start & Stop` add-on:* This schema represents the integration of the `Start & Stop` add-on into the `PhysiCell` and `PhysiBoSS 2.0` framework (adapted from [13]).

2.4 Usage

To configure all the parameters for the simulation, `PhysiCell` and `PhysiBoSS 2.0` use an `XML` file. The `Start & Stop` add-on extends this file to contain two additional flags: `start_and_stop` and `auto_stop`. To enable the `auto_stop` functionality, the user must set the `auto_stop` flag to `true`. In this way, if the stopping conditions are met during the simulation, the simulation stops and the simulation state is s automatically saved. If the conditions are not met, the simulation continues until the pre-set time is reached. It is important to note that, even in the latter case, the simulation state is saved at the end of the simulation, allowing the user to resume it later. To restart a simulation from the last saved state, the user must set the `start_and_stop` flag to `true`. This ensures that the simulation resumes by loading the last saved state. An illustration of the functioning of the `Start & Stop` add-on in the context of a generic simulation is presented in figure 2

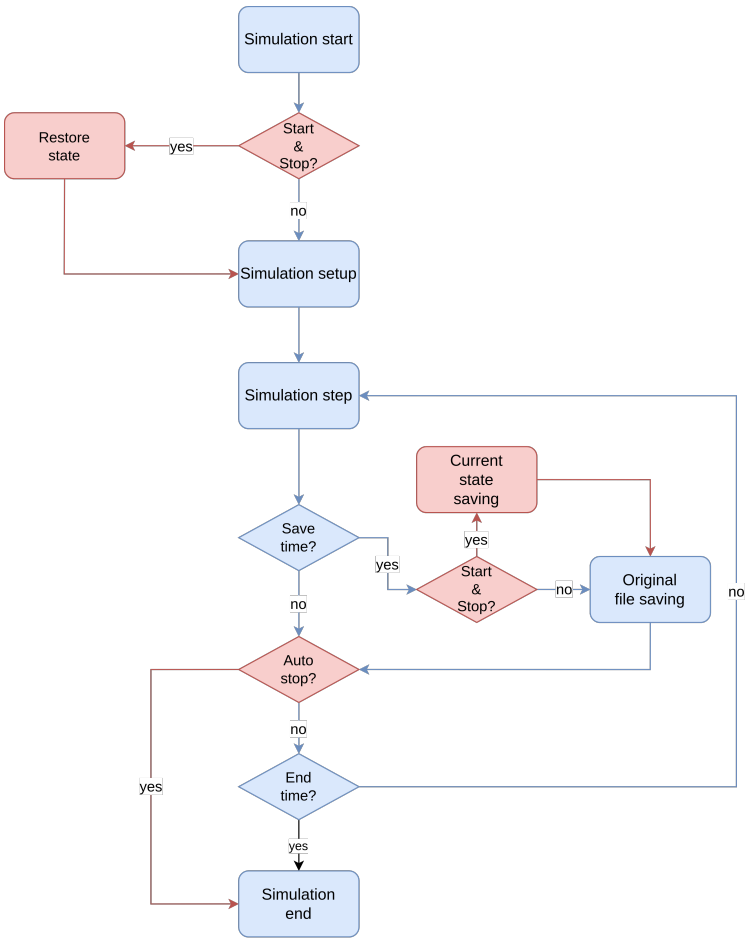

Fig. 2 *`Start & Stop` add-on functioning for a simulation.* This flow chart illustrates a generic simulation in `PhysiCell` and `PhysiBoSS 2.0`, with the inclusion of the `Start & Stop` steps. The blue steps represent the native `PhysiCell` and `PhysiBoSS 2.0` functioning, while the red ones represent the new `Start & Stop` add-on functionalities.

3 Results and Discussion

To demonstrate the functionalities of the `Start & Stop` add-on, we selected two well-established and validated `PhysiBoSS 2.0` use cases. One simulates a tumor spheroid 3T3 mouse fibroblasts under TNF stimulation, while the other simulates a lung cancer invasion.

We selected these two use cases for their ability to support the verification and validation of the `Start & Stop` add-on and to showcase the added value of its new functionalities. In particular, the TNF cancer use case is a well-established and studied model [13]. Although it is not highly complex, featuring a Boolean model with 23 nodes, it includes the ability to select a stimulation protocol. This makes it an

ideal test case for fully exploring the capabilities of the `Start & Stop` add-on, allowing us to evaluate its performance across different protocols and demonstrating its integration with `PhysiBoSS` by enabling protocol modifications during the simulation. Moreover, this use case serves as a valuable test not only for showcasing the capability of the `Start & Stop` add-on but also for verifying that it does not alter the core functionalities of the simulator. The model includes distinct cellular cycle phases that are straightforward to analyze and track, particularly *apoptosis*, *necrosis*, and *survival* states. This characteristic facilitates the assessment of whether stopping and resuming the simulation preserves the expected biological behavior.

The second use case is more complex than the first, incorporating a Boolean model with 48 nodes. In addition to multiple cell cycle phases, this model features different cell types and simulates the transition from an *epithelial* to a *mesenchymal* state. This allows us to test the `Start & Stop` add-on in scenarios involving cell state transitions, further validating its robustness. Although this use case does not include a predefined stimulation protocol, it still demonstrates the add-on's flexibility by enabling the testing of different simulation conditions through modifications of internal cell parameters, such as triggering or inhibiting specific transitions.

This work provides four demonstrative tests to show both the seamless integration with `PhysiCell` and `PhysiBoSS 2.0` and the relevance of the flexibility offered by the `Start & Stop` add-on.

3.1 Use case I: TNF tumor spheroid

This use case examines the cellular dynamics of a tumor spheroid under various TNF regimes [12]. This multi-scale model focuses on 3T3 fibroblast spheroids to explore the complex dynamics observed during *in vitro* drug administration. In particular, the simulation starts with the tumor spheroid composed of alive cells in a proliferative state. Subsequently, the tumor undergoes periodic TNF stimulations that possibly trigger them to transition from a proliferative state towards necrotic or apoptotic death states [13]. Intracellular signaling of the cells underlying the complex response to TNF stimulation relies on the boolean model provided in [5]. When the system undergoes a too-intensive TNF administration, cells can develop resistance to TNF itself, escaping necrotic death, and remaining in a stable state of survival (as described in [5], Figure 2). By default, drug administration occurs in a pulsed manner, where TNF is delivered in pulses with a defined interval, referred to as the *period*. Pulse administration occurs at a fixed *concentration* and *duration*. Users can adjust these parameters for each single simulation; however, TNF stimuli have fixed *period*, *duration*, and *concentration* values along the entire simulation. For the tumor spheroid use case, different protocols, corresponding to different combinations of these parameters, yield varying outcomes, defining the cell state in terms of the number of *alive*, *necrotic*, and *apoptotic* cells (see Section 2.2). In this context, the `Start & Stop` add-on allows users to modify the stimulation protocol during the simulation.

The first step for the validation of the `Start & Stop` add-on is to demonstrate that it is not introducing differences in the functioning of the simulators. To verify this, one approach is to compare the outcomes of simulations using the same protocol, with and without the use of the `Start & Stop` add-on.

Since both `PhysiCell` and `PhysiBoSS 2.0` are stochastic simulators, the outcome of the same simulation can vary from one trial to another, making it challenging to determine if activating the `Start & Stop` feature introduces any differences. To address this, for the TNF tumor use case, we estimated the number of simulations needed to ensure the sample mean differs from the actual distribution mean by no more than ± 10 *alive* cells at the end of the simulation, using a method based on the Standard Error of the Mean (SEM) [15], which yielded 30 simulations. In particular, this method involves computing the SEM as the standard deviation of the sample divided by the square root of the sample size (N). Assuming the data follows an approximately normal distribution, the true mean is expected to lie within $\pm 1.96 \times SEM$ of the sample mean with 95% confidence. By setting this range to 10 cells and estimating the standard deviation from preliminary simulations, we solved for N in the equation $1.96 \times \frac{\sigma}{\sqrt{N}} = 10$, obtaining $N = 30$. To be more conservative, we considered 50 simulations.

As drug administration scheme for testing, we performed two sets of 50 simulations using the same parameters: a TNF injection lasting 10 minutes every 150 minutes at a concentration of 0.005 TNF/μm^3. In the first set, the simulations were executed continuously until completion without interruptions. In the second set, the simulations were stopped at every TNF administration period (every 150 minutes) and then resumed. This comparison allowed us to verify that introducing the `Start & Stop` does not alter the simulator's behavior, demonstrating that the `Start & Stop` add-on does not introduce artifacts into the simulation process. Figure 3 reports the results. Figure 3-a, b, and c represent the box-plot of the distribution of *alive*, *necrotic*, and *apoptotic* cells, respectively, at each time step where the simulations with the `Start & Stop` add-on were stopped and restarted.

Figure 3-a and Figure 3-b show that the behavior of both *alive* and *necrotic* cells is consistent when comparing the simulations with and without the `Start & Stop` activation. Regarding the behavior of *apoptotic* cells (Figure 3-c), it is still consistent. However, in both cases, the standard deviation of the distributions is inherently higher, especially in the middle of simulation courses (between 450min and 1050min). This behavior is because the trend of the *apoptotic* cells in this specific use case of `PhysiBoSS 2.0` show a distribution with a higher variance than that of *necrotic* and *alive* cells. Moreover, the calculations of the minimum number of simulations to perform are based on the *alive* cells alone; thus, the result does not necessarily hold for the other types of cells.

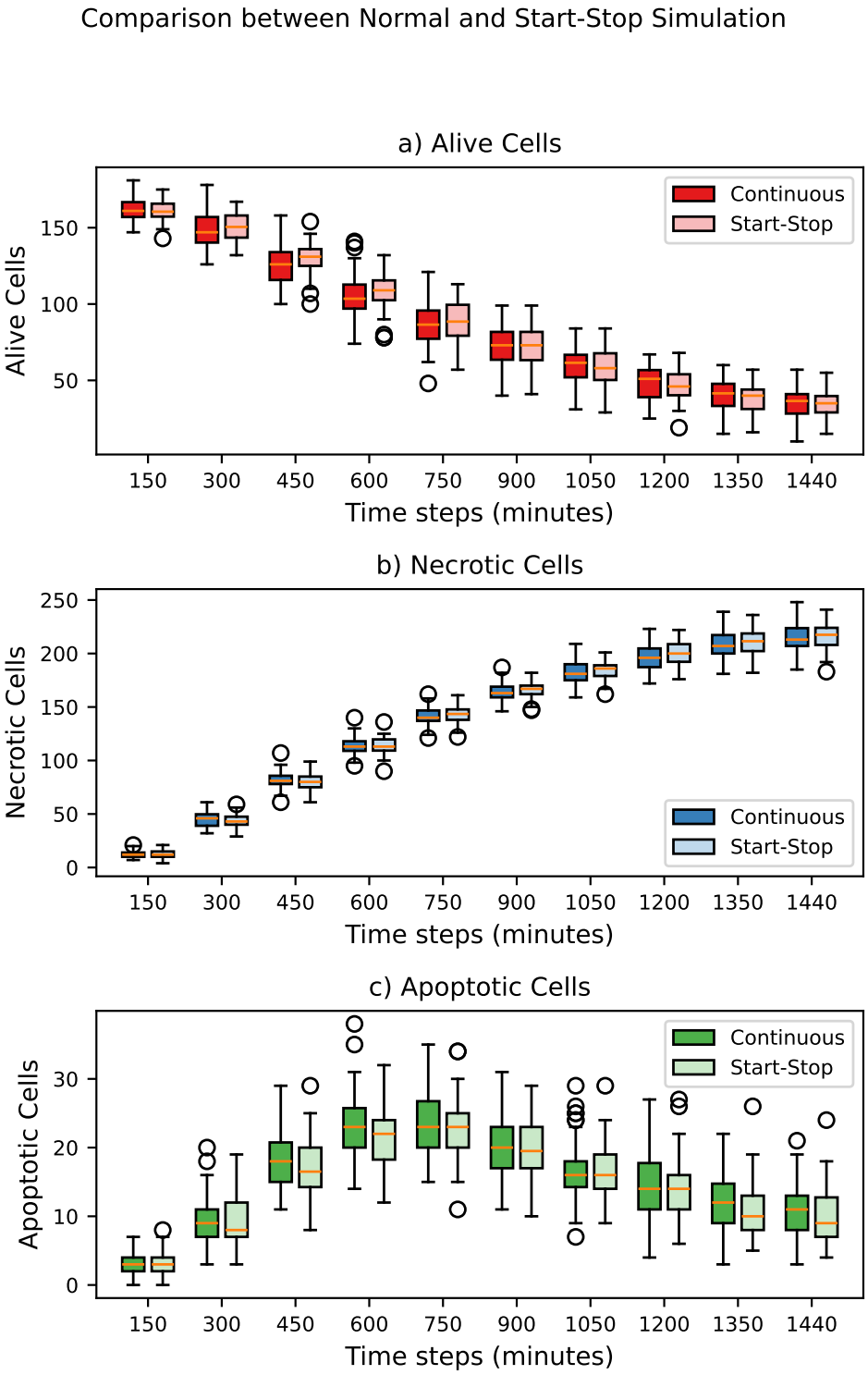

Fig. 3 *Simulation Evolution with and without the* `Start & Stop` *add-on:* a) Comparison between simulations with and without activation of the `Start & Stop` add-on, for a simulation configuration with an administration every 150 minutes of 0.005 TNF/μm^3 lasting 10 minutes, in particular for *alive* cells, b) *necrotic* cells, and c) *apoptotic* cells. Box plots are obtained by running 50 different simulations. In all simulations for the `Start & Stop` test, a stop and a restart occurred at each TNF administration time step.

To provide an applicative example of the relevance of the `Start & Stop` feature in the context of *in vitro* drug administration protocol design, we demonstrate its use for managing the risk of an undesirable outcome of a pharmacological test that is, drug-induced resistance in tumoral cells. In the context of 3T3 fibroblast tumor spheroids, tumor cells can survive treatment in two cases: either the stimulation is not strong enough to induce necrosis, or the cells develop resistance to the effects of the cytokines [5]. Both phenomena are modeled in this `PhysiBoSS 2.0` use case.

Figure 4 illustrates both cases. The figure shows a simulation with continuous TNF administration at a concentration of 0.1 TNF/μm^3. At some point, almost all cells became resistant to the treatment, leading to the continuous growth of living cells, as shown in Figure 4-a. To identify resistant cells, we analyzed the state of their Boolean model and marked as resistant all stable states that resulted in survival, following the criteria described in [5].

To demonstrate the capabilities of the `Start & Stop` add-on, we used the `auto-stop` function (see Section 2.1) to monitor and counteract cell resistance to TNF. Specifically, we created a monitor able to stop the simulation when more than 80% of the living cells acquire a resistant state. Figure 4-b shows that the simulation automatically stopped after 240 simulated minutes. At this point, we adjusted drug administration by providing stimulation every 150 minutes with 0.005 TNF/μm^3 for 10 minutes. This change rapidly reduced cell resistance, allowing the drug to become effective.

Moreover, *in vitro* drug administration can fail due to insufficient stimulation. Figure 4-c illustrates this scenario, where we administered 0.005 TNF/μm^3 every 600 minutes for 10 minutes. Although the *alive* cells did not develop resistance as in Figure 4-a, the treatment remained ineffective. To monitor this issue, we implemented another `auto-stop` function to track the number of *alive* cells and detect continuous tumor growth. As shown in Figure 4-d, the simulation stopped at simulated minute 300. We then modified the drug administration scheme to administrate every 150 minutes with 0.005 TNF/μm^3 for 10 minutes. This adjustment successfully reduced the number of *alive* cells.

These two examples demonstrate the effectiveness and flexibility of the `Start & Stop` add-on.

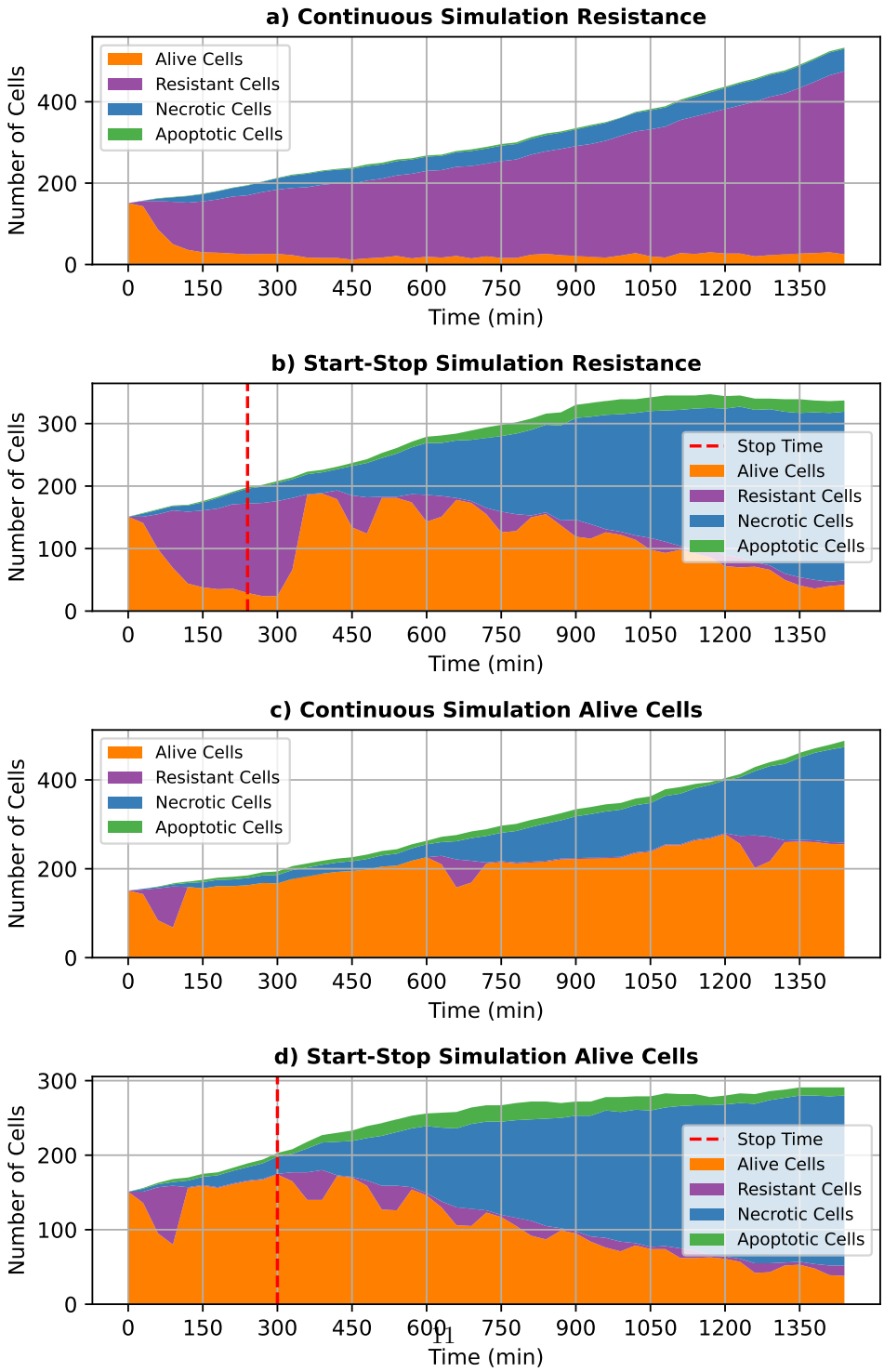

Fig. 4 *Application of the* `Start & Stop` *add-on in an in vitro drug administration Simulation*: a) Behavior of apoptotic, necrotic, alive, and resistant cells during the *in vitro* drug administration with continuous administration of 0.1 TNF/μm^3 over time. Cells develop resistance to the therapy. b) Behavior of alive, apoptotic, necrotic, and resistant cells in a simulation starting similarly to Figure 4-a, but stopped at minute 240 and restarted with administration every 150 minutes of 0.005 TNF/μm^3 for 10 minutes. c) Behavior of apoptotic, necrotic, alive, and resistant cells in a simulation with administration every 600 minutes of 0.005 TNF/μm^3 for 10 minutes, where the *in vitro* drug administration efficacy is insufficient. d) Behavior of alive, apoptotic, necrotic, and resistant cells in a simulation starting similarly to Figure 4-c, but stopped at minute 300 and restarted with administration every 150 minutes of 0.005 TNF/μm^3 for 10 minutes.

3.2 Use case II: Cancer invasion

This use case describes a lung cancer cell invasion model including different cell migration modes, where the underlying intracellular regulations are based on the boolean model provided in [16]. The latter takes as a starting point the boolean model described in [7], which includes two inputs: "ECMenv," which monitors the status of the Extracellular Matrix (ECM), and "DNA_damage," which considers DNA alterations that trigger death signals. To account for the presence of "Oxygen," "GFs," "TGFbeta," and the contact with neighboring cells, the authors of [16] added Four additional inputs. The phenotypes, or outputs, of the model include "CellCycleArrest," "Apoptosis," "EMT," "ECM_adh" (for cell adhesion), "ECM_degrad" (for cell degradation), "Cell_growth" (for tumor growth dynamics), and "Cell_freeze" (for cell motility ability). The authors also included an epithelial-to-mesenchymal transition (EMT) node to simulate the *epithelial*-to-*mesenchymal* transition [16].

Also for this second use case, we ran two sets of 50 simulations. This choice minimized variations in the results while maintaining a reasonable computation time, as suggested in [16]. In the first set, the simulations were executed continuously until completion without interruptions. In the second set, the simulations were stopped and then resumed every 600 minutes to accurately capture the behavior of the involved cells. Figure 5 presents the results. Figure 5-a shows the behavior of the *epithelial* cells, while figure 5-b shows the behavior of *mesenchymal* cells. The distribution of both cell types is consistent with and without the `Start & Stop` add-on with no significant difference highlighted. The results of this comparison again validates that the `Start & Stop` add-on does not alter the function of `PhysiBoSS 2.0`.

In the context of lung cancer invasion, one of the key aspects is the transition from *epithelial* to *mesenchymal* cells. During this process, internal factors such as mutations or external factors such as environmental changes may occur, serving as knockout factors that influence cell transitions. To demonstrate an application of the `Start & Stop` add-on on this use case, we decided to simulate the effect of a mutation that prevents the *epithelial*-to-*mesenchymal* transition. In particular, to achieve this, we introduced an *EMT_Knockout* node in the Boolean model, which inhibits the activation of the *EMT* node, the key regulator triggering the transformation of the cells. The results are shown in Figure 6.

Figure 6-a illustrates the behavior of *epithelial* and *mesenchymal* cells with the *EMT_Knockout* node inactive throughout the simulation. Initially, all cells are *epithelial*, but as the simulation progresses, they begin transitioning into *mesenchymal* cells. Figure 6-b presents a simulation where, during the first 1440 minutes, the *EMT_Knockout* node remains inactive. At 1440 minutes, the simulation is stopped, the `Start & Stop` add-on is activated, and during the state restoring phase, the *EMT_Knockout* node is activated. It is evident that between 1440 minutes and the end of the simulation, the number of *mesenchymal* cells remains stable, while there is no significant change in the behavior of the *epithelial* cells.

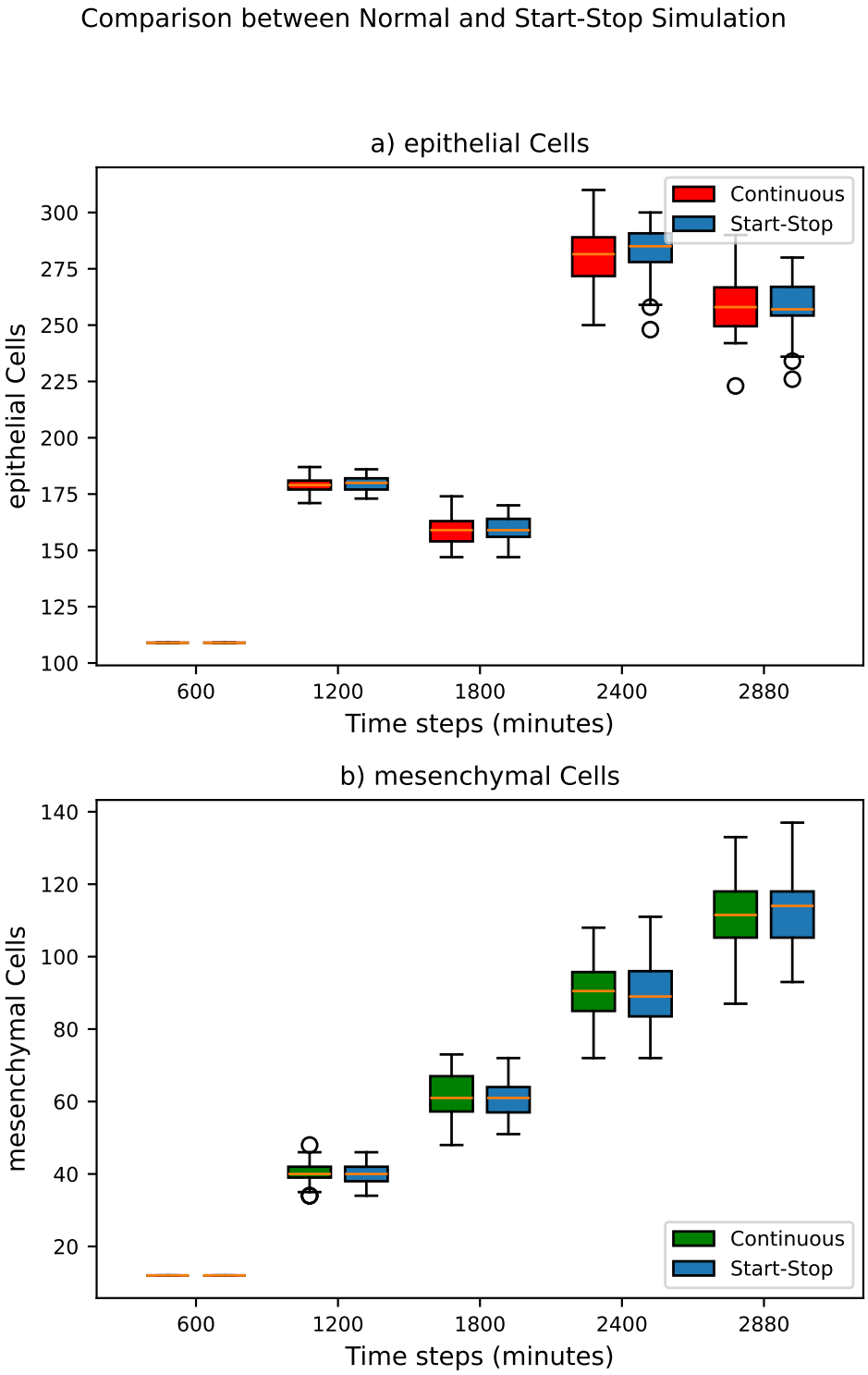

Fig. 5 *Simulation Evolution with and without the* `Start & Stop` *add-on:* a) Comparison between simulations with and without activation of the `Start & Stop` add-on for *epithelial* cells. b) Comparison between simulations with and without activation of the `Start & Stop` add-on for *mesenchymal* cells.

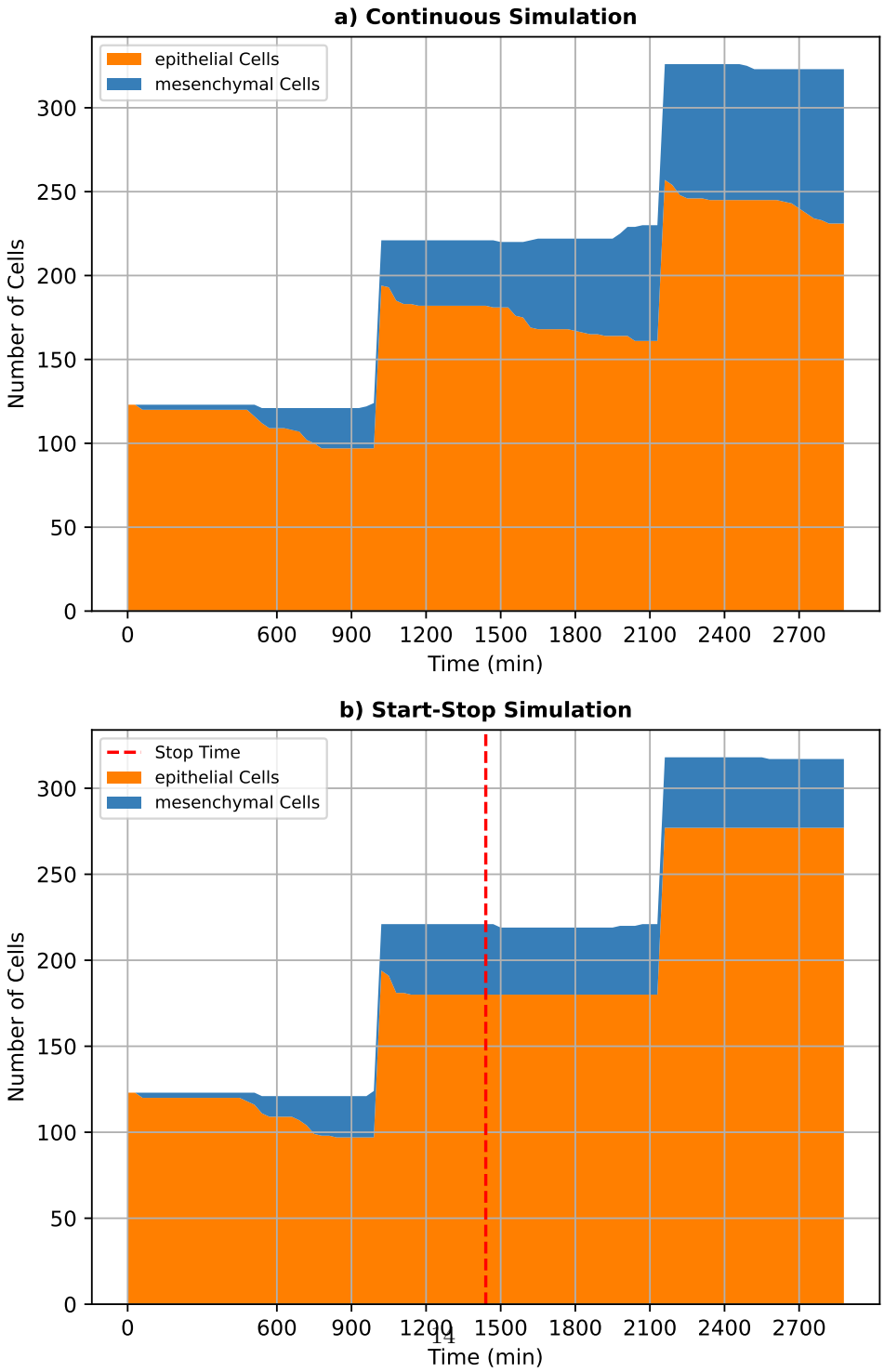

Fig. 6 *Application of the* `Start & Stop` *add-on in cancer invasion simulation*: a) Behavior of *epithelial* and *mesenchymal* cells during the simulation. b) Behavior of textitepithelial and *mesenchymal* cells in a simulation that starts similarly to Figure 6 a but is stopped at 1440 minutes and restarted with the *knockout_EMT* node activated. In this second case, *epithelial* cells stop transforming into *mesenchymal* cells, and the number of *mesenchymal* cells remains constant throughout the second part of the simulation.

3.3 Computational Costs

To further evaluate the impact of the `Start & Stop` add-on, we analyzed the storage and simulation time overhead introduced by the add-on compared to `PhysiCell` and `PhysiBoSS 2.0`.

Regarding the storage required to save the simulation state, it is proportional to the complexity of the model and simulation. In the first use case, at the end of a simulation with TNF administration every 150 minutes at a concentration of 0.005 TNF/μm^3 for 10 minutes, the `Start & Stop` saved files occupied a total of 1.6 MB. Specifically, the simulation stored data for 263 cells, each with 23 nodes in the Boolean model. In the second use case, at the end of a simulation with the predefined initial parameters, the total storage occupied by the `Start & Stop` saved files, was 3.2 MB. Since this use case is more complex than the previously discussed one, with a total of 377 cells and 48 nodes in the boolean model, the occupied storage is twice as large. In general, we can conclude that the overhead in terms of storage is not critical in the considered use cases but must be carefully considered in relation to the complexity of the model.

In terms of time overhead, the time of a `PhysiCell` or `PhysiBoSS` simulation including the `Start & Stop` add-on can be estimaated as follows:

$$T_{\text{total}} = \sum_{i=1}^{n} \left(T^i_{\text{reload}} + T^i_{\text{main}} + T^i_{\text{save}} \right) \tag{1}$$

where n represents the number of simulation stops.

T_{reload} represents the simulation state reload time that depends on the number of cells and the dimension of the model. It can be computed as:

$$T_{\text{reload}} = T_{\text{init}} + T_{\text{S\&S_reload}} \tag{2}$$

with T_{init} representing the time required to initialize the simulation, which also depends on the number of cells and the model size, and $T_{\text{S\&S_reload}}$ representing the time needed to restore the last saved state. Please note that at the beginning of the simulation, $T_{\text{reload}} = T_{\text{init}}$, since no previous state is reloaded.

T_{main} represents the time required for the execution of the main simulation loop.

Finally, T_{save} represents the time needed to save the files for the `Start & Stop` add-on.

The overhead caused by the `Start & Stop` add-on can be defined as:

$$T_{\text{total}S\&S} = \sum_{i=1}^{n} \left(T^i_{\text{reload}} + T^i_{\text{save}} \right) \tag{3}$$

Figures 7 and 8 show the mean CPU time for each simulation phase as a function of the simulated time, for the TNF cancer and cancer invasion use cases, respectively. These times were extracted from the same simulations presented in Figures 3 and 5.

Figure 7 shows that T_{main} represents the most time-consuming phase of the simulation and the overhead introduced to stop and resume the simulation is low (T_{Reload}

and T_{Save}). It is interesting to note that T_{main} gradually decreases as the simulation progresses. This reduction occurs because, as shown in Figure 3, an increasing number of cells are dying and they are skipped during updates, reducing the computational cost of the simulation. Additionally, the simulated time in the last stop is shorter than in the previous ones. Regarding the overhead introduced by the `Start & Stop` add-on, both T_{save} and T_{reload} increase as the total number of cells grows. Although the overhead increases, the most significant computational cost remains T_{main}.

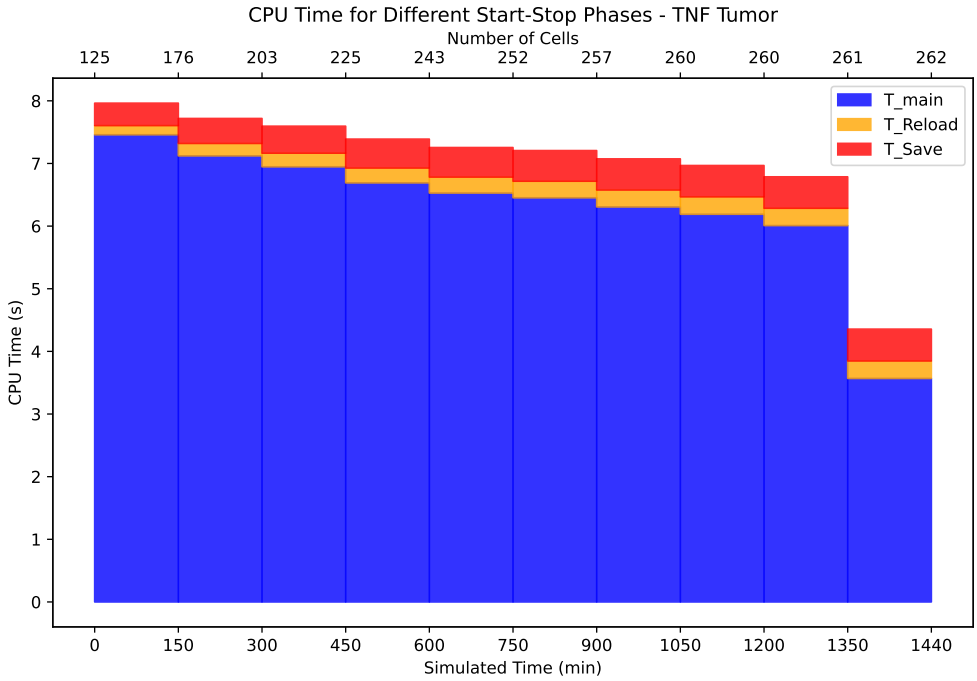

Fig. 7 *CPU time of the TNF cancer simulation as a function of the simulated time*: This plot represents the CPU time required by the different phases of the simulation, shown as a function of both the simulated time and the number of cells at each stop for the TNF cancer use case.

This behavior is even more evident in the cancer invasion use case, shown in Figure 8. Due to its greater complexity, this simulation requires significantly more CPU time for the execution of the main simulation loop, making the overhead introduced by the `Start & Stop` add-on almost negligible. However, as in the previous case, both T_{save} and T_{reload} increase as the total number of cells grows.

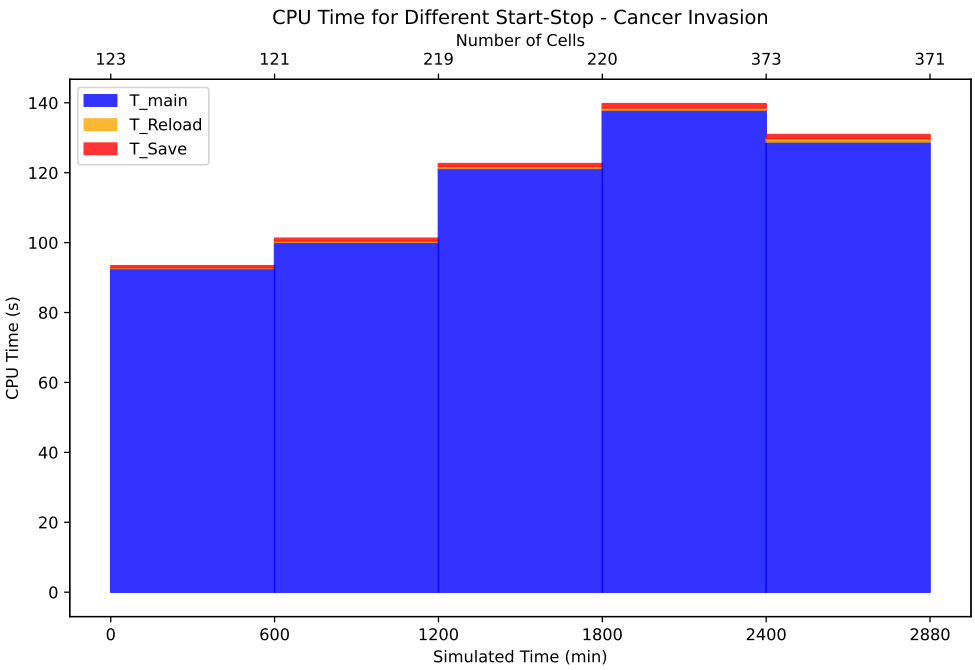

Fig. 8 *CPU time of the cancer invasion simulation as a function of the simulated time*: This plot represents the CPU time required by the different phases of the simulation, shown as a function of both the simulated time and the number of cells at each stop for the cancer invasion use case.

A final comparison between the mean total CPU time of the simulations with and without the `Start & Stop` add-on is shown in Table 1. From this table, it is evident that the inclusion of the `Start & Stop` add-on does not significantly increase the total CPU time of the simulation, as the values for continuous and `Start & Stop` simulations are comparable.

	Time Continuous [s]	**Time Start & Stop [s]**
TNF Cancer	72.97 ± 9.02	71.90 ± 3.43
Cancer Invasion	581.06 ± 16.47	588.84 ± 11.99

Table 1 Comparison of continuous and `Start & Stop` simulations for TNF Cancer and Cancer Invasion.

4 Conclusions and Future Works

In this paper, we presented the `Start & Stop` add-on for `PhysiCell` and `PhysiBoSS 2.0`. This add-on enhances the flexibility of the simulators by:

- enabling the user to stop and restart the simulation from the same state at the point of interruption;
- providing the capability to interact with simulation parameters such as *duration*, *period*, and *concentration* of an *in vitro* drug administration;
- allowing dynamic adjustments to a broader set of parameters, increasing user control and interaction with the simulation;
- enabling the monitoring of the simulation and the execution of necessary actions;
- facilitating the debugging of simulations in case of issues, by allowing the user to monitor and easily interact with saved parameters in an accessible way.

Performed experiments demonstrated that the `Start & Stop` add-on does not change the behavior of a simulation when stopping and restarting, thereby maintaining the validated multi-scale simulation state of the original `PhysiBoSS 2.0` use case implementation. It contributes to the computational ecosystem of hybrid and multi-level models for systems biology by enhancing their representation and predictive power through interactive and automated simulation control within the `PhysiCell` and `PhysiBoSS 2.0` framework.

Current limitations of this work include:

- The fact that the files are saved at each stop and overwritten, as discussed in Section 2.2, reduces storage usage but also decreases the flexibility of the add-on, allowing simulations to be restarted only from the last saved state. To address this limitation, we plan to make the saving phase less impactful in terms of storage consumption and, saving and storing files, more frequently.
- Currently, the only ways to modify certain internal cell parameters, such as their volume or the state of their Boolean model, are:

  1. Manually modifying the parameters inside the saved files. However, this carries the risk of human errors in file modification, potentially causing the tool to crash during file reading.
  2. Implementing a custom function to actively change certain parameters between the state-restoring phase and the start of the simulation, as described in the second use case.

  To address this issue, we plan to develop a user-friendly interface to facilitate the management of saved parameters. This enhancement would significantly expand the range of new simulations allowed, improving the accessibility and usability of the `Start & Stop` add-on.

In conclusion, the `Start & Stop` add-on can significantly expand the capabilities of the `PhysiCell` and `PhysiBoSS 2.0` simulator, making *in silico* experiments closer to *in vitro* experiments.

5 Availability and requirements

5.1 Project name

PhysiS&S

5.2 Project home page

https://github.com/smilies-polito/PhysiSandS

5.3 Operating systems

The project was developed and tested on Linux Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-131-generic x86_64)

5.4 Programming languages

`C++` for the add-on implementation.
`Python` for the test implementation.

5.5 Other requirements

All the dependencies to run the code independently are listed in the https://github.com/smilies-polito/PhysiSandS Repository.

5.6 License

BSD 3-Clause License.

5.7 Any restrictions to use by non-academics

Not applicable.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not Applicable.

Availability of data and materials

The set of functions required for the `Start & Stop` functionality is implemented in `C++` and contained in the `add-ons/start_and_stop` folder, as required by `PhysiCell`. These functions need to be called in the `main.cpp` file of the selected use case. Implemented at the core of `PhysiCell`, it is possible to use the `Start & Stop` add-on for all `PhysiCell` and `PhysiBoSS 2.0` use cases. The full implementation and the script for reproducing the tests provided in this paper are publicly available on GitHub: https://github.com/smilies-polito/PhysiSandS. The add-on code is currently under review by the authors of the `PhysiCell` simulator as part of a pull request to the `PhysiCell` original repository: https://github.com/MathCancer/PhysiCell.

Competing interests

The authors declare no competing interests.

Funding

This study was carried out within the ”SAISEI - Multi-Scale Protocols Generation for Intelligent Biofabrication” [Prot. 20222RT5LC] project and the ”BIGMECH - Knowledge-generation framework combining a novel Biomimetic Investigation platform and high throughput screening for unraveling bone MECHanotransduction mechanisms in view of precision orthopedic medicine” [Prot. 2022F7M2A] project – both funded by European Union – Next Generation EU within the PRIN 2022 program (D.D. 104 - 02/02/2022 Ministero dell'Università e della Ricerca). This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them. This publication is part of the project PNRR which has received funding from the MUR–DM 118/2023.

Authors' contributions

RS developed the code. RS and RB wrote the manuscript. RB, AS, and SDC supervised the work and reviewed the manuscript.

Acknowledgment

We would like to thank Vincent Noël, Miguel Ponce de Leon, Arnau Montagud, and all the `PhysiCell` and `PhysiBoSS 2.0` teams for their insightful suggestions and feedback.

References

[1] Roberta Bardini, Alfredo Benso, Gianfranco Politano, and Stefano Di Carlo. Nets-within-nets for modeling emergent patterns in ontogenetic processes. *Computational and Structural Biotechnology Journal*, 19:5701–5721, 2021.

[2] Roberta Bardini and Stefano Di Carlo. Computational methods for biofabrication in tissue engineering and regenerative medicine-a literature review. *Computational and Structural Biotechnology Journal*, 2024.

[3] Roberta Bardini, Gianfranco Politano, Alfredo Benso, and Stefano Di Carlo. Multi-level and hybrid modelling approaches for systems biology. *Computational and structural biotechnology journal*, 15:396–402, 2017.

[4] Ezio Bartocci and Pietro Lió. Computational modeling, formal analysis, and tools for systems biology. *PLoS computational biology*, 12(1):e1004591, 2016.

[5] Laurence Calzone, Laurent Tournier, Simon Fourquet, Denis Thieffry, Boris Zhivotovsky, Emmanuel Barillot, and Andrei Zinovyev. Mathematical modelling of cell-fate decision in response to death receptor engagement. *PLoS computational biology*, 6(3):e1000702, 2010.

[6] Alberto Castrignanò, Roberta Bardini, Alessandro Savino, and Stefano Di Carlo. A methodology combining reinforcement learning and simulation to optimize the

in silico culture of epithelial sheets. *Journal of Computational Science*, 76:102226, 2024.

[7] David PA Cohen, Loredana Martignetti, Sylvie Robine, Emmanuel Barillot, Andrei Zinovyev, and Laurence Calzone. Mathematical modelling of molecular pathways enabling tumour cell invasion and migration. *PLoS computational biology*, 11(11):e1004571, 2015.

[8] Ahmadreza Ghaffarizadeh, Samuel H Friedman, and Paul Macklin. Biofvm: an efficient, parallelized diffusive transport solver for 3-d biological simulations. *Bioinformatics*, 32(8):1256–1258, 2016.

[9] Ahmadreza Ghaffarizadeh, Randy Heiland, Samuel H Friedman, Shannon M Mumenthaler, and Paul Macklin. Physicell: An open source physics-based cell simulator for 3-d multicellular systems. *PLoS computational biology*, 14(2):e1005991, 2018.

[10] Leonardo Giannantoni, Roberta Bardini, and Stefano Di Carlo. A methodology for co-simulation-based optimization of biofabrication protocols. In *International Work-Conference on Bioinformatics and Biomedical Engineering*, pages 179–192. Springer, 2022.

[11] Zhiwei Ji, Ke Yan, Wenyang Li, Haigen Hu, and Xiaoliang Zhu. Mathematical and computational modeling in complex biological systems. *BioMed research international*, 2017(1):5958321, 2017.

[12] Robin EC Lee, Mohammad A Qasaimeh, Xianfang Xia, David Juncker, and Suzanne Gaudet. Nf-κb signalling and cell fate decisions in response to a short pulse of tumour necrosis factor. *Scientific Reports*, 6(1):39519, 2016.

[13] Miguel Ponce-de Leon, Arnau Montagud, Vincent Noël, Annika Meert, Gerard Pradas, Emmanuel Barillot, Laurence Calzone, and Alfonso Valencia. Physiboss 2.0: a sustainable integration of stochastic boolean and agent-based modelling frameworks. *npj Systems Biology and Applications*, 9(1):54, 2023.

[14] Fateme Pourhasanzade, SH Sabzpoushan, Ali Mohammad Alizadeh, and Ebrahim Esmati. An agent-based model of avascular tumor growth: Immune response tendency to prevent cancer development. *Simulation*, 93(8):641–657, 2017.

[15] Frank E Ritter, Michael J Schoelles, Karen S Quigley, and Laura Cousino Klein. Determining the number of simulation runs: Treating simulations as theories by not sampling their behavior. *Human-in-the-loop simulations: Methods and practice*, pages 97–116, 2011.

[16] Marco Ruscone, Arnau Montagud, Philippe Chavrier, Olivier Destaing, Isabelle Bonnet, Andrei Zinovyev, Emmanuel Barillot, Vincent Noël, and Laurence Calzone. Multiscale model of the different modes of cancer cell invasion. *Bioinformatics*, 39(6):btad374, 2023.

[17] Gautier Stoll, Barthélémy Caron, Eric Viara, Aurélien Dugourd, Andrei Zinovyev, Aurélien Naldi, Guido Kroemer, Emmanuel Barillot, and Laurence Calzone. Maboss 2.0: an environment for stochastic boolean modeling. *Bioinformatics*, 33(14):2226–2228, 2017.

[18] Errikos Streviniotis, Nikos Giatrakos, Yannis Kotidis, Thaleia Ntiniakou, and Miguel Ponce de Leon. Optimizing resource allocation for tumor simulations

over hpc infrastructures. In *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2023.