

Accelerated Simulation of Hybrid Biological Models with Quasi-Disjoint Deterministic and Stochastic Subnets

Mostafa Herajy¹(✉) and Monika Heiner²

¹ Department of Mathematics and Computer Science, Faculty of Science,
Port Said University, Port Said 42521, Egypt

mherajy@sci.psu.edu.eg

² Computer Science Institute, Brandenburg University of Technology,
Postbox 10 13 44, 03013 Cottbus, Germany
<http://www-dssz.informatik.tu-cottbus.de>

Abstract. Computational biological models are indispensable tools for in silico hypothesis testing. But with the increasing complexity of biological systems, traditional simulators become inefficient to tackle emerging computational challenges. Hybrid simulation, which combines deterministic and stochastic parts, is a promising direction to deal with such challenges. However, currently existing algorithms of hybrid simulation are impractical for implementing real and complex biological systems. One reason for such limitation is that the performance of hybrid simulation not only relies on the number of stochastic events, but also on the type as well as the efficiency of the deterministic solver. In this paper, a new method is proposed for improving the performance of hybrid simulators by reducing the frequent reinitialisation of the deterministic solver. The proposed approach works well with models that contain a substantial number of stochastic events and higher numbers of continuous variables with limited connections between the deterministic and stochastic regimes. We tested these improvements on a number of case studies and it turns out that, for certain examples, the amended algorithm is ten times faster than the exact method.

Keywords: Accelerated hybrid simulation · Deterministic and stochastic simulation · Dependency graph · Computational modelling

1 Introduction

Hybrid simulation of biological reaction networks [11, 23, 28], which combines stochastic, deterministic, as well as other simulation algorithms, represents one important direction to deal with the recent challenges due to the increasing complexity of biological models [11, 16, 19]. It can successfully execute complex systems where molecular fluctuations drive the model results, but (discrete) stochastic simulation fails to produce any results in reasonable time. Moreover,

hybrid simulation provides an elegant tool to control the accuracy as well as the simulation efficiency by enabling the modeller to select an appropriate simulator to execute individual parts of a reaction network [14, 18, 19]. In addition, it can effectively deal with stiff models where reaction species and/or the reaction rate constants exhibit a considerable variation [14]. In this paper, we confine ourselves to a type of hybrid simulation that combines the deterministic and stochastic regime [11]. The deterministic part is executed by solving a system of ordinary differential equations (ODEs) and the stochastic part by considering all reactions as discrete and stochastic processes.

The hybrid simulation algorithm works by first partitioning a reaction network into two subnets: stochastic and continuous [11]. Afterwards, each subnet is executed using the corresponding simulator. Usually, the two regimes are not completely disconnected from each other. Instead, they share interfacing species/reactions that interconnect the execution of the two subnets. Thus, a synchronisation module is highly required to transfer the control between the two simulators. There are two main approaches to synchronise the operation of the internal simulators: approximate [11, 23] and exact [1, 28].

An approximate synchronisation starts by solving a system of ODEs until a discrete event occurs. The time at which the discrete event occurs is roughly located by repeatedly trying to take an appropriate time step. On the contrary, the exact synchronisation approach finds the correct time where the discrete event is scheduled to occur by incorporating an additional ODE to the deterministic regime representing a time transformation. In either of the two approaches, the stochastic module fires the occurring reaction and gives the control back to the deterministic solver.

Furthermore, approximate synchronisation does not provide sufficient accuracy as it does not capture all stochastic events. It aims to let the ODE solver to take a longer step size and subsequently improves the simulator performance. In contrast, the exact synchronisation approach can successfully locate all stochastic events in the discrete part. However, its performance may be prohibitively slow, which in many examples renders the hybrid simulation ironically slower than the stochastic one. There are many factors that contribute to such behaviour. One of the major reasons is the type of ODE solver.

A hybrid simulator requires an ODE solver that takes flexible steps such that the exact location of the stochastic event is precisely captured. Unfortunately, such type of ODE solver makes a significant effort to initially select the appropriate step size that satisfies certain accuracy criteria. However, hybrid simulation necessitates the reinitialisation of the ODE solver each time the simulator switches from the discrete to the continuous regime. This step is imperative such that the deterministic solver can account for the discontinuities due to the firing of the stochastic reactions. Consequently, as the number of stochastic events increases, the performance of the hybrid simulation rapidly decreases.

In this paper, we propose an approach for improving the performance of hybrid simulation by significantly reducing the number of times the ODE solver is reinitialised. We utilise two ideas to achieve this goal: the dependency graph and the approximation of the ODE due to the cumulative propensity.

The dependency graph [5] is widely deployed to improve the performance of stochastic simulation by recording for each reaction the set of related reactions that may be affected when this reaction occurs. This idea can also help in the case of hybrid simulation by deciding, each time a stochastic reaction occurs, whether the ODE solver should be reinitialised. If there is any dependency between the just occurred reaction and any of the continuous reactions, the ODE solver should be reinitialised.

Furthermore, we isolate the ODE due to the cumulative propensities so that the deterministic solver becomes independent from the stochastic part for longer time periods. We replace the simultaneous numerical integration by an approximation of this ODE through one Euler step. This approach works well when there are only a few interface nodes between the stochastic and the deterministic regime, but a significant number of stochastic events. Fortunately, these two conditions are often satisfied for many real and complex biological models.

This paper is organised as follows: we start with an overview of biological network simulation by summarising the three main simulation approaches. Next, the proposed method of improving the hybrid simulation is presented by introducing two algorithms: one algorithm for finding the interface reactions, and the other one makes use of this information to improve the hybrid simulation. To illustrate the performance as well as the accuracy of this method, we show the result of three numerical experiments. Finally, we conclude with a few remarks and an outlook on future work.

2 Simulation of Biochemical Reaction Networks

We consider the problem of simulating N biochemical species S_1, S_2, \dots, S_N interacting through M reaction channels r_1, r_2, \dots, r_M in a well-mixed system of molecules [8]. Each reaction is associated with a rate constant k_i which serves as parameter for the state-dependent reaction rate. The state-dependent reaction rates usually follow specific kinetic rate laws, e.g., mass action, as all examples used in this paper.

There are many different simulation methods; see [8, 26] for surveys. However, all of those procedures can be classified into three main approaches: deterministic, stochastic, or hybrid.

The traditional approach for studying the evolution of individual species with respect to time is to consider the chemical species evolving deterministically and continuously over time. According to this view, an ordinary differential equation (ODE) is constructed for each species S_i , and each reaction, in which S_i takes part, contributes to the species' ODE. The generation as well as the numerical solution of the system of ODEs are straightforward, but the simulation results do not always offer an accurate approximation of the dynamics of the corresponding biochemical system [8, 25, 26, 30]. For instance, when one or more of the species participating in the biochemical reaction system exhibit a few number of molecules, the ODE approach fails to account for intrinsic noise due to the molecular fluctuation of such species [8, 30]. Unfortunately, such fluctuations are often crucial when simulating biological systems; see, e.g., [2, 22, 30].

As an alternative approach, a stochastic simulation can be used by solving the Chemical Master Equation (CME), which calculates the probability that each species will have a certain number of molecules at a future time [8], or following the idea of Gillespie [6] by generating a set of trajectories by means of the stochastic simulation algorithm (SSA). In fact, the SSA algorithm generates a numerical realisations of the CME [8]. There are different variations of Gillespie's basic idea as well as many improvements [5, 26].

Nevertheless, the stochastic simulation is prohibitively slow to simulate real biological models that include many reaction events. As a discrete simulation, the stochastic simulation executes each individual event. For instance, in [21] the authors were not able to experiment with stochastic simulation due to the involvement of species with a big abundance of molecules. The performance of the stochastic simulation generally depends on the number of molecules of the participating species. For instance, for certain models, if the initial values of certain species are doubled, the time required to simulate the model rapidly increases [11].

With the recent increasing interest in using stochastic simulation to study the behaviour of biological models, new methods have been developed to improve the performance of the stochastic simulation by approximating the dynamics of biological systems. For example, in [4, 27] approximation techniques of the exact stochastic simulation have been developed by combining a number of events and execute them together instead of just simulating one reaction at a time. This improves the performance of the stochastic simulation by taking longer time steps. However, one difficulty of such a method is the selection of a suitable time step that makes a good trade-off between the simulation accuracy and efficiency.

Another direction to overcome the limitation of stochastic simulation is to resort to hybrid simulation [11, 23]. In this approach, the set of reactions are partitioned (either statically or dynamically) into two groups: slow and fast. The slow group contains reactions that have rate constants with small values and their corresponding substrate species do not exhibit high number of molecules, which yields in total low reaction rates, while the fast group contains the set of reactions with large rate constants and/or substrate species with abundant number of molecules. Afterwards, the slow group is simulated stochastically, while the fast group is simulated continuously.

The reactions in the two groups need to be synchronised due to interfacing species and/or reactions. With other words, the hybrid simulation algorithm has to decide when to jump from one simulation regime to the other. This step is crucial for the performance of hybrid simulation, and different synchronisation mechanisms have been proposed [11, 23, 26]. This synchronisation procedure is occasionally referred to as time transformation, since it switches the simulation time calculation from the continuous to the stochastic one.

Gillespie derives in [7] a general method for time-dependent propensities. This equation has been used afterwards in [1, 9, 11] to decide when to jump from the deterministic simulator to the stochastic one. It calculates the time when a stochastic event is to occur, while the ODE solver is integrating the system of

ODEs. According to the time-dependent propensity, a stochastic event occurs when the following equation is satisfied [11]:

$$\int_t^{t+\tau} a_0^s(\mathbf{x})dt + \log(p_1) = 0, \quad (1)$$

where \mathbf{x} is the state vector of the model at time t , p_1 is a random number uniformly distributed in $[0, 1]$, and a_0^s is the cumulative (total) propensity of the reactions in the slow group.

Equation (1) means that we carry out the numerical integration of the cumulative propensities that correspond to the slow reactions from the occurrence time of the previous reaction until the sum of the integral value and the log of a random number p_1 is equal to zero.

After that, the next reaction to occur is selected as the first index μ satisfying

$$\sum_{j=1}^{\mu} a_j^s(\mathbf{x}) > p_2 a_0^s(\mathbf{x}), \quad (2)$$

where p_2 is a random number generated from the uniform distribution, and a_j^s is the propensity of the j^{th} slow reaction.

In [1] a set of algorithms have been derived to move from the deterministic regime to the stochastic one, when (1) is satisfied. The authors suggest to use an ODE solver with event handling to accurately detect the exact time point when (1) is satisfied. However, in [11] it has been previously asserted that exactly satisfying (1) will rapidly slow down the simulation. Therefore, in [11] they use an approximation of Eq. (1) to generate the trajectories by introducing a probability of no reaction (a dummy reaction).

However, this approximation approach will miss important events which has a negative impact on the accuracy of the hybrid simulator. Using an exact detection of (1), where the time-varying propensity is fully captured, will result in a hybrid simulator which is even slower than the pure stochastic one. This happens when many stochastic events occur in the slow part. This drawback has been practically encountered and reported by many researchers (see e.g., [24]).

One reason for this limitation is the frequent switch to the ODE solver. When there are many events in the stochastic regime, the cumulative propensity of slow reactions becomes larger and time steps between two successive stochastic events become smaller. Thus the time steps taken by the ODE solver will become smaller, too. So, the time saved by approximating some reactions using the deterministic simulation will be lost. Moreover, each time a stochastic event is fired, there is a discontinuity in the system of ODEs. Thus, the ODE solver should be reinitialised.

Frequent reinitialisation of the ODE solver decreases the performance of the whole hybrid simulation. Fortunately, not all firing events directly affect the system state of the ODE solver. Only reactions related to the interfacing species have an effect which needs to be taken into account. When a stochastic event occurs, which is not related to any interface reaction, the deterministic solver can

simply continue working, after the firing of the stochastic event. On the contrary, when the occurring event is related to an interface reaction, the deterministic solver should restart the integration from this time point on.

In the subsequent section, we propose an algorithm to identify those interfacing reactions and then we show how the hybrid simulation can take advantage of this information to accelerate the overall simulation.

3 Accelerated Hybrid Simulation

In this section, we discuss the proposed procedure to improve the performance of the hybrid simulation. We start with describing the process of identifying interface reactions in the slow group that have a direct connection to the set of reactions in the fast group. Next, we outline the steps of the hybrid simulation that makes use of the information collected by the interface reaction detection procedure.

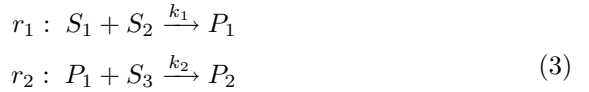
3.1 Detecting Interface Reactions

Suppose we have two groups of reactions: slow and fast. The occurrence of a reaction in the slow group can directly or indirectly influence the state vector of the variables in the fast group that are numerically integrated by an ODE solver.

A *direct dependency* exists between the two groups when the occurrence of a stochastic event changes one or more values of the state vector related to the fast reactions. This happens when one or more variables are shared between the two groups.

On the contrary, an *indirect dependency* exists when the firing of a stochastic event does not change any variable values related to the fast reactions, but it alters certain variables in the slow group which have an impact on the rates of the variables in the fast group during the numerical integration. That is, discrete variables appear in this case as multiplication factor with the rate constant (e.g., as in reactions related to positive and negative feedback loops).

For an example, consider the following two reaction sets:



For the two reactions in (3), r_1 and r_2 share a common species (P_1). Therefore, when r_1 takes place, it affects the system state significant for the reaction r_2 . But, for the reaction set in (4), when r_3 occurs, it does not directly affect the state of the species participating in r_4 , but it influences the kinetic rate of this reaction.

Now consider the case, where r_1 is simulated stochastically, while r_2 is simulated deterministically by solving an ODE. Each firing of r_1 results in a discontinuity in the ODE involving r_2 . But, when r_3 fires, it does not always cause discontinuities; such changes can be easily taken into account during the evaluation of the ODE function. Assuming that r_1 and r_2 are part of a bigger reaction network, then we call r_1 an *interface reaction*. When we list all interface reactions, we consider only those ones that have a direct dependency with the fast group.

Please note, we assume that the ODE solver is able to deal with small changes due the effect of indirect dependency of reactions. If this is not the case, then such reactions should also be added to the set of interface reactions. For instance, if the reaction r_3 increases the number of molecules of P_1 from 0 to 1, an abrupt change occurs which causes a discontinuity in the reaction set (4). As this behaviour does not occur during the whole simulation time, the specific time step where r_3 is anticipated to cause a discontinuity can be located during the simulation and therefore the ODE solver is reinitialised.

To find the set of all interface reactions whose occurrences affect the state vector of the ODE solver, we use an idea similar to the dependency graph [5]. The dependency graph is popular in accelerating stochastic simulations by storing for each reaction the set of other reactions that will be affected by every occurrence, so that the propensities do not have to be updated for all reactions each time a given reaction takes place. Please note that the dependency graph is created only once, but is extensively used during the simulation. Only the propensities of these dependent reactions are updated. Therefore, a substantial amount of processing time is saved in general. We use this information to detect the dependency between each stochastic reaction in the slow group and other related reactions in the fast group. Unlike the dependency graph used in the stochastic simulation, we record only those reactions that have a direct dependency with the fast group. Algorithm 1 summarises the steps for detecting all interface reactions that have a direct influence on the fast group.

The algorithm takes as input the two reaction groups: G_s and G_f , which denote the sets of slow and fast reactions, respectively. The set of interface reactions R^* is initially empty (step 1). Next, in steps 2–10, the procedure iterates over all reactions in the slow group. For each reaction r_i we identify the set of species that are altered when this reaction takes place (step 3). We call such variables *manipulated species*. This information can easily be found by help of the state change vector corresponding to such a reaction. After that the algorithm iterates for each manipulated species to find other reactions in the fast group that also manipulate the same species (that is it adds or subtracts molecules from that species, when the reaction occurs) (step 5). If such a reaction is found, we mark r_i as an interface reaction (steps 6–7). On termination the algorithm returns the set of marked interface reactions (step 11).

At step 5, the algorithm finds first the set of reactions that manipulate the species in the slow group, then it searches for the existence of these species in the fast group. While this step seems to be redundant, the algorithm assumes that this information already exists through the dependency of all reactions

Algorithm 1. Finding Interface Reactions

Require: G_s the set of slow reactions;
Require: G_f the set of fast reactions;
1: $R^* = \phi$ {the set of marked interface reactions is initially empty}
2: **for each** r_i in G_s **do**
3: let S_i denotes the set of manipulated species when r_i fires;
4: **for each** s_{ij} in S_i **do**
5: Find the set of other reactions, R_{ij} , that manipulate s_{ij} when they fire;
6: **if** $\exists r_j \in R_{ij}$ **and** $r_j \in G_f$ **then**
7: Add r_i to R^* ; {Mark r_i as an interface reaction}
8: **end if**
9: **end for**
10: **end for**
11: **return** R^* ;

with each others, as it has been discussed in [5]. This dependency information is also required to increase the performance of the stochastic simulation of the slow subnet when updating the propensities of a fired reaction along with their dependent ones. If this information is not available, then we can alternatively search the reactions that manipulate s_{ij} directly from the fast reaction set.

As an example for the steps performed by Algorithm 1, consider the reactions in Table 1, assuming that these reactions are partitioned into two groups: the first group contains the slow reactions $G_s = \{r_1, \dots, r_5\}$, while the second group contains the set of fast reactions $G_f = \{r_6, \dots, r_8\}$. The set of manipulated species for the reactions r_1, r_2, r_3, r_4 are $\{A\}, \{A\}, \{B\}, \{B\}$, respectively. None of them is manipulated by any of the reactions in the fast group. Similarly, the set of manipulated species of the reaction r_5 is $\{A, B, C\}$. A and B are not manipulated by any reaction in the fast group. However, C is manipulated by the two reactions r_6 and r_7 . Therefore, the reaction r_5 is identified as an interface reaction.

Obviously, as the number of interface reactions increases, the number of times the ODE solver is initialised also increases. This means that it is always advantageous to minimise the elements in the set of interface reactions while performing the partitioning. In what follows, we show how the information collected by this algorithm helps to accelerate the hybrid simulation.

3.2 Improving the Performance of the Hybrid Simulation Algorithm

Algorithm 2 lists the proposed steps to speed up the hybrid simulation algorithms presented in [1, 11] by introducing two additional improvements: exploiting the dependency information collected by Algorithm 1, and replacing the simultaneous integration of the system of ODEs and Eq. (1) by a fixed integration step.

Reinitialising the ODE solver each time an event occurs at the stochastic regime predominately affects the performance of the hybrid simulator. As soon as the number of stochastic events increases, the repetitive reinitialisations of the ODE solver consumes most of the time of the hybrid simulator to switch from

Table 1. An example of a set of slow and fast reactions

#	Slow reactions	#	Fast reactions
r_1	$\phi \xrightarrow{s} A$	r_6	$C + E \xrightarrow{k_2} D$
r_2	$A \xrightarrow{d} \phi$	r_7	$D \xrightarrow{k_3} C + E$
r_3	$\phi \xrightarrow{s} B$	r_8	$D \xrightarrow{dd} \phi$
r_4	$B \xrightarrow{d} \phi$		
r_5	$A + B \xrightarrow{k_1} B + C$		

the stochastic to the continuous regime. In fact, this step hampers the practical implementation of hybrid simulation due to the computational expense required to reinitialise the ODE solver.

To clarify this point, consider the task of selecting an appropriate ODE solver to numerically integrate a system of ODEs. One can choose between simple fixed step size solvers (e.g., Euler) [10] or a more complicated numerical adaptive integration algorithm (e.g., backward differentiation formula [20]). Fixed step size solvers do not provide a good accuracy unless a very small step is chosen. However, using small steps rapidly decreases the performance of the simulation algorithm. Besides, even when using a small step size, the result is not satisfactory for many real applications, and such type of solvers do not scale well with the currently observed rapid increase of the systems biology's model sizes.

As an alternative to fixed step size solvers, adaptive ODE solvers can be used. To increase the performance of the numerical integrator, adaptive ODE solvers take small steps when the solution is not smooth and longer ones when the solution is smooth. Initially, such type of solvers employ certain algorithms to select a step to start with. Moreover, many libraries implement more or less complicated procedures to decide which step size to take by conducting many accuracy and convergence checks.

For instance, in a typical ODE library (e.g., [20]), each time the solver takes a step, it uses its own history information to advance the solution. Such type of solvers are highly required to improve the performance of hybrid simulation. However, in hybrid simulation, each time a stochastic event occurs, we have to clear all of this information by reinitialising the ODE solver, so that it can deal with the discontinuity due to the firing of a reaction in the slow regime. Moreover, the ODE solver will have to start from scratch each time this information is cleared. Indeed many libraries offer a lightweight reinitialisation to decrease the effect of this problem. However, also this version of reinitialisation takes a considerable time.

When we reset the ODE solver just a few times as in simple discrete-continuous systems, this issue does not present an intricate problem, since only a few discrete events will take place. However, for hybrid simulation it is compulsory to repeatedly clear the solution history each time an event occurs. Unfortunately, even for simple hybrid models there are thousands of such events (cf. Table 2).

Therefore, reducing the number of times the ODE solver is reinitialised will inevitably improve the performance of hybrid simulation.

Nevertheless, the hybrid simulation approaches in [1, 11] require the simultaneous integration of the cumulative propensity (Eq. (1)) with the rest of the system of ODEs. This condition mandates the reinitialisation of the ODE solver each time an event occurs so that we restart the integration of (1) each time the state vector of the slow group is changed. It has been asserted in [11] that this step is computationally very expensive.

To relax this condition, we approximate the simultaneous integration of (1) by calculating its value using

$$a_0^s(\mathbf{x}) \cdot \Delta\tau + \log(p_1) = 0, \quad (5)$$

where $\Delta\tau$ is the time difference between the occurrence time of the previous event and the current event, and p_1 is a random number generated from the uniform distribution.

Therefore, each time the ODE solver checks for the fulfillment of (1), it computes the value of (5) instead. The latter equation does not require the reinitialisation of the ODE solver, because there is no additional variable added to the system of ODEs that represents the cumulative propensity. When there are a substantial number of stochastic events, $\Delta\tau$ becomes very small which results in a good approximation of (1).

Algorithm 2 summarises the steps involved in improving the performance of the hybrid simulation algorithm in [11]. This algorithm takes advantage of the dependency graph in two locations. First, when it returns from the continuous integration, it updates all the propensities of stochastic reactions that share a species with the fast group. Second, it utilises the dependency information to decide whether to clear the ODE solver history information. We define two functions for this purpose: **Base**(r_i) and **Manipulated**(r_i). **Base**(r_i) returns all the species that are used to define the propensity of the reaction r_i , while **Manipulated**(r_i) returns all species that are affected by the occurrence of the reaction r_i . This information can easily be calculated from the dependency graph.

The algorithm takes as input the two sets of reactions G_s and G_f as well as the set of marked interface reactions (R^*). At step 1, we initialise the ODE solver with the current concentration of species in the fast group. Initially, at step 2, we set the current time (τ) as well as the previous event time (τ_{old}) to zero. Afterwards the algorithm iterates until the end simulation time is reached (steps 3–16). At each iteration two random numbers (p_1 and p_2) are generated (step 3). p_1 is used to calculate (5), while p_2 is used to find the next event in (2). In the steps 5–7, we repeatedly integrate the system of ODE until (5) is satisfied. Equation (5) is recalculated each time the ODE solver checks for a root. When (5) is satisfied (step 7), the algorithm exits from the numerical integration to find the next stochastic event to occur using (2) (step 9). However, before finding the next stochastic reaction to fire, at step 8, the propensities of the reactions in the slow group are updated first using the information collected by the dependency graph. The function **Update**($a(r_i), a_0^s$) updates the propensity

Algorithm 2. Accelerated Hybrid Simulation

Require: G_s and G_f : the sets of slow and fast reactions respectively;
Require: R^* the set of reactions marked as interface reactions;
1: Initialise the ODE solver with the initial concentration of the variables in G_f ;
2: set $\tau = \tau_{old} = 0$;
3: **while** we did not reach end simulation time **do**
4: Generate two random numbers p_1 and p_2 from the uniform distribution;
5: **repeat**
6: Numerically integrate the system of ODEs;
7: **until** $a_0^s(\mathbf{x}) \cdot (\tau - \tau_{old}) + \log(p_1) = 0$ {cf., Eq., (5)}
8: **Update**($a(r_i), a_0^s$), $\forall r_i \in G_s, \forall r_j \in G_f : \mathbf{Base}(r_i) \cap \mathbf{Manipulated}(r_j) \neq \emptyset$;
9: Find the reaction r_μ that satisfies (2) using p_2 ;
10: Fire r_μ and update the system state as well as the current time τ ;
11: **Update**($a(r_i), a_0^s$), $\forall r_i : \mathbf{Base}(r_i) \cap \mathbf{Manipulated}(r_\mu) \neq \emptyset$;
12: Set $\tau_{old} = \tau$
13: **if** $r_\mu \in R^*$ **then**
14: Reinitialise the ODE solver
15: **end if**
16: **end while**

of the reaction r_i ($a(r_i)$) as well as the cumulative one (a_0^s). Afterwards, the selected reaction is fired and the system state as well as the reaction propensities are updated (steps 9–11). The current event time is recorded as the previous event occurrence time (step 12). Thereafter the algorithm checks if the occurred reaction is in the marked list. If so, the ODE solver is reinitialised (steps (13–15)).

There are two updates of the slow reaction propensities. The first update takes place when we switch from the deterministic to the stochastic regime, while the second update is required when a reaction in the slow set has fired. In the former, all slow reactions that have a dependency with the fast reaction are updated, while in the latter, only reactions that have a dependency with the fired reaction are updated.

Algorithm 2 follows the direct method to implement hybrid simulation. However, it can be extended to include the first and next reaction methods as it has been proposed in [1, 11, 28]. But in our opinion, the direct method is the best choice in terms of implementation and performance unless a parallelisation approach is used. For instance, the first reaction method requires the generation of random numbers equal to the number of reactions in the reaction network which turns out to be a performance bottleneck. This drawback can be tackled by the next reaction method, but the latter complicates the implementation by introducing a specialised data structure: the priority queue.

Furthermore, this procedure assumes that the set of reactions are partitioned offline into slow and fast ones. However, this assumption does not prevent the use of an adaptive scheme as the one presented in [9] to run this algorithm. Nevertheless, the set of interface reactions will need to be updated each time a repartitioning is performed. To minimise the computational overhead due to the re-computation of the set of interface reactions, the already found interface reactions can be updated

(e.g., by adding/removing reactions) instead of searching for them from scratch. Moreover, minimising the number of interface reactions can be introduced as an additional criteria for the dynamic partitioning procedure.

The check in step 13 can decrease the performance of the simulation algorithm if it will be implemented by searching the list of interface reactions each time a slow reaction is fired. However, a frequent search for the interface reactions can be avoided by storing a flag for each reaction that determines whether this reaction is marked as an interface one.

In the next section we show some experimental results of this algorithm.

4 Numerical Experiments

In this section, we present three examples to assess the performance of our accelerated hybrid simulation approach: a model for the circadian oscillation and two versions of a cell cycle model. In all examples, molecular fluctuations play a crucial role such that stochastic simulation is mandatory to reproduce the corresponding actual biological behaviours. We compare the performance of the exact algorithm with the accelerated one. For this purpose, we use the number of stochastic events in the slow group as a measure of the closeness of the simulation results.

These case studies have been tested using an implementation of Algorithms 1 and 2 in Snoopy [12] – a Petri net editing and simulation tool, and its related Steering Server for Collaborative Simulation, S^4 [15]. This implementation allows the simulation of (coloured) hybrid Petri nets [14, 16]. An external ODE solver from the Sundials CVODE library [20] is used to simulate the deterministic part, while the direct method [6] is used to simulate the stochastic part. The runtime as well as the number of stochastic events are determined by executing the hybrid simulation 10 times and taking the average.

We provide all details for the first case study, which immediately permit to reproduce our results reported, but refer to the literature for the other two case studies because of the large number of reactions and species involved.

4.1 Circadian Oscillation

We use a rather small example with a few reactions and species to initially test the performance of the accelerated hybrid simulation. We adopt the circadian oscillation model introduced in [32]. The set of reactions as well as the kinetic rate constants are given in Appendix A.

The model comprises two genes: *gene1* and *gene2*. The two genes are transcribed into their corresponding mRNAs, which in turn are translated into the proteins *A* (activator) and *R* (repressor), respectively. The reactions for the two genes/mRNAs/proteins are similar to each other, unless for those reactions that model the interaction between the two proteins.

This simple circuit together with the corresponding kinetic information fails to produce any oscillation when deterministic simulation is used [32]. Therefore,

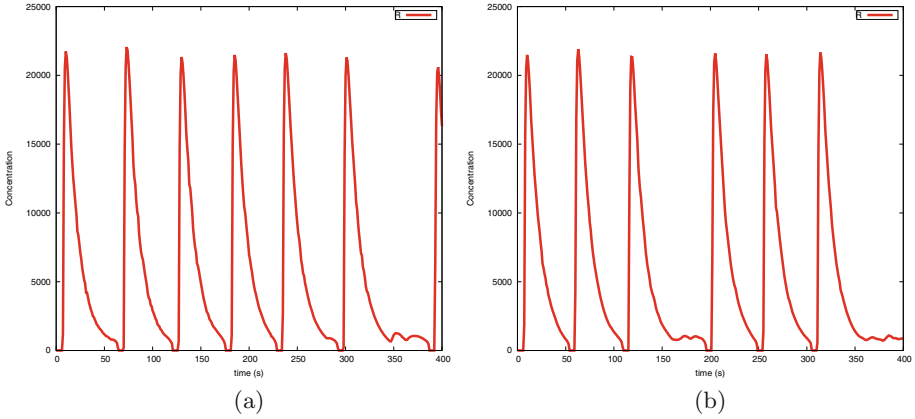


Fig. 1. Simulation results of the circadian oscillation model in [32] (single run) for: (a) exact, and (b) accelerated hybrid simulation

stochastic and hybrid simulation play a crucial role to simulate this model. We partition this reaction network such that reactions related to the mRNA of gene2 (R_{11} , R_{12} , R_{13}) are simulated stochastically, while the remaining reactions are executed via the deterministic solver. Such type of partitioning is sufficient to force the model to produce oscillations. Fortunately, the set of marked reactions (R^*) is empty since the three slow reactions do not influence the state vector of the fast reactions.

Figure 1 depicts the simulation result of this model when it has been executed using both the exact as well as the accelerated version of the hybrid simulation algorithm. Moreover, Table 2 lists the runtime behaviours of the two algorithms.

According to the data in Table 2, there is an improvement in terms of runtime when the accelerated algorithm is used. The accelerated simulation algorithm is about three times faster than the exact method. Moreover, the number of generated stochastic events for both simulators are comparable. Therefore, this saving in runtime behaviour is due to the avoidance of repeated reinitialisation of the ODE solver and not because of reducing the number of stochastic events.

4.2 Yeast Cell Cycle

As another test and motivation example for the improvements presented in this paper, we deploy cell cycle modelling [31]. Cell cycle models are used to study the regulation of a cell during its replication and division. There are many kinetic models in the literature to study such biological behaviour. One important aspect of these models is that it is crucial to capture intrinsic noise in order to reproduce wet-lab experiments related to the variation in the age and volume of daughter cells [2, 22, 31]. In [22], Kar et al., constructed a stochastic model to study the effects of molecular fluctuation of species with low copies of molecules on the variation of cellular volume. This model includes many species with different

Table 2. Statistical information related to the three case studies used to evaluate the performance of Algorithm 2

Criteria/models	Circadian Oscillation	Cell Cycle Model (V1)	Cell Cycle Model (V2)
Number of species	9	26	60
Number of reactions	16	48	190
Number of stochastic reactions	3	19	19
Number of deterministic reactions	13	29	171
Number of stochastic events (exact)	35,650	780,318	112,908
Number of stochastic events (accelerated)	35,533	776,192	112,789
Run times (exact) (s)	3.8	731	495
Run times (accelerated) (s)	1.278	445	53
Number of interface reactions	0	8	0

abundance of molecules. Therefore, in [17,19,24], it has been suggested to use hybrid simulation to study the dynamics of this model. However, it has also been reported there that the hybrid simulation algorithm is computationally very slow when simulating this model [24].

In [2], a new deterministic and stochastic model to study the yeast cell cycle has been developed. The dimension of the model has been substantially increased, as it is based on the approach of multi-site phosphorylation to reproduce the level of nonlinearity that is required to reconstruct the bistable switch behaviour. Similarly, stochastic simulation is of paramount importance to capture the variation of the cellular volume. However, simulation becomes more intricate due to the many species and reactions involved in the construction of this model.

In this section we use the accelerated hybrid simulation algorithm to study the behaviour of this model. We adopt two model versions. The simpler one (V1) is based on the stochastic and hybrid models constructed in [22] and [19], respectively, and the bigger one (V2) on the model in [2]. A summary of the model information as well as the result statistics of applying Algorithm 2 are given in Table 2. To run the simulation, we use the same kinetic information as well as initial state as given in [2,22].

The two models have been statically partitioned such that all reactions related to mRNAs are stochastically simulated, while all other reactions are deterministically simulated. Figures 2 and 3 present the simulation results of a single run of the exact as well as the accelerated version of the hybrid simulation for the cell cycle model V1 and V2, respectively. Both simulator versions successfully capture the variation of the cell volume over the whole simulation time. However, the accelerated simulator version is ten times faster than the exact one as it has been illustrated in Table 2.

We use again the number of stochastic events as a measure of the simulation accuracy. According to the data in Table 2, both simulation approaches generate

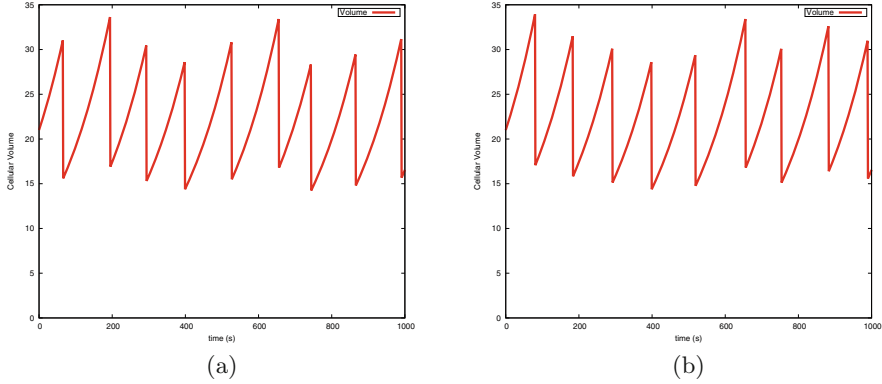


Fig. 2. Results of the yeast cell cycle model V1 (cellular volume) when simulated via (a) the exact algorithm, (b) the accelerated algorithm. Please note that single runs were used to generate this figure

comparable numbers of stochastic events. However, as the many reinitialisations of the ODE solver have been avoided, the accelerated version runs much faster than the exact one. As another measure of simulation accuracy, we refer to the number of divisions during the whole simulation period. The cell cycle model V1 produces nine divisions in both the exact and accelerated simulators, while the cell cycle model V2 results in five divisions when running the two simulators.

The cell cycle model in its first version (V1) is much smaller in terms of number of species and reactions compared to the larger second version (V2). However, the number of stochastic events is much higher due to the abundance

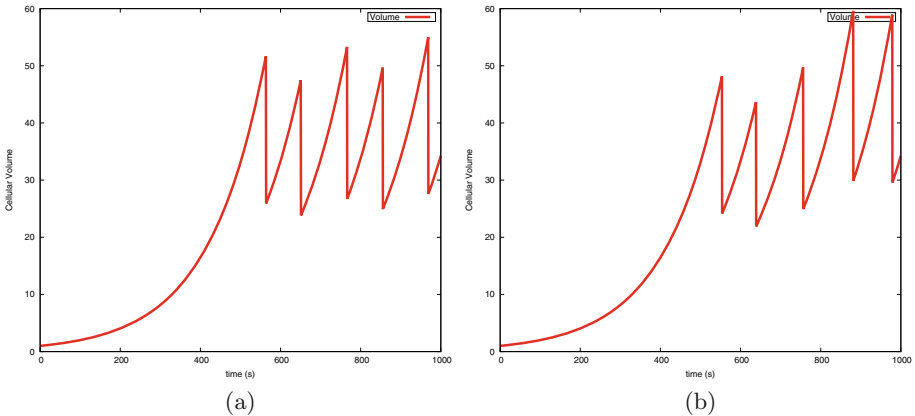


Fig. 3. Results of the yeast cell cycle model V2 (cellular volume) when simulated via (a) the exact algorithm, (b) the accelerated algorithm. Please note that single runs were used to generate this figure

of molecules of the mRNA species. Moreover, it was not feasible to completely isolate the deterministic and stochastic parts as we did for V2. The cell cycle model V1 contains 8 interface reactions which have an noticeable effect on the runtime behaviour of the model simulation due to the required reinitialisations, while there are no interface reactions in V2.

5 Conclusions and Future Work

In this paper, we have presented an approach for improving the performance of hybrid simulation algorithms that combine stochastic and (continuous) deterministic solvers. The proposed method has been tested using three case studies; and the test results have also been presented. The suggested improvements will be useful to cope with the rapid growth of (biological) models.

According to the test results given in Table 2 it is obvious that there is a noticeable improvement in terms of runtime even for small models. The speed of the accelerated algorithm is about three times faster than the exact method. However, this is not very significant since there are only a few continuous variables used inside the ODE solver. Avoiding the reinitialisation in this case will not save much runtime.

On the contrary, for bigger models, there is a substantial improvement in terms of runtime (about ten times faster). This result is due to saving the ODE solver from repeating the work required to build accuracy and history information. Therefore, as soon as the model size is increased, the simulator performance is also improved. Moreover, there is another advantage when using these improvements for larger models: as the number of stochastic events increases, the time step between two successive events will decrease. This will also increase the algorithm accuracy due to the calculation of the cumulative propensity via Eq. (5).

For smaller models with a few stochastic events, the time steps between two successive events will be larger. Therefore, the calculation in Eq. (5) will decrease the simulation accuracy. One workaround to this limitation is to switch to the exact method whenever the time step is large, while using the accelerated method when the time step is small. One measure to help deciding (dynamically) which approach to use is the current value of the cumulative propensity.

The partitioning process including the dependency algorithm presented in this paper can be automated such that users of the hybrid simulator obtain suggestions for the best partitioning settings that result in an optimal performance of the hybrid simulation, while pertaining an acceptable level of accuracy.

Furthermore, the set of slow and fast reactions are partitioned so that there is no direct dependency between the two groups. Fortunately, this has completely eliminated the reinitialisation step for each switch from the stochastic to the continuous solvers. While completely isolated subnets are not common for all biological model (cf., cell cycle model V1), the partitioning process can be designed so that the set of interface reactions is minimal.

Finally, in Sect. 4 we have presented three case studies to assess the performance of the proposed method. Adopting more examples with different sizes

and partitioning strategies will provide more insights about the performance of Algorithm 2.

Acknowledgments. This work has been partially funded by the GE-SEED grant (7934) which is administrated by STDF(Science and Technology Development Fund, Egypt) and DAAD (German Academic Exchange Service). We also acknowledge the helpful comments of the anonymous reviewers for improving a previous version of the paper.

A Reactions of the Circadian Oscillation Model

Table 3 provides a complete specification of the circadian oscillation model [32] used in Sect. 4.1. This reaction set has been derived from the system of ODEs given in [32] following the approach described in [13]. The given ODEs fulfil the criteria established in [29]; so the result is unique. See [3] for a graphical representation of this reaction set by use of Petri nets and their treatment in the different paradigms.

Table 3. The reaction set of the circadian oscillation model

#	Reaction	Propensity	Parameter Value
r_1	$gene1_active \xrightarrow{k_1} gene1 + A$	$k_1 \cdot gene1_active$	$k_1 = 50$
r_2	$gene1 + A \xrightarrow{k_2} gene1_active$	$k_2 \cdot gene1 \cdot A$	$k_2 = 1.0$
r_3	$gene2_active \xrightarrow{k_3} gene2 + A$	$k_3 \cdot gene2_active$	$k_3 = 100$
r_4	$gene2 + A \xrightarrow{k_4} gene2_active$	$k_4 \cdot gene2 \cdot A$	$k_4 = 1.0$
r_5	$\phi \xrightarrow{k_5} mRNA1$	$k_5 \cdot gene1_active$	$k_5 = 500$
r_6	$\phi \xrightarrow{k_6} mRNA1$	$k_6 \cdot gene1$	$k_6 = 50$
r_7	$mRNA1 \xrightarrow{k_7} \phi$	$k_7 \cdot mRNA1$	$k_7 = 10$
r_8	$\phi \xrightarrow{k_8} A$	$k_8 \cdot mRNA1$	$k_8 = 50$
r_9	$A \xrightarrow{k_9} \phi$	$k_9 \cdot A$	$k_9 = 1.0$
r_{10}	$A + R \xrightarrow{k_{10}} A_R$	$k_{10} \cdot A \cdot R$	$k_{10} = 2.0$
r_{11}	$\phi \xrightarrow{k_{11}} mRNA2$	$k_{11} \cdot gene2_active$	$k_{11} = 50$
r_{12}	$\phi \xrightarrow{k_{12}} mRNA2$	$k_{12} \cdot gene2$	$k_{12} = 0.01$
r_{13}	$mRNA2 \xrightarrow{k_{13}} \phi$	$k_{13} \cdot mRNA2$	$k_{13} = 0.5$
r_{14}	$\phi \xrightarrow{k_{14}} R$	$k_{14} \cdot mRNA2$	$k_{14} = 5$
r_{15}	$R \xrightarrow{k_{15}} \phi$	$k_{15} \cdot R$	$k_{15} = 0.08$
r_{16}	$A_R \xrightarrow{k_{16}} R$	$k_{16} \cdot A_R$	$k_{16} = 1.0$

References

1. Alfonsi, A., Cancès, E., Turinici, G., Ventura, B., Huisinga, W.: Adaptive simulation of hybrid stochastic and deterministic models for biochemical systems. *ESAIM: Proc.* **14**, 1–13 (2005)
2. Barik, D., Baumann, W.T., Paul, M.R., Novak, B., Tyson, J.J.: A model of yeast cell-cycle regulation based on multisite phosphorylation. *Molecular Syst. Biol.* **6**(1), 405 (2010)
3. Blätke, M., Heiner, M., Marwan, W.: *BioModel engineering with Petri nets*, chap. 7, pp. 141–193. Elsevier Inc. (2015). <http://store.elsevier.com/product.jsp?isbn=9780128012130>
4. Cao, Y., Gillespie, D., Petzold, L.: Adaptive explicit-implicit tau-leaping method with automatic tau selection. *J. Chem. Phys.* **126**(22), 224101 (2007)
5. Gibson, M., Bruck, J.: Exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem.* **105**, 1876–89 (2000)
6. Gillespie, D.: Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**(25), 2340–2361 (1977)
7. Gillespie, D.: *Markov Processes: An Introduction for Physical Scientists*. Academic Press, San Diego (1991)
8. Gillespie, D.: Stochastic simulation of chemical kinetics. *Annual Rev. Phys. Chem.* **58**(1), 35–55 (2007)
9. Griffith, M., Courtney, T., Peccoud, J., Sanders, W.H.: Dynamic partitioning for hybrid simulation of the bistable HIV-1 transactivation network. *Bioinformatics* **22**(22), 2782–2789 (2006)
10. Hairer, E., Wanner, G.: *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer Series in Computer Mathematics. Springer Series in Computer Mathematics, vol. 14. Springer, Berlin (1996)
11. Haseltine, E., Rawlings, J.: Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *J. Chem. Phys.* **117**(15), 6959–6969 (2002)
12. Heiner, M., Herajy, M., Liu, F., Rohr, C., Schwarick, M.: Snoopy – a unifying Petri net tool. In: Haddad, S., Pomello, L. (eds.) *PETRI NETS 2012*. LNCS, vol. 7347, pp. 398–407. Springer, Heidelberg (2012)
13. Hellander, A., Lötstedt, P.: Hybrid method for the chemical master equation. *J. Comput. Phys.* **227**(1), 100–122 (2007)
14. Herajy, M., Heiner, M.: Hybrid representation and simulation of stiff biochemical networks. *J. Nonlinear Anal. Hybrid Syst.* **6**(4), 942–959 (2012)
15. Herajy, M., Heiner, M.: A steering server for collaborative simulation of quantitative Petri nets. In: Ciardo, G., Kindler, E. (eds.) *PETRI NETS 2014*. LNCS, vol. 8489, pp. 374–384. Springer, Heidelberg (2014)
16. Herajy, M., Liu, F., Rohr, C.: Coloured hybrid Petri nets for systems biology. In: *Proceedings of the 5th International Workshop on Biological Processes & Petri Nets (BioPPN)*, Satellite Event of PETRI NETS 2014, CEUR Workshop Proceedings, vol. 1159, pp. 60–76 (2014)
17. Herajy, M., Schwarick, M.: A hybrid Petri net model of the eukaryotic cell cycle. In: *Proceedings of the 3rd International Workshop on Biological Processes and Petri Nets (BioPPN)*, Satellite Event of PETRI NETS 2012, CEUR Workshop Proceedings, vol. 852, pp. 29–43 (2012). CEUR-WS.org. <http://ceur-ws.org/Vol-852/>
18. Herajy, M., Heiner, M.: Modeling and simulation of multi-scale environmental systems with generalized hybrid Petri nets. *Front. Environ. Sci.* **3**(53) (2015)

19. Herajy, M., Schwarick, M., Heiner, M.: Hybrid Petri nets for modelling the eukaryotic cell cycle. In: Koutny, M., Aalst, W.M.P., Yakovlev, A. (eds.) *ToPNoC VIII*. LNCS, vol. 8100, pp. 123–141. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-40465-8_7](https://doi.org/10.1007/978-3-642-40465-8_7)
20. Hindmarsh, A., Brown, P., Grant, K., Lee, S., Serban, R., Shumaker, D., Woodward, C.: Sundials: suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.* **31**, 363–396 (2005)
21. Iwamoto, K., Hamada, H., Eguchi, Y., Okamoto, M.: Stochasticity of intranuclear biochemical reaction processes controls the final decision of cell fate associated with DNA damage. *PLoS ONE* **9**, 1–12 (2014)
22. Kar, S., Baumann, W.T., Paul, M.R., Tyson, J.J.: Exploring the roles of noise in the eukaryotic cell cycle. *Proc. Natl. Acad. Sci. U.S.A.* **106**(16), 6471–6476 (2009)
23. Kiehl, T., Mattheyses, R., Simmons, M.: Hybrid simulation of cellular behavior. *Bioinformatics* **20**, 316–322 (2004)
24. Liu, Z., Pu, Y., Li, F., Shaffer, C.A., Hoops, S., Tyson, J.J., Cao, Y.: Hybrid modeling and simulation of stochastic effects on progression through the eukaryotic cell cycle. *J. Chem. Phys.* **136**(3), 034105 (2012)
25. Mcadams, H., Arkin, A.: It's a noisy business!. *Trends Genet.* **15**(2), 65–69 (1999)
26. Pahle, J.: Biochemical simulations: stochastic, approximate stochastic and hybrid approaches. *Brief Bioinform.* **10**(1), 53–64 (2009)
27. Rathinam, M., Petzold, L., Cao, Y., Gillespie, D.: Stiffness in stochastic chemically reacting systems: the implicit tau-leaping method. *J. Chem. Phys.* **119**, 12784–12794 (2003)
28. Salis, H., Kaznessis, Y.: Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *J. Chem. Phys.* **122**(5), 54103 (2005)
29. Soliman, S., Heiner, M.: A unique transformation from ordinary differential equations to reaction networks. *PLoS ONE* **5**(12), e14284 (2010)
30. Srivastava, R., You, L., Summers, J., Yin, J.: Stochastic vs. deterministic modeling of intracellular viral kinetics. *J. Theor. Biol.* **218**(3), 309–321 (2002)
31. Tyson, J.J., Novk, B.: Irreversible transitions, bistability and checkpoint controls in the eukaryotic cell cycle: a systems-level understanding, Chapt. 14. In: Walhout, A.M., Vidal, M., Dekker, J. (eds.) *Handbook of Systems Biology*, pp. 265–285. Academic Press, San Diego (2013)
32. Vilar, J., Kueh, H., Barkai, N., Leibler, S.: Mechanisms of noise resistance in genetic oscillators. *PNAS* **99**, 59885992 (2002)

Hybrid Systems Biology

5th International Workshop, HSB 2016, Grenoble,
France, October 20-21, 2016, Proceedings

Cinquemani, E.; Donzé, A. (Eds.)

2016, X, 179 p. 43 illus., Softcover

ISBN: 978-3-319-47150-1