FiCoS: a fine- and coarse-grained GPU-powered deterministic simulator for biochemical networks

Andrea Tangherloni[1,*], Marco S. Nobile[2,3], Paolo Cazzaniga[1,3], Giulia Capitoli[4], Simone Spolaor[5], Leonardo Rundo[6,7], Giancarlo Mauri[3,5,8], Daniela Besozzi[3,5,8]

1 Department of Human and Social Sciences, University of Bergamo, Bergamo, Italy
2 Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Eindhoven, The Netherlands
3 SYSBIO/ISBE.IT Centre of Systems Biology, Milan, Italy
4 School of Medicine and Surgery, University of Milano-Bicocca, Monza, Italy
5 Department of Informatics, Systems and Communication, University of Milano-Bicocca, Milan, Italy
6 Department of Radiology, University of Cambridge, Cambridge, UK
7 Cancer Research UK Cambridge Centre, University of Cambridge, Cambridge, UK
8 Bicocca Bioinformatics, Biostatistics and Bioimaging Centre (B4), University of Milano-Bicocca, Vedano al Lambro, Italy

* andrea.tangherloni@unibg.it

Abstract

Mathematical models of biochemical networks can largely facilitate the comprehension of the mechanisms at the basis of cellular processes, as well as the formulation of hypotheses that can then be tested with targeted laboratory experiments. However, two issues might hamper the achievement of fruitful outcomes. On the one hand, detailed mechanistic models can involve hundreds or thousands of molecular species and their intermediate complexes, as well as hundreds or thousands of chemical reactions, a situation generally occurring when rule-based models are analysed. On the other hand, the computational analysis of a model typically requires the execution of a large number of simulations for its calibration or to test the effect of perturbations. As a consequence, the computational capabilities of modern Central Processing Units can be easily overtaken, possibly making the modeling of biochemical networks a worthless or ineffective effort. To the aim of overcoming the limitations of the current state-of-the-art simulation approaches, we present in this paper FiCoS, a novel "black-box" deterministic simulator that effectively realizes both a fine- and a coarse-grained parallelization on Graphics Processing Units. In particular, FiCoS exploits two different integration methods, namely the Dormand–Prince and the Radau IIA, to efficiently solve both non-stiff and stiff systems of coupled Ordinary Differential Equations. We tested the performance of FiCoS against different deterministic simulators, by considering models of increasing size and by running analyses with increasing computational demands. FiCoS was able to dramatically speedup the computations up to 855×, showing to be a promising solution for the simulation and analysis of large-scale models of complex biological processes.

Author summary

Systems Biology is an interdisciplinary research area focusing on the integration of biology and *in-silico* simulation of mathematical models to unravel and predict the emergent behavior of complex biological systems. The ultimate goal is the understanding of the complex mechanisms at the basis of biological processes together with the formulation of novel hypotheses that can be then tested with laboratory experiments. In such a context, detailed mechanistic models can be used to describe biological networks. Unfortunately, these models can be characterized by hundreds or thousands of molecular species and chemical reactions, making their simulation unfeasible using classic simulators running on modern Central Processing Units (CPUs). In addition, a large number of simulations might be required to calibrate the models or to test the effect of perturbations. In order to overcome the limitations imposed by CPUs, Graphics Processing Units (GPUs) can be effectively used to accelerate the simulations of these detailed models. We thus designed and developed a novel GPU-based tool, called FiCoS, to speed-up the computational analyses typically required in Systems Biology.

Introduction

The computational analysis of complex biological processes relies on the definition and simulation of mathematical models capable of providing an accurate representation of the emergent dynamics of these processes both in physiological and perturbed conditions [1]. Among others, Reaction-Based Models (RBMs), allow for proposing a mechanistic description of the biological process, being one of the most suitable solutions to achieve a detailed comprehension of the underlying behavior [2] of the cellular system under analysis. Following the research approach of Systems Biology [1], RBMs can be employed within different computational tasks, such as Parameter Estimation, Sensitivity Analysis, and Parameter Sweep Analysis [3-5], to gain insights about the system and drive the design of lab experiments. However, these tasks typically require a huge amount simulations of the same RBM with distinct parameterizations, that is, different initial molecular concentrations and kinetic constants. In addition, when the RBM under investigation is characterized by hundreds or thousands of molecular species and chemical reactions, the single simulation of the dynamics can be extremely demanding, so that the capability of modern Central Processing Units (CPUs) is rapidly overtaken.

In the last decade, many efforts have been made to develop efficient computational tools able to perform massive numbers of simulations of large-scale RBMs. Among others, high-performance computing architectures, like Graphics Processing Units (GPUs), have gained ground and have been widely adopted for the parallelization of a variety of tasks [6]. GPUs are powerful, energy-efficient, and low-cost many-core co-processors, capable of tera-scale performances on common workstations, which can be exploited to dramatically reduce the running time required by traditional CPU-based approaches.

Considering GPU-powered biochemical simulation, the scheme in Fig. 1 summarizes the state-of-the-art. The existing tools can be categorized with respect to two main concepts: (*i*) the simulation granularity and (*ii*) the simulation type [6]. The former category determines how the threads of the GPU can be used: in coarse-grained simulations, each thread corresponds to an independent simulation; in fine-grained simulations, the calculations of a single simulation are distributed over multiple threads. The latter category specifies the type of simulation: deterministic or stochastic. Indeed, different simulation strategies rely on deterministic, stochastic, or hybrid

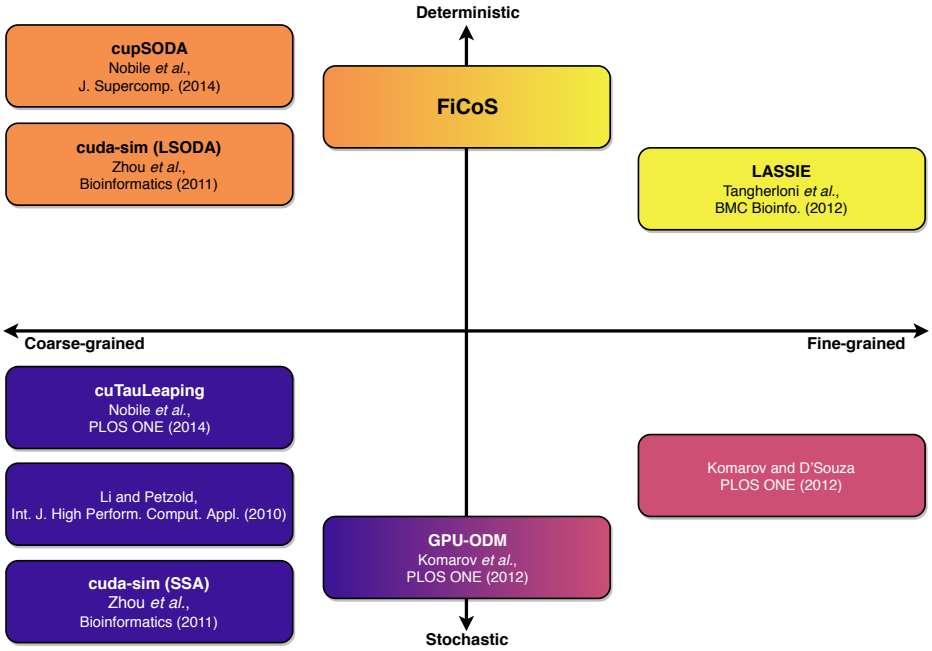

Fig 1. The "semiotic square" of GPU-powered simulators. The horizontal axis represents the parallelization strategy that can be realized to leverage the GPU capabilities: coarse-grained (left), in which multiple simulations are performed in a parallel fashion; fine-grained (right), in which a single simulation is accelerated by distributing the computation over multiple threads. The vertical axis partitions the two types of simulation: deterministic (top) and stochastic (bottom). FiCoS is a deterministic approach implementing both a fine- and a coarse-grained parallelization strategy, filling the gap in the state-of-the-art.

algorithms [7–9].

The first example of GPU-powered simulator was the implementation of the Stochastic Simulation Algorithm (SSA) [10] proposed by Li and Petzold [11]. SSA was later implemented in cuda-sim [12], a Python tool for Systems Biology that also supports deterministic simulation. Finally, Nobile *et al.* proposed in 2014 two further coarse-grained implementations: cupSODA [13], a deterministic simulator based on the Livermore Solver of Ordinary Differential Equations (LSODA) [14], and cuTauLeaping [15], a stochastic simulator based on a variant of the tau-leaping algorithm [16]. It is worth noting that cupSODA has been integrated with PySB [17], a well-known modeling and simulation framework that allows the user to create biological models as Python programs [18]. There are only two purely fine-grained implementations in literature: LASSIE (LArge-Scale SImulator) [19], a fine-grained deterministic simulator tailored on large-scale models (e.g., > 100 chemical species); the large-scale tau-leaping variant proposed by Komarov and D'Souza [20], designed to accelerate very large-scale stochastic models (e.g., $> 10^5$ reactions). Two mixed approaches of fine- and coarse-grained acceleration are GPU-ODM [21], which is based on a variant of SSA called the Optimized Direct Method, and the SSA proposed by Sumiyoshi *et al.* [22]. Besides these GPU-powered tools, other solutions have been recently proposed, such as the Stochastic Simulation as a Service (StochSS) [23] and the Systems Biology Simulation Core Library (SBSCL) [24]. StochSS is an integrated development environment that can be used to model and simulate both deterministic

and discrete stochastic biochemical systems in up to three dimensions. SBSCL provides interpreters and solvers for different Systems Biology formats, like the Systems Biology Markup Language. SBSCL allows the user to simulate several models by supporting the resolution of ODEs and Stochastic Differential Equations.

Differently from the state-of-the-art approaches, we propose here FiCoS (**Fi**ne- and **Co**arse-grained **S**imulator), an efficient deterministic simulator that fully exploits the parallelism provided by modern GPUs, by using a combined fine- and coarse-grained parallelization strategy. In particular, the fine-grained parallelization is applied to distribute the calculations required to solve each ODE over multiple GPU cores, while the coarse-grained parallelization is exploited to perform parallel independent simulations. To the best of our knowledge, FiCoS is the first deterministic simulator that takes advantage of both parallelization strategies, filling the gap in the GPU-powered deterministic simulators (see Fig. 1).

Finally, FiCoS improves the approach proposed in LASSIE [19] (a GPU-powered simulator previously presented by the same authors), by leveraging more efficient integration methods, i.e., the Dormand–Prince (DOPRI) method [25–27] for stiffness-free systems, and the Radau IIA method [28,29] when the system of ODEs is stiff. As a matter of fact, the systems of ODEs, describing biochemical networks, are often affected by a well-known phenomenon named stiffness [30], which generally reduces the step-size of explicit integration algorithms up to extremely small values, thus increasing the running time. A system may be stiff when two well-separated dynamical modes, determined by fast and slow reactions, occur [31,32]: after a short transient the fast modes lead to stable trajectories, where only the slow modes drive the dynamics.

The performance of FiCoS has been assessed by means of a batch of tests involving up to 2048 parallel simulations of synthetic models of increasing size (up to hundreds of species and reactions), and a model describing the Autophagy/Translation switch based on the mutual inhibition of MTORC1 and ULK1 [33], characterized by 173 molecular species and 6581 reactions. The running time required by FiCoS to perform such analyses has been compared to the CPU-based ODE solvers LSODA and Variable-coefficient ODE (VODE) [34], as well as to the GPU-powered simulators cupSODA and LASSIE. Our results show that FiCoS represents the best solution in the case of computational tasks involving either a high number of simulations or large-scale models, overcoming the performance of all state-of-the-art competitors.

Results

In this section, the computational performance of FiCoS is compared against state-of-the-art ODE solvers, that is, LSODA [14], VODE [34], cupSODA [13], and LASSIE [19]. LSODA [14] is based on two families of multi-step methods, namely, the Adams methods [35], which are explicit integration algorithms inefficient for solving stiff systems, and the Backward Differentiation Formulae (BDF) [36], which are implicit integration methods suitable to solve stiff systems. LSODA can efficiently solve stiff systems by switching among the most appropriate integration algorithm during the simulation. Conversely, despite being based on the same integration algorithms used by LSODA, VODE exploits, at the beginning of the simulation, a heuristic to decide the best algorithm that should be used to integrate the system of ODEs. LASSIE solves the systems of ODEs by automatically switching between the Runge-Kutta-Fehlberg method in the absence of stiffness, and the first order BDF in presence of stiffness. Finally, FiCoS exploits a heuristic to determine if the system is stiff and which integration method is the most appropriate, between DOPRI and RADAU IIA methods (see Supplementary Text S1).

We performed two batches of tests using synthetic RBMs of increasing size (i.e.,

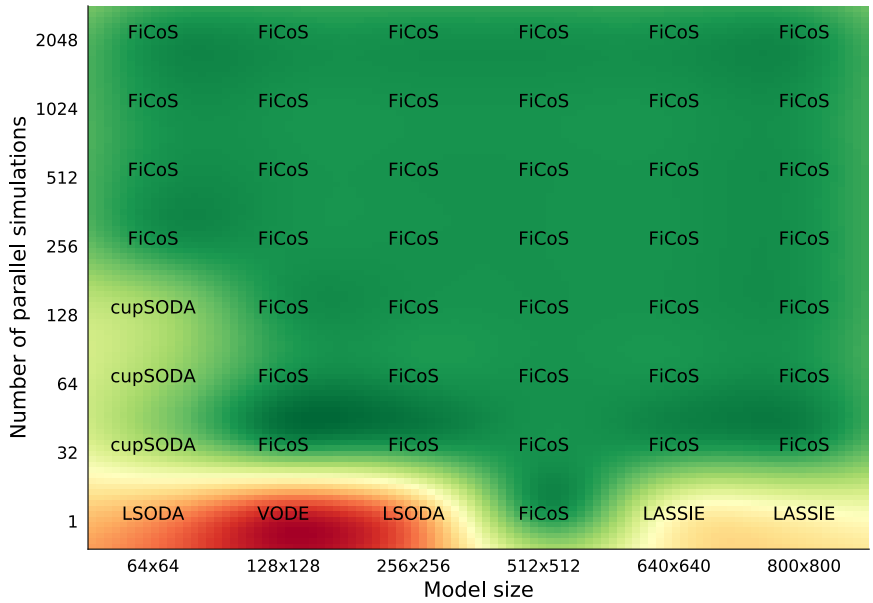

Fig 2. Comparison map representing the best simulator in terms of running time required to simulate symmetric RBMs of size $N \times M$, where N indicates the number of species and M the number of reactions composing the models. The running time was analyzed increasing both the size of the RBMs (horizontal axis) and the number of simulations (vertical axis).

number of species and number of reactions), generated as described in the Section "Materials and methods". Moreover, as a final case study, we propose an in-depth investigation of the RBM describing the Autophagy/Translation Switch Based on Mutual Inhibition of MTORC1 and ULK1 [33]. In all simulations, we save the dynamics of all species of the RBM.

All tests were executed on a workstation equipped with an Intel Core i7-2600 CPU (clock 3.4 GHz) and 8 GB of RAM, running on Ubuntu 16.04 LTS. The GPU used in the tests was an Nvidia GeForce GTX Titan X (3072 cores, clock 1.075 GHz, RAM 12 GB), CUDA toolkit version 8 (driver 387.26). All simulations were performed with the following settings: absolute error tolerance $\varepsilon_a = 10^{-12}$, relative error tolerance $\varepsilon_r = 10^{-6}$, and maximum number of allowed steps equal to 10^4 (note that these values are widely used in the literature and in the main computational tools like COPASI [37]). In all tests, we leveraged the LSODA and VODE implementations provided by the SciPy scientific library (v.0.18.1) [38] along with the NumPy library (v.1.11.2) under Python (v.2.7.12).

Performance analysis on synthetic models

The computational analyses carried out on synthetic models aimed at determining what is the best simulator to employ under specific conditions, i.e., considering computational tasks with increasing number of parallel simulations (up to 2048) and different RBM sizes (from tens to hundreds or even thousands of chemical species and reactions).

The map in Fig. 2 shows the results obtained in the case of "symmetric" RBMs, i.e., having the same number of chemical species and reactions $N = M$. of different size.

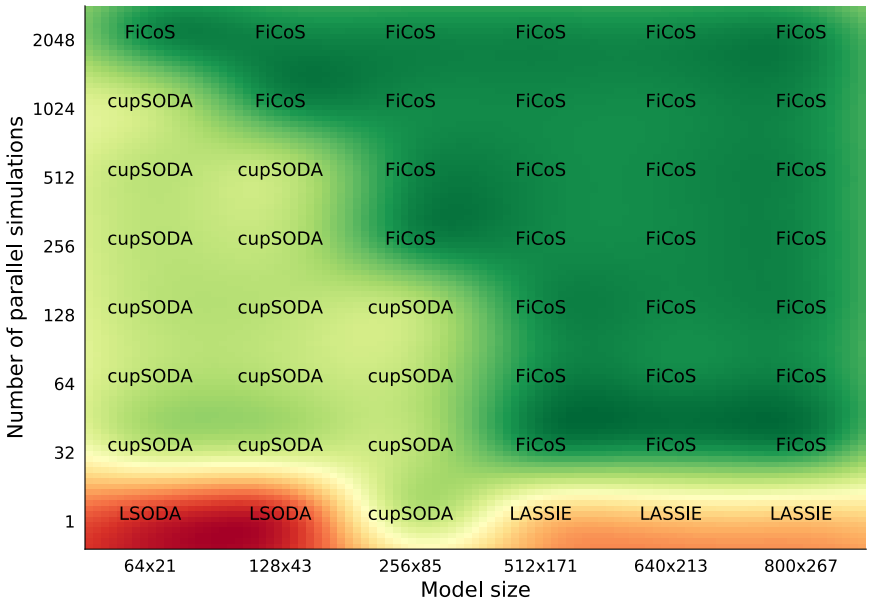

Fig 3. Comparison map representing the best simulator in terms of running time required to simulate asymmetric RBMs of size $N \times M$, where N indicates the number of species and M the number of reactions composing the models. These RBMs are characterized by a number of species higher than the number of reactions. The running time was analyzed increasing both the size of the RBMs (horizontal axis) and the number of simulations (vertical axis).

When a single simulation is performed, the CPU-based solvers (i.e., LSODA and VODE) achieved the best performance in the case of RBMs smaller than $N \times M = 512 \times 512$, which is the break-even point between CPU- and GPU-based simulators. Indeed, when larger models are considered, FiCoS and LASSIE are the best choices, thanks to their parallelization strategies that offload the calculations onto the GPU.

Considering the case of multiple independent simulations, our results indicate that cupSODA should be adopted with small-scale RBMs (i.e., less than 128 species and reactions) when the execution of a limited number of parallel simulations is required (i.e., less than 256). As a matter of fact, under these circumstances, cupSODA can leverage the most performing GPU memories, that is, the constant and the shared memory, which provide a relevant reduction of memory access latencies.

Finally, as shown by the green area of Fig. 2, FiCoS outperforms all its competitors in all the other cases thanks to its fine- and coarse-grained parallelization strategy, that allows for simultaneously distributing both the parallel simulations required in the task and the ODE calculations on the available GPU cores.

It is worth noting that the performance of the simulators can drastically change when dealing with RBMs characterized by a different number of chemical species and reactions (i.e., asymmetric RBMs, where $N \neq M$). As a matter of fact, since each ODE corresponds to a specific chemical species, the higher the number of the species, the higher the parallelization that can be achieved by exploiting the fine-grained strategy. On the contrary, the number of reactions is roughly related to the *length* (in terms of mathematical complexity) of each ODE, thus increasing the number of operations that

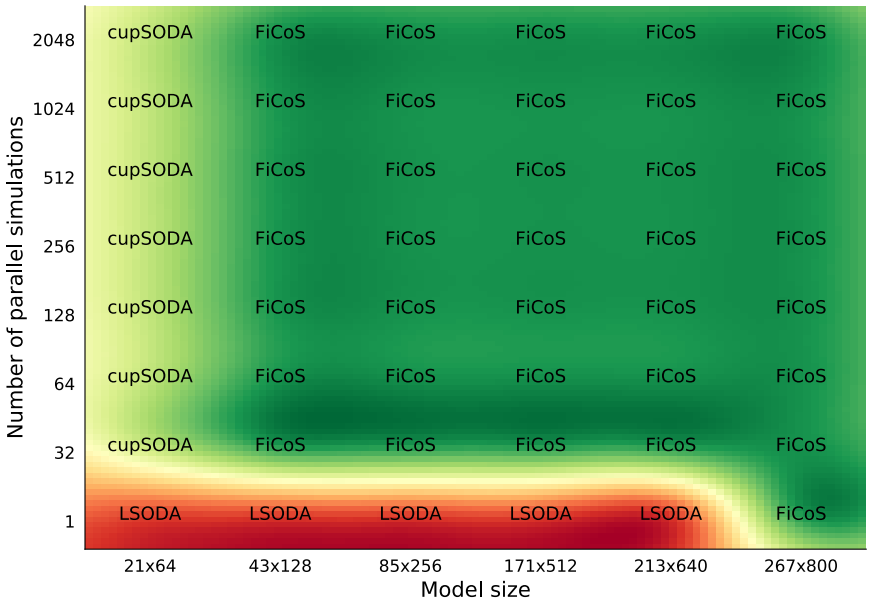

Fig 4. Comparison map representing the best simulator in terms of running time required to simulate asymmetric RBMs of size $N \times M$, where N indicates the number of species and M the number of reactions composing the models. These RBMs are characterized by a number of reactions higher than the number of species. The running time was analyzed increasing both the size of the RBMs (horizontal axis) and the number of simulations (vertical axis).

must be performed by each thread. In addition, since the clock frequency of the GPU cores is generally lower than that of the CPU, the time required to perform a single instruction on the GPU is higher. For this reason, the performance that can be achieved on the GPU decreases when the number of reactions M is larger than the number of species N.

To investigate this specific circumstance, we defined a set of "asymmetric" synthetic RBMs and tested the performance of the simulators. Figs. 3 and 4 show the results of the tests performed on asymmetric synthetic RBMs with $N > M$ and $M > N$, respectively. We first observe that the previous results are confirmed, as GPU-powered simulators most of the times overcome the CPU-based simulators. In particular, in the case of a single simulation of RBMs with $N > M$ (Fig. 3), LSODA is effective only for RBMs with a limited number of chemical species ($N < 128$), while cupSODA and LASSIE represent the best option for large-scale RBMs. Increasing the number of simulations, either cupSODA or FiCoS must be employed; the former for small-scale RBMs and the latter in all other cases.

Considering the case of RBMs with $M > N$ (Fig. 4), the best options for the single simulation are LSODA for RBMs with size up to 213×640, FiCoS otherwise. Again, increasing the number of simulations, cupSODA and FiCoS confirmed to be the best solutions for small-scale and large-scale RBMs, respectively. These results confirm that the number of reactions affects the performance that can be achieved with the GPU; as a matter of fact, cupSODA is the best solution only for RMBs with size 21×64, while FiCoS should be adopted in all other cases.

Overall, these analyses demonstrate the relevance of GPU-powered deterministic simulators, especially when high numbers of simulations must be performed or large-scale RBMs are taken into account.

Autophagy/Translation model

In order to show the advantages provided by FiCoS in the investigation of models of real biological systems, we performed a bi-dimensional Parameter Sweep Analysis (PSA-2D) on the mathematical model of the Autophagy/Translation switch based on the mutual inhibition of MTORC1 and ULK1 presented in [33]. These two proteins are part of a large regulatory network that is responsible of maintaining cellular energy and nutrients' homeostasis. As a matter of fact, autophagic processes are involved in cell survival during starvation and clearing of damaged cellular components, and thus their investigation is fundamental to understand aging-related and neurodegenerative diseases, as well as immunity and tumorigenesis.

The PSA-2D presented hereafter was already performed in [33] to study the oscillations in the phosphorylation levels of AMBRA1, a protein responsible of the activation of autophagy, and EIF4EBP1, a repressor of translation. Such oscillations generate alternating periods of autophagy and translation inside the cell, in response to varying levels of phosphorylated AMPK (AMPK*), whose presence represents a stressed condition in the cell, and the parameter P_9, which represents the strength of the negative regulation of MTORC1 by AMPK*.

The mathematical model of the Autophagy/Translation switch was defined using BioNetGen's rule-based modeling language (BNGL) [39]. Rule-based modeling is a practical means to describe, in a concise way, the interactions between molecules, keeping track of site-specific details (e.g., the phosphorylation states). We simulated the rule-based model by deriving the corresponding network of reactions, written as in conventional RBMs. In particular, the rule-based model is characterized by 7 initial molecule types, involved in 29 rules; the associated RBM contains 173 molecular species and 6581 reactions.

The PSA-2D was performed by varying simultaneously the two parameters that affect the oscillatory behavior, that is the initial amount of AMPK* and the value of the parameter of rule P_9, which modifies 5476 kinetic constants of the corresponding RBM. The initial values of both parameters were obtained by uniformly sampling within the sweep intervals proposed in the original paper [33], namely: $[0, 10^4]$ (molecules/cell) for the initial amount of AMPK*, and $[10^{-9}, 10^{-6}]$ ((molecules/cell)$^{-1}$s^{-1}) for the value of P_9. We performed a total of 36864 simulations with FiCoS, by running 72 batches of 512 simulations, a value that allowed us to maximize the performance of the simulator on the GPU, as evidenced by specific computational tests (data not shown). FiCoS completed all simulations in approximately 24 hours; in the same time-span, LSODA and VODE were capable of performing only 2090 and 1363 simulations, respectively.

Fig. 5 shows the average amplitude of $EIF4EBP$ (top plots) and $AMBRA$ (bottom plots) oscillations obtained with the PSA-2D using FiCoS (left), LSODA (center), and VODE (right). Amplitude values equal to zero (black points), denote a non-oscillating dynamics of the molecular species. As can be observed in the figure, although some information about the behavior of the system can be extrapolated by the results achieved with LSODA and VODE, FiCoS is the only simulation strategy that can provide biologists with extremely detailed analyses of complex cellular models in a limited time-span.

It is worth noting that we compared FiCoS only against LSODA and VODE for the following motivations: (i) LSODA and VODE are the most used simulators; (ii) the tests on the asymmetric synthetic RBMs with more reactions than species showed that the performance of LASSIE is much lower than FiCoS (see Supplemenray Text S1); and

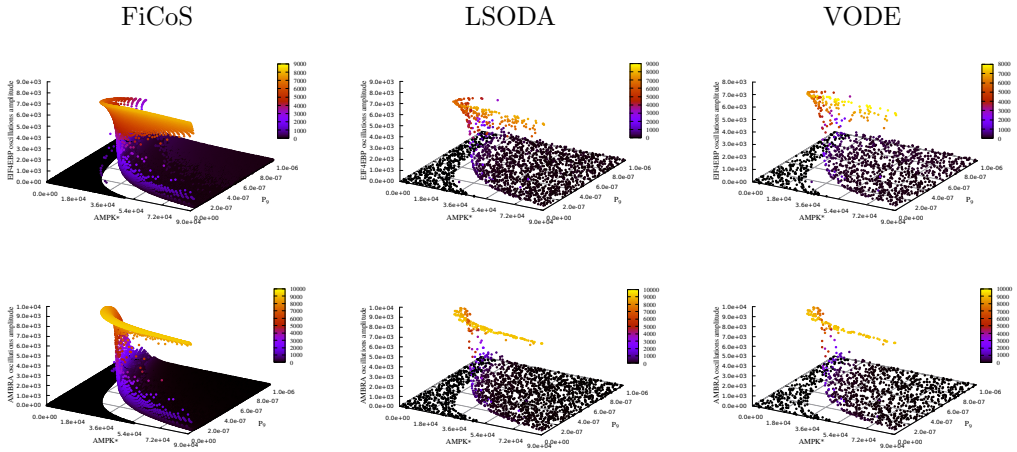

Fig 5. Results of a PSA-2D on the Autophagy/Translation model realized by varying the initial concentration of AMPK* and the value of the parameter P_9, which modifies 5476 kinetic parameters. The plots show the average amplitude oscillations of the species $EIF4EBP$ (top) and $AMBRA$ (bottom). Considering a time budget of 24 hours, FiCoS (left) was capable of performing 36864 simulations, while LSODA (center) and VODE (right) performed 2090 and 1363 simulations, respectively.

(*iii*) cupSODA is not suitable for the simulation of large-scale RBMs, as its memory requirements prevent the execution of this kind of tasks.

Discussion

The emerging field of General Purpose Computing on Graphics Processing Units (GPGPU computing)—in which CPUs and GPUs are combined to address the increasing demand of parallel and throughput-intensive algorithms—has had a relevant impact to many fields in both science and engineering [40]. Specifically, GPGPU computing enabled the development of novel software tools that have been successfully applied to tackle the huge computational efforts required by several disciplines, including Computational and Systems Biology [6]. The accelerated biochemical simulation method proposed here, named FiCoS, represents the first open-source deterministic simulator that takes advantage of both fine- and coarse-grained strategies to distribute the massive amount of calculations required by multiple simulations of large-scale RBMs. FiCoS can be effectively applied to tackle the demanding tasks that are generally required in Computational and Systems Biology, as in the case of the PSA presented in this work.

In terms of performance, FiCoS is capable of solving systems of non-stiff and stiff ODEs with similar and often higher precision and accuracy compared with other well-known ODE solvers (see Section "Simulation accuracy of FiCoS" in the Supplementary Text S1), but with a dramatic reduction of the execution time. In our tests, FiCoS resulted faster than CPU-based integration methods like LSODA and VODE when multiple simulations need to be executed. Moreover, in many cases FiCoS shown to be faster than LASSIE, which was designed to exploit the fine-grained strategy to distribute the calculations required by a single simulation of large-scale RBMs. As shown in Supplementary Text S1 (Section "Computational performance"), FiCoS achieves relevant speed-ups, resulting up to 487× faster than VODE in terms of simulation time and up to 855× considering only the integration time; up to 366× faster

than LSODA for the simulation time and up to 79× faster for the integration time only. Considering the comparison with the other GPU-based simulators, FiCoS is 298× faster than LASSIE in terms of simulation time and 760× faster for the integration time only, thanks to the double level of parallelism and the more efficient integration methods employed. FiCos was also capable of outperforming cupSODA, the GPU implementation of LSODA that exploits a coarse-grained parallelization strategy, resulting 7× faster in terms of simulation time and 17× considering only the integration time. To clarify, the integration time indicates the running time spent by a numerical integration algorithm to solve the system of ODEs, while the simulation time is the overall running time required to perform a simulation, including the I/O operations (i.e., reading and writing operations).

The analyses presented in this paper allowed us to also identify the limitations of FiCoS. First, since FiCoS was specifically designed to simulate large-scale RBMs, the intra-GPU communication overhead—caused by the fine-grain kernels—affects its performance in the case of small size models. Moreover, as expected, when only a few parallel simulations are required, other CPU- or GPU-based simulators might be preferred. Second, the performance of FiCoS related to the coarse-grained strategy can be affected by the saturation of the computing resources due to "excessive" parallelization. This phenomenon is caused by the computing resources required by the kernels realizing the fine-grained strategy—exploited to parallelize the resolution of the system of ODEs—which increase along with the RBM size. As a matter of fact, the blocks of threads generated using the Dynamic Parallelism (DP) [41,42] (see Supplementary Text S1 for details) easily saturate the GPU resources and reduce the speed-up. As shown in [43], the launching time of child kernels slightly increases after 512 kernels, and dramatically increases around 2000 kernels. Note that the parent grid is launched by the host, while a child grid (composed of child blocks, threads, and kernels) can be launched from each parent thread, resulting in the parallel execution of several child kernels. In addition, the register usage raises along with the number of threads running in parallel, decreasing the performance if the execution requires the access to the global memory, which has high latencies. According to our tests, the performance of FiCoS decreases when more than 2048 parallel simulations of large-scale RBMs are executed.

Conclusion

In this work we proposed FiCoS, a GPU-powered deterministic simulator, designed to deal with the computational demanding tasks generally required in the field of Computational and Systems Biology. FiCoS is capable of simulating large-scale RBMs based on mass-action kinetics relying on fine- and coarse-grained parallelization strategies.

FiCoS was designed to be a "black-box" simulator able to automatically convert RBMs, representing complex biochemical networks, into the corresponding systems of ODEs. These systems are solved by exploiting two Runge-Kutta methods, that is the DOPRI method [25-27] in the absence of stiffness and the Radau IIA method [28,29] when the system is stiff. This mixed parallelization strategy that takes advantage of the DP provided by modern GPUs allows for exploiting the massive parallel capabilities of GPUs, which are essential to obtain a relevant reduction of the computation time.

The performance of FiCoS was evaluated considering two sets of synthetic RBMs of increasing size (up to 800 molecular species and reactions) and a real model characterized by 173 molecular species and 6581 reactions. We performed an in-depth analysis to determine the best simulator that should be employed both considering the size of the RBM and the number of simulation required. In particular, we compared the

performance of FiCoS against two CPU-based simulators (i.e., LSODA and VODE), a fine-grained GPU-powered simulator designed to simulate large-scale models (i.e., LASSIE) and a coarse-grained GPU-powered simulator developed to run high numbers of simulations in a parallel fashion (i.e., cupSODA). Exploiting both the fine- and the coarse-grained parallelization strategies, FiCoS resulted the natural choice when multiple simulations are required, and when the number of species in the RMB is greater than 64, noticeably outperforming all the other simulators. We also performed a bi-dimensional PSA of a model of the Autophagy/Translation switch based on the mutual inhibition of MTORC1 and ULK1 [33], showing that FiCoS was able to execute 36864 simulations in 24 hours, while LSODA and VODE completed only 2090 and 1363 simulations in the same time-span, respectively.

Although FiCoS showed an excellent performance, we did not fully exploit the CUDA memory hierarchy that should be used as much as possible to obtain the best performance. Since both explicit and implicit integration algorithms are intrinsically sequential, FiCoS kernels generally do not reuse any variable, so that the (fast) shared memory of the GPU cannot be leveraged. Moreover, the DP does not currently allow for sharing variables among the threads belonging to the parent grid and those inside the child grids. For these reasons, the current version of FiCoS relies on the global memory (characterized by high latencies) and registers. We plan to develop an improved version of FiCoS tailored for RBMs characterized by a few number of species, leveraging both constant and shared memories, where the former can be used to store the kinetic constants and the structures used to correctly decode the ODEs, while the latter can be exploited to save the states of the system.

Materials and methods

Reaction-based model and ODE generation

RBMs are defined following a mechanistic, quantitative and parametric formalism created to describe and simulate networks of biochemical reactions [44]. An RBM can be defined by specifying the set $\mathcal{S} = \{S_1, \ldots, S_N\}$ of N molecular species and the set $\mathcal{R} = \{R_1, \ldots, R_M\}$ of M biochemical reactions that describes the interactions among the species in $\mathcal{S}$. A generic reaction R_i, with $i = 1, \ldots, M$, is defined as follows:

$$R_i : \sum_{j=1}^{N} a_{ij} S_j \xrightarrow{k_i} \sum_{j=1}^{N} b_{ij} S_j, \quad i = 1, \ldots, M, \tag{1}$$

where a_{ij}, $b_{ij} \in \mathbb{N}$ are the so-called stoichiometric coefficients and $k_i \in \mathbb{R}^+$ is the kinetic constant associated with R_i. Note that $\mathcal{R}$ can be also written in a compact matrix-vector form: $\mathbf{AS} \xrightarrow{\mathbf{K}} \mathbf{BS}$, where $\mathbf{S} = [S_1 \cdots S_N]^\top$ represents the N-dimensional column vector of molecular species, $\mathbf{K} = [k_1 \cdots k_M]^\top$ the M-dimensional column vector of kinetic constants, and $\mathbf{A}, \mathbf{B} \in \mathbf{N}^{M \times N}$ the so-called stoichiometric matrices whose (non-negative) elements $[A]_{i,j}$ and $[B]_{i,j}$ are the stoichiometric coefficients a_{ij} and b_{ij} of the reactants and the products of all reactions, respectively. Given an arbitrary RBM and assuming the law of mass-action [45,46], the system of coupled ODEs describing the variation in time of the species concentrations can be derived as follows:

$$\frac{d\mathbf{X}}{dt} = (\mathbf{B} - \mathbf{A})^T [\mathbf{K} \circ \mathbf{X}^{\mathbf{A}}], \tag{2}$$

where $\mathbf{X}$ is the N-dimensional vector of concentration values (representing the state of the system) at time t, the symbol $\circ$ denotes the entry-by-entry matrix multiplication (Hadamard product), and $\mathbf{X}^{\mathbf{A}}$ denotes the vector-matrix exponentiation form [45]. Note

that each species S_j at time t is characterized by its concentration X_j, where $X_j \in \mathbf{R}^{\geq 0}$ for $j = 1, \ldots, N$. Formally, $\mathbf{X}^{\mathbf{A}}$ is a M-dimensional vector whose i-th component is equal to $X_1^{Ai1} \cdots X_N^{AiN}$, for $i = 1, \ldots, M$. Note that, since we assume the law of mass-action, each ODE composing the system described in Eq. 2 is a polynomial function with at least one monomial associated with a specific kinetic constant.

Synthetic RBM generation

In order to investigate the computational performance of FiCoS, both the sets of randomly synthetic RBMs of increasing size were generated exploiting a custom tool developed by our group. The synthetic RBMs satisfy the following characteristics:

- a log-uniform distribution in the interval $[10^{-4}, 1)$ was applied to sample the initial concentrations of the molecular species;
- a log-uniform distribution in the interval $[10^{-6}, 10]$ was used to sample values of the kinetic constants;
- the stoichiometric matrix $\mathbf{A}$ was generated allowing only zero, first, and second-order reactions (i.e., in each reaction can appear at most two reactant molecules of the same or different species). The rationale is that the probability that a reaction simultaneously involving more than two reactants is almost equal to zero. Anyhow, FiCoS can simulate RBMs with reactions of higher-orders;
- the stoichiometric matrix $\mathbf{B}$ is created allowing at most 2 product molecules for each reaction.

It is therefore clear that the matrices $\mathbf{A}$ and $\mathbf{B}$ are sparse. A log-uniform distribution (i.e., a uniform distribution in the logarithmic space) was used to sample both the initial conditions and the kinetic constants to capture the typical dispersion of both the concentrations and kinetic parameters, which span over multiple orders of magnitude [2,47].

For each model, the initial value of the kinetic parameters was perturbed to generate different parameterizations (up to 2048). For each kinetic parameter k_i, with $i = 1, \ldots, M$, we applied the following perturbation:

$$k_i = \exp(\ln(k_i - 0.25 \cdot k_i) + (\ln(k_i + 0.25 \cdot k_i) - \ln(k_i - 0.25 \cdot k_i)) \cdot \mathrm{rnd}), \qquad (3)$$

where $\mathrm{rnd} \sim \mathrm{Uniform}(0, 1)$ is a random number sampled from the uniform distribution in $[0, 1)$.

Implementation of FiCoS

In order to efficiently deal with both stiff and non-stiff systems of ODEs, FiCoS relies on two advanced integration methods belonging to the Runge-Kutta families: (i) the DOPRI explicit method [25–27] for solving non-stiff systems; (ii) the Radau IIA implicit method [28,29] when the system under investigation is characterized by stiffness. In particular, we exploited DOPRI5 and RADAU5 methods, which are both Runge-Kutta methods of order 5. We also point out that our implementation was inspired by Blake Ashby's source code, who ported to `C++` the original `Fortran` code developed by Hairer and Wanner [27,28].

DOPRI is one of the most widespread explicit Runge-Kutta method, since it is a variable step-size algorithm and implements a stiffness detection approach [27]. Implicit Runge-Kutta methods are single-step approaches characterized by both a higher order of accuracy and more favorable numerical stability properties with respect to BDF

algorithms [48]. Among the existing implicit Runge-Kutta methods, the Radau IIA family was proved to hold several useful properties: it is strongly A-stable and strongly S-stable, implying that it is both A-stable and S-stable, as well as stiffly accurate [49]. One of the main advantages of the S-stable methods is their ability to produce stable computations when solving highly stiff problems. In addition, when a fixed step-size is used, strongly S-stable methods are more accurate than the other methods. Several numerical tests showed that only S-stable methods can produce accurate simulations by using significantly large step-size for stiff problems [50]. Finally, RADAU5 is capable of arranging the step-size along with the desired error—controlled by means of ε_a and ε_r tolerances, which are user-defined parameter—and the detected level of stiffness.

We exploited a coarse-grained strategy to perform a large number of simulations in a parallel fashion, by distributing the simulations over the available GPU cores. Moreover, in order to accelerate the ODE resolutions of each simulation, we exploited a fine-grained strategy using the DP to distribute the integration algorithms on the GPU cores. When DP is used, each thread belonging to the grid called by the host (named parent grid) can launch a novel grid (named child grid) composed of several threads. All child grids that are executed by the thread of the parent grid can be synchronized, so that the parent threads can consume the output produced by the child threads without involving the CPU. Specifically, we leveraged the DP to distribute the calculations related to the Butcher tableau (see Additional File, Section "GPU implementation", for additional details) characterizing our integration methods. As a side note, FiCoS also exploits the cuBLAS library [51] for both the matrix decompositions and the linear system resolutions required by RADAU5.

Overall, FiCoS work-flow consists in 5 different phases and 26 CUDA kernels, which can be summarized as follows:

- **Phase $\mathbf{P_1}$**: it implements the generation of the data structures encoding the system of ODEs. These data structures are used to evaluate the ODEs during the resolution of the system.

- **Phase $\mathbf{P_2}$**: for each simulation, the estimation of the dominant eigenvalue of the Jacobian matrix $\mathbf{J}$ is performed. Every simulation characterized by a dominant eigenvalue lower than 500 will be solved exploiting the DOPRI5 method. The other simulations will be tackled by means of the RADAU5 method. As a matter of fact, when the $\mathbf{J}$ has a large spectral radius, the system of ODEs can be considered stiff [30].

- **Phase $\mathbf{P_3}$**: it implements the DOPRI5 method. Note that if the DOPRI5 fails at solving some simulations, they are re-executed and solved relying on the RADAU5 method. It is worth noting that each thread, associated with a specific simulation, exploits the DP to call the kernels, developed to distribute the calculations required by DOPRI5, launching a novel grid of threads.

- **Phase $\mathbf{P_4}$**: it implements the RADAU5 method. As in the case of the DOPRI5 method, each thread takes advantage of the DP to run the kernels.

- **Phase $\mathbf{P_5}$**: for each simulation, the dynamics of the species are stored and written in the output files.

Notice that only phases P_1 and P_5 are executed on the CPU, while the others are entirely executed on the GPU.

Supporting information

Supplementary Text S1. It describes the design and implementation of FiCoS. In addition, it shows the achieved results in more details.

Supplementary Text S2. It explains the files required by FiCoS to perform the desired simulations.

References

1. Szallasi Z, Stelling J, Periwal V. System Modeling in Cellular Biology: From Concepts to Nuts and Bolts. The MIT Press; 2006.
2. Cazzaniga P, Nobile MS, Besozzi D. The impact of particles initialization in PSO: parameter estimation as a case in point. In: Proc. Conf. Computational Intelligence in Bioinformatics and Computational Biology. IEEE; 2015. p. 1–8.
3. Aldridge BB, Burke JM, Lauffenburger DA, Sorger PK. Physicochemical modelling of cell signalling pathways. Nat Cell Biol. 2006;8(11):1195–1203.
4. Chou IC, Voit EO. Recent developments in parameter estimation and structure identification of biochemical and genomic systems. Math Biosci. 2009;219(2):57–83.
5. Tangherloni A, Spolaor S, Cazzaniga P, Besozzi D, Rundo L, Mauri G, et al. Biochemical parameter estimation vs. benchmark functions: a comparative study of optimization performance and representation design. Appl Soft Comput. 2019;81:105494.
6. Nobile MS, Cazzaniga P, Tangherloni A, Besozzi D. Graphics processing units in bioinformatics, computational biology and systems biology. Brief Bioinform. 2017;18(5):870–885.
7. Wilkinson D. Stochastic modelling for quantitative description of heterogeneous biological systems. Nat Rev Genet. 2009;10(2):122–133.
8. Székely TJ, K B. Stochastic simulation in systems biology. Comput Struct Biotechnol J. 2014;12(20–21):14–25.
9. Harris LA, Clancy P. A “partitioned leaping” approach for multiscale modeling of chemical reaction dynamics. J Chem Phys. 2006;125(14):144107.
10. Gillespie DT. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J Comput Phys. 1976;22(4):403–434.
11. Li H, Petzold L. Efficient parallelization of the stochastic simulation algorithm for chemically reacting systems on the graphics processing unit. Int J High Perform Comput Appl. 2010;24(2):107–116.
12. Zhou Y, Liepe J, Sheng X, Stumpf MP, Barnes C. GPU accelerated biochemical network simulation. Bioinformatics. 2011;27(6):874–876.
13. Nobile MS, Cazzaniga P, Besozzi D, Mauri G. GPU-accelerated simulations of mass-action kinetics models with cupSODA. J Supercomput. 2014;69(1):17–24.
14. Petzold L. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. SIAM J Sci Stat Comp. 1983;4(1):136–148. doi:10.1137/0904010.

15. Nobile MS, Cazzaniga P, Besozzi D, Pescini D, Mauri G. cuTauLeaping: a GPU-powered tau-leaping stochastic simulator for massive parallel analyses of biological systems. PLOS ONE. 2014;9(3):e91963.

16. Cao Y, Gillespie DT, Petzold LR. Efficient step size selection for the tau-leaping simulation method. J Chem Phys. 2006;124(4):044109.

17. Harris LA, Nobile MS, Pino JC, Lubbock AL, Besozzi D, Mauri G, et al. GPU-powered model analysis with PySB/cupSODA. Bioinformatics. 2017;33(21):3492–3494.

18. Lopez CF, Muhlich JL, Bachman JA, Sorger PK. Programming biological models in Python using PySB. Mol Syst Biol. 2013;9(1).

19. Tangherloni A, Nobile MS, Besozzi D, Mauri G, Cazzaniga P. LASSIE: simulating large-scale models of biochemical systems on GPUs. BMC Bioinform. 2017;18(1):246.

20. Komarov I, D'Souza RM, Tapia J. Accelerating the Gillespie τ-leaping method using graphics processing units. PLOS ONE. 2012;7(6):e37370.

21. Komarov I, D'Souza RM. Accelerating the Gillespie exact stochastic simulation algorithm using hybrid parallel execution on graphics processing units. PLOS ONE. 2012;7(11):e46693.

22. Sumiyoshi K, Hirata K, Hiroi N, Funahashi A. Acceleration of discrete stochastic biochemical simulation using GPGPU. Front Physiol. 2015;6:42.

23. Drawert B, Hellander A, Bales B, Banerjee D, Bellesia G, Daigle Jr BJ, et al. Stochastic simulation service: bridging the gap between the computational expert and the biologist. PLOS Comput Biol. 2016;12(12):e1005220.

24. Panchiwala H, Shah S, Planatscher H, Zakharchuk M, König M, Dräger A. The Systems Biology Simulation Core Library. Preprints. 2020;.

25. Dormand JR, Prince PJ. A family of embedded Runge-Kutta formulae. J Comput Appl Math. 1980;6(1):19–26.

26. Dormand JR. Numerical methods for differential equations: a computational approach. vol. 3. CRC Press; 1996.

27. Hairer E, Nørsett SP, Wanner G. Solving ordinary differential equations I. vol. 8. Springer-Verlag, Berlin; 2008.

28. Hairer E, Wanner G. Solving ordinary differential equations II. vol. 14. Springer-Verlag, Berlin; 2002.

29. Hairer E, Wanner G. Stiff differential equations solved by Radau methods. J Comput Appl Math. 1999;111(1-2):93–111.

30. Higham DJ, Trefethen LN. Stiffness of ODEs. BIT Numer Math. 1993;33(2):285–303.

31. Gillespie DT. Stochastic simulation of chemical kinetics. Annu Rev Phys Chem. 2007;58:35–55.

32. Rathinam M, Petzold LR, Cao Y, Gillespie DT. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. J Chem Phys. 2003;119(24):12784–12794.

33. Szymańska P, Martin KR, MacKeigan JP, Hlavacek WS, Lipniacki T. Computational analysis of an Autophagy/Translation switch based on mutual inhibition of MTORC1 and ULK1. PLOS ONE. 2015;10(3):e0116550.

34. Brown PN, Byrne GD, Hindmarsh AC. VODE: A variable-coefficient ODE solver. SIAM J Sci Stat Comp. 1989;10(5):1038–1051. doi:10.1137/0910062.

35. Butcher JC. Numerical methods for ordinary differential equations. Chichester West Sussex, UK: John Wiley & Sons; 2008.

36. Cash JR. Backward Differentiation Formulae. In: Encyclopedia of Applied and Computational Mathematics, B. Engquist (ed.). Berlin Heidelberg: Springer; 2015. p. 97–101.

37. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, et al. COPASI - a COmplex PAthway SImulator. Bioinformatics. 2006;22(24):3067–3074.

38. Jones E, Oliphant T, Peterson P, et al.. SciPy: Open source scientific tools for Python; 2001. Available from: http://www.scipy.org/.

39. Harris LA, Hogg JS, Tapia JJ, Sekar JA, Gupta S, Korsunsky I, et al. BioNetGen 2.2: advances in rule-based modeling. Bioinformatics. 2016;32(21):3366–3368.

40. Przybylski A, Thiel B, Keller-Findeisen J, Stock B, Bates M. Gpufit: An open-source toolkit for GPU-accelerated curve fitting. Sci Rep. 2017;7(1):15722.

41. Nvidia. CUDA C Programming Guide, version 8.0; 2016.

42. Jones S. Introduction to dynamic parallelism. In: GPU Technology Conference Presentation S. vol. 338; 2012. p. 2012.

43. Wang J, Yalamanchili S. Characterization and analysis of dynamic parallelism in unstructured GPU applications. In: Proc. International Symposium on Workload Characterization. IEEE; 2014. p. 51–60.

44. Besozzi D. Reaction-Based Models of Biochemical Networks. In: Beckmann A, Bienvenu L, Jonoska N, editors. Pursuit of the Universal. 12th Conference on Computability in Europe, CiE 2016, Proceedings. vol. 9709 of LNCS. Switzerland: Springer International Publishing; 2016. p. 24–34.

45. Chellaboina V, Bhat SP, Haddad WM, Bernstein DS. Modeling and analysis of mass-action kinetics. IEEE Control Syst. 2009;29(4):60–78.

46. Voit EO, Martens HA, W OS. 150 years of the mass action law. PLOS Comput Biol. 2015;11(1):e1004012.

47. Yue H, Brown M, He F, Jia J, Kell DB. Sensitivity analysis and robust experimental design of a signal transduction pathway system. Int J Chem Kinet. 2008;40(11):730–741.

48. Martín-Vaquero J. A 17th-order Radau IIA method for package RADAU. Applications in mechanical systems. Comput Math Appl. 2010;59(8):2464–2472.

49. Prothero A, Robinson A. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. Math Comp. 1974;28(125):145–162.

50. Du Y, Liu Y. A strongly S-stable low-dissipation and low-dispersion Runge-Kutta scheme for convection diffusion systems. Aerosp Sci Technol. 2019;93:105355.

51. Nvidia. cuBLAS library 8.0; 2015.