RSSALib: A library for stochastic simulation of complex biochemical reactions

Supplementary Material

Vo Hong Thanh

Contents

1 Implementations of simulation algorithms in RSSALib 2
  1.1 RSSA ............................................. 2
  1.2 Delayed RSSA ................................... 4
  1.3 Efficient formulations of RSSA .................... 6

2 Model description and application interfaces 7

3 Simulation validation 9

4 Numerical case studies 11
  4.1 Hes1 model ........................................ 11
  4.2 The heat-shock response model ................. 12
  4.3 FcεRI model ...................................... 14
1 Implementations of simulation algorithms in RSSALib

In the following, we describe the detailed implementations of the rejection-based stochastic simulation algorithm (RSSA) and its various efficient formulations [1, 2, 3, 4, 5] for computing stochastic dynamics of biochemical reactions. We refer to [6, 7, 8, 9] for recent developments and extensions as well as the monograph book [10] for a comprehensive review.

1.1 RSSA

RSSA is outlined in Algorithm 1. It computes for each reaction $R_j$ a propensity lower bound $a_j$ and an upper bound $\bar{a}_j$ such that $a_j \leq a_j \leq \bar{a}_j$, for $j = 1, \ldots, M$. These propensity bounds are derived by imposing an arbitrarily bound $[X, \bar{X}]$, called fluctuation interval, on the population of each species in the state $X(t)$. Specifically, consider a species $S_i$. A lower bound $X_i$, and an upper bound $\bar{X}_i$, is defined as $[X_i, \bar{X}_i] = [(1 - \delta_i)X_i(t), (1 + \delta_i)X_i(t)]$, where $0 \leq \delta_i \leq 1$ is a parameter. If the population of species is $X_i(t) < \lambda$, we will define fluctuation interval as $[X_i, \bar{X}_i] = [X_i(t) - \delta, X_i(t) + \delta]$. We refer to [2] for an analysis of the effects of fluctuation interval on the simulation performance. In our implementation, we set $\delta_i = 0.2$, $\lambda = 25$ and $\delta = 4$ because they often give better performance. Having the fluctuation interval $[X, \bar{X}]$, the propensity bound $[a_j, \bar{a}_j]$ obtained by optimizing the propensity function $a_j$ over such the fluctuation interval. For reaction kinetics such as the mass action kinetics or the Michaelis-Menten kinetics where the propensity $a_j$ is monotonic function of the state, the propensity bounds can be computed easily by putting $a_j = a_j(X)$ and $\bar{a}_j = a_j(\bar{X})$.

The use of propensity bounds $[a_j, \bar{a}_j]$ with $j = 1, \ldots, M$ to select the next reaction firing and generate its firing time is in lines 10 - 22. It consists of two steps. First, a candidate reaction $R_\mu$ is selected with probability $\bar{a}_\mu/\bar{a}_\mu$ where $\bar{a}_\mu = \sum_{j=1}^m \bar{a}_j$. The candidate reaction is realized by summing propensity upper bounds until it finds the smallest reaction index $\mu$ satisfying the inequality $\sum_{j=1}^\mu \bar{a}_j > r_1 \cdot \bar{a}_\mu$ where the random number $r_1 \sim U(0, 1)$. The candidate reaction $R_\mu$ then enters a rejection test with success probability $a_\mu/\bar{a}_\mu$. This test requires to evaluate the exact propensity $a_\mu$, but RSSA tries to avoid computing it by exploiting the fact that if the candidate is accepted with probability $a_\mu/\bar{a}_\mu$, then it is also accepted with $a_\mu/\bar{a}_\mu$ because of $a_\mu \leq a_\mu$. To do that, RSSA generates a random number $r_2 \sim U(0, 1)$ and first checks whether the condition $r_2 \leq a_\mu/\bar{a}_\mu$ holds. Only if the test fails, $a_\mu$ is evaluated with the current state and then is used to check again the condition $r_2 \leq a_\mu/\bar{a}_\mu$. If $R_\mu$ is accepted after the rejection test, its firing time is then computed. Otherwise, a new selection is repeated.

The firing time $\tau$ of the accepted reaction $R_\mu$ in RSSA is generated following an $Erlang(k, \bar{a}_0)$ distribution where the shape parameter $k$ is the number of trials until the acceptance and the rate parameter $\bar{a}_0$ is the sum of propensity upper bounds. It is in fact the sum of $k$ exponentially distributed random numbers with the same rate $\bar{a}_0$. Line 23 implements the generation of the $Erlang$-distributed firing time as $\tau = (-1/\bar{a}_0) \ln(\prod_{i=1}^k u_i)$ where $u_i$ is a random number from $U(0, 1)$. 
Algorithm 1 Rejection-based Stochastic Simulation Algorithm (RSSA)

**Input:** a biochemical reaction network of \( M \) reactions, the initial state \( x_0 \) at time 0 and the simulation ending time \( T_{\text{max}} \)

**Output:** a trajectory \( X(t), 0 \leq t \leq T_{\text{max}} \), of the biochemical reaction network

1: initialize time \( t = 0 \) and state \( X = x_0 \)
2: build the species-reaction (SR) dependency graph \( G \)
3: define a new \( [X_i, X_i] \) around \( X_i \), for \( i = 1, \ldots, N \)
4: compute propensity bounds \( \overline{a}_j \) and \( \underline{a}_j \) for \( j = 1, \ldots, M \)
5: compute total sum \( \overline{a}_0 = \sum_{j=1}^{M} \overline{a}_j \)
6: while \( (t < T_{\text{max}}) \) do
   7:     repeat
   8:         set \( u = 1 \)
   9:         set \( accepted = false \)
   10:        repeat
   11:            generate three random numbers \( r_1, r_2, r_3 \sim \mathbb{U}(0, 1) \)
   12:            select \( R_\mu \) with minimum index \( \mu \) satisfied \( \sum_{j=1}^{\mu} \overline{a}_j \geq r_1 \overline{a}_0 \)
   13:            if \( (r_2 \leq (a_\mu / \overline{a}_\mu)) \) then
   14:                set \( accepted = true \)
   15:            else
   16:                evaluate \( a_\mu \) with state \( X \)
   17:                if \( (r_2 \leq (a_\mu / \overline{a}_\mu)) \) then
   18:                    set \( accepted = true \)
   19:            end if
   20:        end if
   21:        set \( u = u \cdot r_3 \)
   22:    until \( (accepted) \)
   23:    compute firing time \( \tau = (-1/\overline{a}_0) \ln(u) \)
   24:    update state \( X = X + v_\mu \)
   25:    set time \( t = t + \tau \)
   26:    until (exists \( (X_i \notin [X_i, X_i]) \))
   27:    for all \( (X_i \notin [X_i, X_i]) \) do
   28:        define a new \( [X_i, X_i] \) around \( X_i \)
   29:        for all \( (R_j \in \text{ReactionsAffectedBy}(S_i)) \) do
   30:            compute new propensity bounds \( \overline{a}_j \) and \( \underline{a}_j \)
   31:        update \( \overline{a}_0 \)
   32:    end for
   33: end for
   34: end while
Having the reaction $R_\mu$ and its firing time $\tau$, the simulation updates the state $X$ and simulation time $t$ accordingly. However, RSSA does not recompute propensity bounds after each reaction firings if the state $X$ is still confined in the current fluctuation interval $[X, \overline{X}]$, which is often the case because only a few species are changed by a reaction. Only in uncommon cases where the state is outside the current fluctuation interval, i.e. $X(t) \notin [X, \overline{X}]$, a new fluctuation interval is defined and new propensity bounds for reactions are derived. RSSA further reduces the number of reactions having to recompute their propensity bounds by applying a Species-Reaction (SR) dependency graph (built in line 2). The SR dependency graph is a bipartite directed graph showing the dependency of species and reactions. A directed edge between a species $S_i$ and a reaction $R_j$ is included in the SR graph if changes in the population of $S_i$ affect the propensity $a_j$. Let $\text{ReactionsAffectedBy}(S_i)$ be the set of reactions affected by species $S_i$ extracted from the SR graph, then only reaction $R_j \in \text{ReactionsAffectedBy}(S_i)$ should recompute their propensity bounds when species $S_i$ exits its fluctuation interval.

### 1.2 Delayed RSSA

Delayed RSSA is an extension for biochemical reactions with time delays. Reactions with delays are divided into three types [11]: 1) reactions with no delay (ND), 2) consuming delayed reactions (CD) and 3) nonconsuming delayed reactions (NCD). The update of the state depends on the type of the reaction. ND reactions update the populations of reactants and products instantly at initiation, but CD and NCD reactions take a delay time for updating the populations of reactants and products. When a CD reaction occurs, the populations of its reactants change immediately at initiation while its products will be updated at completion. The update of populations of reactants and products of CD reactions is performed separately both at initiation and completion. Because NCD reactions update the populations of reactants and products only at completion, the reactants of a NCD reaction thus can participate in a new reaction even if the reaction has not finished.

The key of Delayed RSSA in Algorithm 2 is a race where the next event, which can be a reaction initiation or delay event, that is selected to update the state is the event having smallest time. Let $\tau$ be the waiting time to the next reaction initiation, and receptively, $t_d$ be the time to next delay event caused by a delayed reaction on the top of the delay event queue. There are two cases. If $t + \tau < t_d$, the next event would be a reaction initiation. The next reaction will be selected with probability $a_\mu/a_0$ to update state. In the other case, a next delay event will be extracted to take place and update the state. The simulation loops over all delayed events with completion time before $t + \tau$ to update the state. If the state caused by the occurrence of a delayed event moves out of its fluctuation interval, then a new firing time will be generated by recycling the old time (lines 10-22). The SR dependency graph is also used to speed up the update of propensity bounds of reactions affected by species whose population moves out of their fluctuation intervals.
Algorithm 2 Delayed RSSA

1: initialize time $t = 0$ with state $X = x_0$
2: build the species-reaction (SR) dependency graph $SRG$
3: define fluctuation interval $[X_i, \overline{X}_i]$ around $X_i$ for each species $S_i$ with $i = 1 \ldots n$
4: compute propensity bounds $\overline{a_j}$ and $\underline{a_j}$, $j = 1 \ldots m$, and sum $\overline{a_0} = \sum_{j=1}^{m} \overline{a_j}$
5: while ($t < T_{max}$) do
6: generate random numbers $r_1, r_2, r_3 \sim U(0, 1)$
7: compute $\tau = (-1/\overline{a_0}) \ln(r_1)$
8: set $t_{next} = t + \tau$
9: repeat
10: while (exists delayed event $(R_j, t_d)$ on top of delayed event queue and $t_d < t_{next}$) do
11: case $R_j$ of
12: CD: update state by products of $R_j$
13: NCD: update state by reactants and products of $R_j$
14: end case
15: remove the top of delayed event queue
16: if (exists $X_i \not\in [X_i, \overline{X}_i]$) then
17: define new fluctuation interval $[X_i, \overline{X}_i]$ for species $S_i$
18: compute new propensity bounds $\overline{a_j}^{new}$ and $\underline{a_j}^{new}$ for reactions affected by species extracting from $SRG$ and sum $\overline{a_0}^{new} = \sum_{j=1}^{M} \overline{a_j}^{new}$
19: update $t_{next} = t_d + (\overline{a_0}/\overline{a_0}^{new})(t_{next} - t_d)$
20: set $\overline{a_j} = \overline{a_j}^{new}$ for reactions and $\overline{a_0} = \overline{a_0}^{new}$
21: end if
22: end while
23: select a reaction $R_\mu$ with probability $\overline{a_j}/\overline{a_0}$ by inverse transformation of $r_2$
24: case $R_\mu$ of
25: ND: update state by reactants and products of $R_\mu$
26: CD with delay $\tau_d$: update state by reactants of $R_\mu$ and enqueue event $(\mu, t + \tau_d)$
27: NCD with delay $\tau_d$: enqueue an event $(\mu, t + \tau_d)$
28: end case
29: set $t = t_{next}$
30: until (exists $X_i \not\in [X_i, \overline{X}_i]$)
31: define new fluctuation interval $[X_i, \overline{X}_i]$ for species $S_i$
32: compute new propensity bounds $\overline{a_j}$ and $\underline{a_j}$ for reactions affected by species extracting from $SRG$
33: update sum $\overline{a_0}$
34: end while
1.3 Efficient formulations of RSSA

The search for the candidate reaction in RSSA is equivalent to a linear search, which is not efficient for large reaction networks. We therefore implement several efficient formulations to the original RSSA to improve its computational efficiency in simulating large reaction networks. These improvements include.

**RSSA with modified search method** [4]. The formulation focuses on improving the search of the next reaction $R_\mu$ in RSSA by reusing the previously computed sum of propensities in the last step, hence improving the cache-friendliness. Specifically, let $F_k = \sum_{j=1}^{k} \pi_j$ be the sum of reaction propensities up to reaction $R_k$ in the reaction list. The search moves forward in the list of reactions if $F_k < r_1 \overline{a_0}$, where $r_1$ be a random number in $\mathbb{U}(0, 1)$. Otherwise, it moves backward by subtracting $F_k$ by propensities $a_j$.

**Partial-propensity RSSA (PRSSA)** [5]. The formulation employs the factorization of the mass-action propensity $a_j(X(t)) = X_i(t) \pi_{ij}(X(t))$, where $\pi_{ij}(X(t))$ is the partial-propensity with respect to species $S_i$, to factorize the propensity bounds $a_j/a_j$ of all reactions. Specifically, by applying interval analysis, we can factorize $a_j = X_i \pi_{ij}(X(t))$ and $\overline{a_j} = X_i \overline{\pi_{ij}(X(t))}$. The partial propensity upper bounds $\pi_{ij}(X(t))$ and lower bounds $\pi_{ij}(X(t))$ are grouped by common reactant species $S_i$, for $i = 1, \ldots, N$, in the so-called partial propensity structure. The selection of candidate reaction in the PRSSA is performed in two consecutive search steps in which the first one selects a group and then the second locates the reaction in that group. The time complexity of the search in PRSSA is proportional to the number of species, i.e., $O(N)$, instead of the number of reactions.

**RSSA with tree-based search (RSSA-Binary)** [2]. Tree-based search method provides an alternative for selecting the candidate reaction in logarithmic time. The approach requires to build a (binary) tree in which its leave store propensities $\overline{a_j}$ of reactions and internal nodes store the sum value of their children. In our implementation, the tree is represented by an array with $2^M - 1$ elements where $M$ is the number of reactions. The array in the tree search requires $O(m)$ more elements than linear search because it has to store also partial sum of propensity bounds of reactions. The search for the candidate reaction $R_\mu$ with the search value $r_1 \cdot \overline{a_0}$, where $r_1$ be a random number in $\mathbb{U}(0, 1)$, is as follows. Starting at the tree root, denoting as the current node, the search selects the next branch by comparing on the search value with the value stored in the left child of the current node. If the search value is less than this value, the search selects the left branch to traverse. Otherwise, the right branch is chosen. If the right branch is selected, the search value is subtracted by the value stored in current node. The search repeats until it reaches a leaf, whose the reaction in this leaf is chosen as the candidate. The depth of the tree, hence the search cost of the RSSA-Binary, is proportional to logarithmic of the number of reactions. Furthermore, because the structure of the tree remains unchanged during the simulation, the computation of RSSA-Binary is stable.

**RSSA with composition-rejection search (RSSA-CR)** [3]. The formulation employs the composition-rejection search method to reduce the search time complexity to be independent with the number of reactions. In order to do that reactions are first partitioned
into \( L \) groups \( G_i, i = 1, \ldots, L \), depending on their propensity bounds \( \pi_j \). Specifically, a reaction \( R_j \) is put into a group \( G_i \) if its propensity \( \pi_j \) satisfies \( 2^{q_i-1} \leq \pi_j \leq 2^{q_i} \). The selection of the candidate reaction in the composition-rejection search is composed of two steps. First, it selects the group \( G_i \) proportional the sum of propensity bounds of reactions in that group. Then, the next reaction firing \( R_{\mu} \) in the group \( G_i \) is located by applying the acceptance-rejection with hat function \( 2^{q_i} \). Because the acceptance probability of reaction \( R_{\mu} \) in the group \( G_i \) is \( \pi_{\mu}/2^{q_i} \geq 1/2 \), the average numbers of trials until a candidate is selected is 2. The composition-rejection search thus depends only on the number of groups, i.e., \( O(L) \). If the number of groups is bounded by a small constant, the search time complexity is constant.

**RSSA with table lookup search (RSSA-lookup)** [2]. The formulation uses the table lookup search, called the Alias method, for the selection of the candidate reaction. The principle of the method is to partition the \( M \) probabilities \( \pi_j/\pi_0 \) for \( j = 1 \ldots M \) into an equi-probable mixture of \( M \) two-point distributions and store these values in two tables, called the cut-off table and alias table. The former table stores the probability of the first values of the two-point mixtures and the latter contains the alias to the second parts of the mixtures. For selection of the next reaction, a random number \( r_1 \sim \mathcal{U}(0, 1) \) is first used to lookup the position of the equi-probable mixture. It is then rescaled to select which part of the two-point. These steps take only one comparison to choose the part of the two-point mixture and (at most) two table accesses to select the candidate reaction. The search of the candidate reaction in RSSA-Lookup is thus constant. Its drawback, however, is that it requires to build the lookup tables which are linear time in the number of reactions.

## 2 Model description and application interfaces

We adopt a simple and flexible model representation to describe biochemical reactions. A model supported by RSSALib consists of three parts: 1) the definition of constants, 2) initial populations of species, and 3) reactions between species accompanying with kinetic information. Fig. 1 shows an example of the Hes1 dimer model used in Section 4. RSSALib also provides a SBML (Systems Biology Markup Language) importer to allow importing a reaction model in SBML format [12]. The importer relies on the JSBML API [13] to parse SBML.

The definition of a constant \( c \) in the model is specified by simple assignment as

\[
c = 1.0
\]

We use the same assignment for defining a species \( S \) and its initial population as

\[
S = 100
\]

We note that the population of a species should be an integer value. A reaction showing the interaction between species has the form

\[
v_1^- S_1 + \ldots + v_n^- S_n \rightarrow v_1^+ S_1 + \ldots + v_n^+ S_n, \text{rate } [\text{delay}]
\]
Figure 1: Hes1 dimer model used by RSSALib. The model consists of the definition of constants, species with their initial population, and reactions annotated with kinetics and time delays.

rate constants (min^-1)
c1 = 0.03
c2 = 0.06
c3 = 0.03
c4 = 0.001
c5 = 0.01
c6 = 10
c7 = 10

delayed time (min)
d = 20

threshold
s0 = 1000

initial population of species
S1 = 300
S2 = 10000
S3 = 10000

reactions
S1 -> _, c1
S2 -> _, c2
S3 -> _, c3
2S2 -> S3, c4
S3 -> 2S2, c5
S1 -> S1 + S2, INHIBITORYHILL(S1, c6, 5, s0), CD(d)
S3 -> S1 + S3, INHIBITORYHILL(S3, c7, 5, s0), CD(d)
where $v_i^-$ and $v_i^+$ denote the number of species $S_i$ that are consumed and produced by the reaction. The reaction is annotated with kinetics and time delay information. The reaction kinetics supported by RSSALib includes:

- **Mass-action kinetics** (default reaction kinetics): The rate is a constant value,

- **Michealis-Menten kinetics**, denoted by the keyword MM: The rate is computed as $MM(S, V_{\text{max}}, K_m) = \frac{V_{\text{max}} \#S}{K_m + \#S}$ with $S$ denoting the substrate, the maximum rate $V_{\text{max}}$ and Michaelis constant $K_m$, and

- **Hill kinetics**: There are two versions of the Hill kinetics: the exhibitory Hill, denoted by the keyword HILL, and respectively, the inhibitory Hill denoted by the keyword INHIBITORYHILL. The exhibitory Hill rate is $HILL(S, c, n, s_0) = \frac{c}{1 + (s_0/\#S)^n}$ where $S$ is the substrate, Hill coefficient $n$, the substrate concentration occupying half of the binding sites $s_0$ and a constant $c$. Similarly, the inhibitory Hill rate is $INHIBITORYHILL(S, c, n, s_0) = \frac{c}{1 + (\#S/s_0)^n}$.

The time delay is an optional part. There are two types of delayed reactions, i.e., consuming delayed reaction CD(d) and nonconsuming delayed reaction NCD(d) where $d$ is the delay until the completion of the reaction after it is initiated. A reaction by default is non delay.

A biochemical model can be simulated with RSSALib either through its click-and-run GUI or manually through its developer API. The use of RSSALib for building stand-alone applications is shown in Figure 2. In this usage, we first load the biochemical model, and call the runSim() method of the simulator to perform the simulation.

### 3 Simulation validation

We validate the implementation of RSSALib against the SBML discrete stochastic models test suite (DSMTS) developed by Evans et al. [14]. In particular, we test the correctness of RSA simulation results in four test cases: 1) the birth-death process
Figure 3: Validation of RSSALib against the SBML discrete stochastic models test suite (DSMTS). The x-axis shows the simulation time.
Table 1: Hes1 dimer model

| Reaction                                      | Rate Constant |
|-----------------------------------------------|---------------|
| $R_1: S_1 \rightarrow \cdot$                  | $c_1 = 0.03 \text{ min}^{-1}$ |
| $R_2: S_2 \rightarrow \cdot$                  | $c_2 = 0.06 \text{ min}^{-1}$ |
| $R_3: S_3 \rightarrow \cdot$                  | $c_3 = 0.03 \text{ min}^{-1}$ |
| $R_4: 2S_2 \rightarrow S_3$                   | $c_4 = 0.001 \text{ min}^{-1}$ |
| $R_5: S_3 \rightarrow 2S_2$                   | $c_5 = 0.01 \text{ min}^{-1}$ |
| Consuming delayed $R_6: S_1 \rightarrow S_1 + S_2$ | $c_6 = \frac{10}{1 + (\#S_1/s_0)^n} \text{ min}^{-1}$ |
| Consuming delayed $R_7: S_3 \rightarrow S_1 + S_3$ | $c_7 = \frac{10}{1 + (\#S_3/s_0)^n} \text{ min}^{-1}$ |

(dsms-001), 2) the immigration-death process (dsmts-002), 3) the Dimerisation process (dsmts-003), and 4) the batch immigration-death process (dsmts-004). We consider the each test reaction system within a single compartment and without external events.

Fig. 3 shows the mean and standard deviation (std) of population of species produced by RSSA in comparison with DSMTS test suite. The simulation results are obtained by executing $10^4$ simulation runs of RSSA in each test system. It is shown in the figure that RSSA strongly agrees with DSMTS test suite in all test cases, thus numerically validating the correctness of our implementation. We refer to the original work [1] for a formal proof of correctness of RSSA.

4 Numerical case studies

This section reports applications of RSSALib in three case studies: the Hes1 model [15], the heat-shock response of E. coli (HSR) [16] and the FccRI model [17]. The models are real biological networks, which are chosen to highlight the applicability and efficiency of RSSALib. The Hes1 model describes a complex reaction mechanisms and, to the best of the author’s knowledge, is not fully supported by available tools. HSR is chosen as a benchmark due to its stiffness property. The FccRI model is a large model that is used to show the superior performance of our computational tool.

4.1 Hes1 model

We use the Hes1-dimer model (see Fig. 1) to demonstrate the applicability of RSSALib in simulating a complex model. The model consists of three species, Hes1 mRNA (S1), monomeric Hes1 (S2), and dimeric Hes1 (S3) involved in seven reactions: where $n = 5$ and $s_0 = 1000$. The time delay of $R_6$ and $R_7$ set to 20 min.

We perform $10^4$ simulation runs of the Hes1 model with simulation time $T_m.ax = 500 \text{ min}$ and the initial populations of species $X(0) = (\#S_1(0), \#S_2(0), \#S_3(0)) = (300, 10000, 10000)$. Fig. 4 shows the population of dimeric Hes1 in the time interval [100, 500]. Due to effects of time delays, the population of dimeric Hes1 exhibits
Hes1 dimer model

Figure 4: Population (mean +/- std) of the dimeric hes1 in the time interval [100, 500].

Fig. 5 shows the CPU times of DelayedRSSA in simulating Hes1 model. Because there is no available tool fully supporting the complex model such as the Hes1 model, we decided to compare performance of DelayedRSSA with our implementation of the delayed modified next reaction method (DelayedMNRM) [18]. The figure shows that DelayedRSSA outperforms DelayedMNRM. Specifically, DelayedRSSA is about 4 times faster than DelayedMNRM in simulating the Hes1 model.

4.2 The heat-shock response model

HSR is a stiff biochemical system [16] consisting of 28 species and 61 reactions which models the response of E. coli to heat stress. For this model, We compare the performance of RSSA with other simulation algorithms implemented in the Dizzy software tool [19] and the StochKit2 [20]. For Dizzy, we consider two algorithms: the next reaction method (NRM) and the tau-leaping, an approximate stochastic simulation algorithm. For StochKit2, we consider SSA and let the tool choose the suitable exact simulation algorithm. We compute the average CPU time of each algorithm by performing $10^4$ repeated runs of the algorithm on the model with the simulation time $T_{max} = 100$.

Fig. 6 plots simulation performance of algorithms. The figure shows that for this model the performance of RSSA is comparable with StochKi2’s SSA and Dizzy’s NRM, even though RSSA is slightly faster.
Figure 5: Performance of DRSSA and DMNRM in simulating Hes1 model.

Figure 6: Performance of RSSA and Dizzy-NRM, Dizzy-tau-leaping and StochKit2-SSA in simulating HSR model.
4.3 FcεRI model

The high-affinity IgE receptor model, referred to as FcεRI model, developed by Liu et al. [17] is a large biochemical network of 380 species and 3862 reactions. It is used to highlight the significant computational efficiency of RSSALib in simulation of large models.

Figure 7 compares simulation runtimes of RSSA implemented in RSSALib with the Dizzy’s NRM and tau-leaping. The average CPU time of each algorithm is calculated by $10^4$ simulation runs of the algorithm on the model with the simulation time $T_{max} = 20$. For this large model, our RSSA algorithm is about 6 times faster than Dizzy’s NRM, and respectively, by order of magnitudes faster than Dizzy’s tau-leaping.

Figure 7 depicts the computational performance of efficient formulations of RSSA. The figure has shown that these formulations significantly reduce the computational times of the simulation in simulating large networks.
Figure 8: Performance of RSSA and its formulations.

References

[1] Vo H. Thanh, Corrado Priami, and Roberto Zunino. Efficient rejection-based simulation of biochemical reactions with stochastic noise and delays. *The Journal of Chemical Physics*, 141(13), 2014.

[2] Vo H. Thanh, Roberto Zunino, and Corrado Priami. On the rejection-based algorithm for simulation and analysis of large-scale reaction networks. *The Journal of Chemical Physics*, 142(24):244106, 2015.

[3] Vo H. Thanh, Roberto Zunino, and Corrado Priami. Efficient constant-time complexity algorithm for stochastic simulation of large reaction networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 14(3):657–667, 2016.

[4] Vo H. Thanh, Roberto Zunino, and Corrado Priami. Efficient stochastic simulation of biochemical reactions with noise and delays. *The Journal of Chemical Physics*, 146(8):084107, 2017.

[5] Vo H. Thanh. Stochastic simulation of biochemical reactions with partial-propensity and rejection-based approaches. *Mathematical biosciences*, 292:67–75, 2017.
[6] Vo H. Thanh and Corrado Priami. Simulation of biochemical reactions with time-dependent rates by the rejection-based algorithm. *The Journal of Chemical Physics*, 143(5):054104, 2015.

[7] Vo H. Thanh, Corrado Priami, and Roberto Zunino. Accelerating rejection-based simulation of biochemical reactions with bounded acceptance probability. *The Journal of Chemical Physics*, 144(22):224108, 2016.

[8] Vo H. Thanh. Efficient anticorrelated variance reduction for stochastic simulation of biochemical reactions. *IET Systems Biology*, 13(1):16–23, 2018.

[9] Vo H. Thanh, Roberto Zunino, and Corrado Priami. Efficient finite-difference method for computing sensitivities of biochemical reactions. *Proceedings of the Royal Society A*, 474(2218):20180303, 2018.

[10] Luca Marchetti, Corrado Priami, and Vo H. Thanh. *Simulation Algorithms for Computational Systems Biology*. Springer, 2017.

[11] Manuel Barrio, Kevin Burrage, Andr Leier, and Tianhai Tian. Oscillatory regulation of hes1: discrete stochastic delay modelling and simulation. *PLoS Comput. Biol.*, 2(9):1017–1030, 2006.

[12] Michael Hucka and et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[13] Andreas Dräger, Nicolas Rodriguez, Alexander Dörr Marine Dumousseau, Clemens Wrzodek, Nicolas Le Novère, Andreas Zell, and Michael Hucka. Jsbml: a flexible java library for working with sbml. *Bioinformatics*, 27(1):2167–2168, 2011.

[14] Thomas W. Evans, Colin S. Gillespie, and Darren J. Wilkinson. The sbml discrete stochastic models test suite. *Bioinformatics*, 24(2):285–286, 2008.

[15] André Leier, Tatiana T. Marquez-Lago, and Kevin Burrage. Generalized binomial tau-leap method for biochemical kinetics incorporating both delay and intrinsic noise. *The Journal of Chemical Physics*, 128(20):205107, 2008.

[16] David B. Straus, William A. Walter, and Carol A. Gross. The heat shock response of e. coli is regulated by changes in the concentration of σ32. *Bioinformatics*, 329:348–351, 1987.

[17] Yanli Liu and et al. Single-cell measurements of IgE-mediated FcεRI signaling using an integrated microfluidic platform. *PLoS ONE*, 8(3):60159, 2013.

[18] David F. Anderson. A modified next reaction method for simulating chemical systems with time-dependent propensities and delays. *The Journal of Chemical Physics*, 127(21):214107, 2007.
[19] Stephen Ramsey, David Orrell, and Hamid Bolouri. Dizzy: stochastic simulation of large-scale genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*, 3(2):415–436, 2005.

[20] Kevin R. Sanft, Sheng Wu, Min Roh, Jin Fu, Rone Kwei Lim, and Linda R. Petzold. Stochkit2: software for discrete stochastic simulation of biochemical systems with events. *Bioinformatics*, 27(17).