Journal of Coupled Systems and Multiscale Dynamics

doi:10.1166/jcsmd.2017.1124
J. Coupled Syst. Multiscale Dyn.
Vol. 5(2)/2330-152X/2017/001/013

Research Article

An adaptive Magnus expansion method for solving the chemical master equation with time-dependent propensities

Khanh N. Dinh and Roger B. Sidje*

Department of Mathematics, University of Alabama, Tuscaloosa, AL, USA

(Received: xx Xxxx xxxx. Accepted: xx Xxxx xxxx)

ABSTRACT

The chemical master equation (CME) is a system of ordinary differential equations (ODEs) to model the chemical interaction of molecular species. The largeness of the state space of the system makes solving the CME difficult, and this has motivated reduction strategies such as the finite state projection (FSP). Moreover, if the reaction rates are functions of the time, the CME becomes an ODE problem with time-dependent coefficients. Solution techniques include Monte Carlo algorithms, such as the stochastic simulation algorithm (SSA) or ODE solvers, such as Adams-PECE, Runge-Kutta and backward-differentiation formula (BDF). There are also Magnus-based solvers that have however not been thoroughly explored in the CME context. Here we introduce an adaptive time-stepping Magnus-SSA algorithm, in which the CME is solved using a Magnus expansion with not only a variable time-step but also with a variable state space that changes at each step via the SSA, and several error approximation approaches are attempted to monitor the adaptivity. We perform comparative tests against the classical Adams-PECE, Runge-Kutta and BDF methods on three biological problems, showing that the proposed adaptive Magnus-based variants can be efficient when the CME with time-dependent rates is stiff.

Keywords: Chemical Master Equation, Magnus Expansion, Stiff ODE Solvers, Adaptive Time-Stepping Scheme.

Section: Mathematical, Physical & Engineering Sciences, Life, Climate & Environmental Sciences, Coupled Human-Natural Systems, Social Sciences.

1. INTRODUCTION

The chemical master equation (CME) describes the dynamics of a chemically reacting system as a Markovian process. It is a common approach to modeling stochastic processes in molecular biology, and can also arise in other fields of science such as ecological networks, pharmacokinetic networks and social networks.[1] It is difficult to solve the CME directly because it can involve a very large, or even infinite, number of ordinary differential equations. Hence, the CME has traditionally been solved indirectly by using Monte Carlo methods, such as Gillespie's Stochastic Simulation Algorithm (SSA)[2,3] or the First Reaction Method (FRM).[4] These algorithms simulate reactive events in the chemical processes, and averaging their results from a large number of trajectories can offer insights into the model as well as statistics of interest. Approximate representations using stochastic differential equations (SDEs) yield other approaches.[5,6]

In recent years, there has been great interest to solve the CME directly, using notably the finite state projection method (FSP)[7] to reduce the size of the CME by setting bounds on the state space. We have surveyed many of the FSP variants that seek to either increase their

*Author to whom correspondence should be addressed.
Email: roger.b.sidje@ua.edu

accuracy or decrease their execution time.[8] However, these FSP algorithms can only be applied when reaction rates are constant. In biological problems where reaction rates can change over time, for instance due to cell volume increase or change in temperature, the CME may be approached with ODE solvers, such as Adams, Runge-Kutta, or backward-differentiation formula.

The Magnus expansion[9] offers an alternative approach to dealing with the CME with time-dependent rates. It expresses the solution to the ODE as the matrix exponential of an infinite series involving multiple integrals and nested commutators. Although originally a theoretical technique, it has recently been developed as a practical ODE solver.[10–14] The Magnus series is truncated, the terms are rearranged to optimize the execution time, and the integrals are approximated by a quadrature formula.

In this paper, we embed the SSA in the Magnus method to reduce the state space, which allows for lower run time while retaining the accuracy. We also revisit two error estimation techniques in the literature, as well as implement two other error control procedures, and employ these to make the Magnus method adaptive both in terms of the time step and the state space. The resulting algorithms are then tested against Adams, Runge-Kutta and backward-differentiation formula for three biological problems in which the CME with time-dependent rates emerges.

The paper is organized as follows: the chemical master equation is defined in Section 2. Section 3 revisits two Monte Carlo methods commonly used to solve the CME. Some traditional ODE solvers are described in Section 4, and the Magnus expansion as well as the Magnus-SSA algorithm are presented in Section 5. Section 6 covers several error estimation methods, together with the resulting adaptive time-stepping Magnus algorithms. Numerical tests are defined and results are given in Section 7, and concluding remarks follow in Section 8.

2. CHEMICAL MASTER EQUATION

Consider a chemical reaction system with N molecular species $S_1, \ldots, S_N$ that interact through M reactions $R_1, \ldots, R_M$. The *state vector* of the system is defined as

$$\boldsymbol{x}(t) = (x_1, \ldots, x_N)^T$$

where x_l is the count for species S_l at time t.

The likelihood for each reaction R_k to occur is the time-dependent *reaction rate* $c_k(t)$. The *propensity function* $\alpha_k(\boldsymbol{x}(t), t)$ at the current state $\boldsymbol{x}(t)$ and current time t is defined so that the probability of reaction R_k occurring during the infinitesimal time interval $[t, t+dt)$ is equal to $\alpha_k(\boldsymbol{x}(t), t)\, dt$. If this reaction occurs, the state vector is updated by the *stoichiometric vector* ν_k, which stores the changes in species counts:

$$\boldsymbol{x}(t) = \boldsymbol{x}(t) + \nu_k$$

The chemical master equation (CME)[15] formulates that $P(\boldsymbol{x}, t) = \text{Prob}\{\boldsymbol{x}(t) = \boldsymbol{x}\}$, the probability that the system is in state $\boldsymbol{x}$ at time t, satisfies:

$$\frac{dP(\boldsymbol{x}, t)}{dt} = \sum_{k=1}^{M} \alpha_k(\boldsymbol{x} - \nu_k, t) P(\boldsymbol{x} - \nu_k, t) - \sum_{k=1}^{M} \alpha_k(\boldsymbol{x}, t) P(\boldsymbol{x}, t) \qquad (1)$$

Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be the ordered set of n possible states, where $\boldsymbol{x}_i = (x_{1i}, \ldots, x_{Ni})^T$. Assuming that the system was initially at a known state $\boldsymbol{x}(0) \in \boldsymbol{X}$ at $t = 0$, Eq. (1) can be rewritten as a system of ordinary differential equations (ODEs) governing the change in $\boldsymbol{p}(t) = (P(\boldsymbol{x}_1, t), \ldots, P(\boldsymbol{x}_n, t))^T$:

$$\begin{cases} \dot{\boldsymbol{p}}(t) = \boldsymbol{A}(t) \cdot \boldsymbol{p}(t), \\ \boldsymbol{p}(0) = \boldsymbol{p}_0 \end{cases} \qquad (2)$$

from the known initial distribution $\boldsymbol{p}_0$, defined using

$$P(\boldsymbol{x}, 0) = \begin{cases} 1, & \text{if } \boldsymbol{x} = \boldsymbol{x}(0), \\ 0, & \text{if } \boldsymbol{x} \neq \boldsymbol{x}(0) \end{cases}$$

and the transition rate matrix $\boldsymbol{A}(t) = [a_{ij}(t)] \in \mathbb{R}^{n \times n}$ is

$$a_{ij} = \begin{cases} -\sum_{k=1}^{M} \alpha_k(\boldsymbol{x}_j, t), & \text{if } i = j \\ \alpha_k(\boldsymbol{x}_j, t), & \text{if } \boldsymbol{x}_i = \boldsymbol{x}_j + \nu_k \\ 0, & \text{otherwise} \end{cases}$$

Since the reaction rates c_k are time-dependent, $\boldsymbol{A}$ changes over time.

The state space $\boldsymbol{X}$ can be infinite in theory, but is kept finite in practice, although n can be very large. In this case, we can apply the finite state projection (FSP),[7] which reduces the state space to only the probable states during the time period of interest. The vectors $\boldsymbol{p}(t)$, $\boldsymbol{p}_0$ and matrix $\boldsymbol{A}(t)$ in (2) are then truncated to only values corresponding to this reduced finite state space.

We will discuss different approaches for solving the ODE system (2) in the Sections 4 and 5. For convenience, we will denote the ODE problem as

$$\dot{\boldsymbol{p}}(t) = f(t, \boldsymbol{p}(t)) \equiv \boldsymbol{A}(t) \cdot \boldsymbol{p}(t)$$

3. MONTE CARLO METHODS

To solve (2) indirectly, Monte Carlo methods simulate trajectories from the initial state $\boldsymbol{x}(0)$ at $t = 0$ to a prescribed final time t_f. Statistics of interest are then drawn from the final states of the trajectories, such as means of the species counts, variance, or marginal distributions. Many Monte Carlo methods are based on the works of Gillespie.[2,3] We describe here two methods, first reaction method (FRM) and stochastic simulation algorithm (SSA).

3.1. First Reaction Method

At every time step in each trajectory, the first reaction method (FRM)[4] seeks the reaction that occurs next, and the time it takes for it to occur. Below is a pseudocode for the overall procedure:

```
1. Start from initial time $t = 0$ and initial state $\boldsymbol{x} = \boldsymbol{x}(0)$
2. Generate $\xi_1, \ldots, \xi_M$, uniformly distributed in $(0, 1)$
3. For every reaction $k$, find the minimal positive number $\tau_k$ so that
```

$$\int_t^{t+\tau_k} \alpha_k(\boldsymbol{x}(t), u)du = \ln\left(\frac{1}{\xi_k}\right)$$

```
4. $\tau$ is the minimum of $\tau_1, \ldots, \tau_M$, and $j$ is the index of the reaction with $\tau = \tau_j$
5. Reaction $j$ is the first to occur next, at $t + \tau$. Update the state and time accordingly:
```

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \nu_k$$
$$t \leftarrow t + \tau$$

```
6. Return to step 2 until reaching final time $t_f$.
```

FRM and other Monte Carlo methods approximate the probability distribution of the system at the final time t_f by simulating a large number of trajectories, and computing the frequency of each state $\boldsymbol{x}$ in the state space as

$$P(\boldsymbol{x}, t_f) \approx \text{frequency}(\boldsymbol{x}) = \frac{n_{\boldsymbol{x}}}{n_{\text{total}}} \tag{3}$$

where $n_{\boldsymbol{x}}$ is the number of trajectories that end up in state $\boldsymbol{x}$, and n_{total} is the total number of trajectories.

FRM is an exact method, because the trajectories are generated according to the correct probability distributions. For each problem in our numerical tests, we use the FRM frequencies to compare the results of the ODE solvers.

3.2. Stochastic Simulation Algorithm

If the reaction rates are time-independent, the first reaction method and the stochastic simulation algorithm (SSA)[2,3] are equivalent and the pseudocode simplifies to:

```
1. Start from initial time $t = 0$ and initial state $\boldsymbol{x} = \boldsymbol{x}(0)$
2. Find the propensity sum $\alpha_{\text{sum}} = \sum_{k=1}^{M} \alpha_k(\boldsymbol{x}(t), t)$
3. Generate $\xi_1$ and $\xi_2$, uniformly distributed in $(0, 1)$
4. $j$ is the smallest integer so that
```

$$\sum_{k=1}^{j} \alpha_k(\boldsymbol{x}, t) > \xi_1 \alpha_{\text{sum}}$$

```
5. Compute $\tau = \ln(1/\xi_2)\alpha_{\text{sum}}$
6. Reaction $j$ is the first to occur next, at $t + \tau$. Update the state and time accordingly:
```

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \nu_k$$
$$t \leftarrow t + \tau$$

```
7. Return to step 2 until reaching final time $t_f$.
```

Note that the SSA is inexact for our purpose, because it does not account for the time dependencies of the reaction rates. However, SSA is much faster than FRM, because the latter contains M optimization problems and possibly many integration problems at every time step.

4. ODE SOLVERS

We will now describe several traditional ODE solvers for the purpose of solving (2) directly. A detailed introduction to different classes of ODE solvers can be found in Refs. [16–18]. Here we summarize the ones used in our numerical tests.

4.1. Adams

Adams methods form a family of linear multi-step methods,[19] among which are explicit Adams-Bashforth and implicit Adams-Moulton. Adams-Bashforth proceeds with the explicit formulae of order r:

$$t_{k+1} = t_k + h_k,$$
$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + h_k(\beta_{r-1}^{AB}\boldsymbol{f}_k + \cdots + \beta_0^{AB}\boldsymbol{f}_{k-r+1}),$$
$$\boldsymbol{f}_{k+1} = f(t_{k+1}, \boldsymbol{p}_{k+1})$$

where $\{\beta_i^{AB}\}_{i=0}^{r-1}$ are given analytically based on a Lagrange interpolation polynomial.

Adams-Mouton, on the other hand, is implemented as

$$t_{k+1} = t_k + h_k,$$
$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + h_k\left(\beta_r^{AM}\boldsymbol{f}_{k+1} + \cdots + \beta_0^{AM}\boldsymbol{f}_{k-r+1}\right),$$
$$\boldsymbol{f}_{k+1} = f(t_{k+1}, \boldsymbol{p}_{k+1})$$

where $\{\beta_i^{AM}\}_{i=0}^{r}$ are given analytically.

We use the *Adams-PECE* scheme by Shampine and Gordon,[20] which implements the implicit Adams-Moulton. The unknown $\boldsymbol{p}_{k+1} \approx \boldsymbol{p}(t_{k+1})$ is involved in both sides of the formula, leading to a nonlinear problem that is approximately solved with a fixed-point scheme starting from the solution of the explicit Adams-Bashforth.

4.2. Runge-Kutta

Runge-Kutta methods form a class of multistage, one-step iteration ODE solvers. The explicit Runge-Kutta of order r proceeds with the scheme

$$t_{k+1} = t_k + h_k,$$
$$\boldsymbol{y}_i = \boldsymbol{p}_k + h_k \sum_{j=1}^{i-1} m_{ij}^{RK} f(t_k + h_k c_j^{RK}, \boldsymbol{y}_j); \quad i = 1, \ldots, r,$$
$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + h_k \sum_{j=1}^{r} b_j^{RK} f(t_k + h_k c_j^{RK}, \boldsymbol{y}_j)$$

in which the coefficients $\{m_{ij}^{RK}\}_{i,j=1}^{s}$, $\{b_i^{RK}\}_{i=1}^{s}$ and $\{c_i^{RK}\}_{i=1}^{s}$ are defined by their Butcher-tableau.

In this comparison, we use the solver RK78 in the RKSUITE by Brankin et al.,[21] which is a reputed Runge-Kutta method that controls the error and stepsize by using embedded Runge-Kutta formulae with orders 7 and 8. The solutions are denoted as *Runge-Kutta* in the numerical tests.

4.3. Backward-Differentiation Formula

Backward-differentiation formula (BDF) methods are linear multi-step and follow the formula of order r:

$$t_{k+1} = t_k + h_k,$$
$$\boldsymbol{p}_{k+1} = h_k \beta_r^{\mathrm{BDF}} f(t_{k+1}, \boldsymbol{p}_{k+1}) + \alpha_{r-1}^{\mathrm{BDF}} \boldsymbol{f}_k + \cdots + \alpha_0^{\mathrm{BDF}} \boldsymbol{f}_{k-r+1},$$
$$\boldsymbol{f}_{k+1} = f(t_{k+1}, \boldsymbol{p}_{k+1})$$

where the coefficients $\{\alpha_i^{\mathrm{BDF}}\}_{i=0}^{r-1}$ and β_r^{BDF} are given analytically. The formula forms a nonlinear problem, because $\boldsymbol{p}_{k+1}$ appears on both sides.

We use the VODPK/BDF implementation[22] which has different options in solving the nonlinear problem. Here we confine to:

1. *BDF-GM-LU*0: Newton root finding scheme; each linear system in the implicit scheme is solved iteratively by SPIGMR (Scaled Preconditioned Incomplete GMRES), preconditioned by the incomplete LU0 decomposition, which discards elements not in the sparsity pattern of $\boldsymbol{A}$.
2. *BDF-LU*0: The linear system in the impiclit scheme is solved directly by incomplete LU0 decomposition.

5. MAGNUS-BASED METHODS

5.1. Magnus Expansion

The Magnus expansion[9–11] expresses the theoretical solution of (2) as

$$\boldsymbol{p}(t) = \exp(\boldsymbol{\Omega}_{(0,t)}) \cdot \boldsymbol{p}_0 \tag{4}$$

where the matrix exponential is defined as

$$\exp(\boldsymbol{\Omega}) = \sum_{l=0}^{\infty} \frac{\boldsymbol{\Omega}^l}{l!}$$

and $\boldsymbol{\Omega}_{(0,t)}$ can be written as an infinite series whose terms involve multiple integrals and nested commutators:

$$\begin{aligned} \Omega_1 &= \int_0^t \boldsymbol{A}(\tau) d\tau, \\ \boldsymbol{S}_n^{(1)} &= [\Omega_{n-1}, \boldsymbol{A}], \\ \boldsymbol{S}_n^{(j)} &= \sum_{m=1}^{n-j} [\Omega_m, \boldsymbol{S}_m^{j-1}]; \quad 2 \le j \le n-1, \\ \boldsymbol{S}_n^{(n-1)} &= \mathrm{ad}_{\Omega_1}^{n-1}(\boldsymbol{A}), \\ \Omega_n &= \sum_{j=1}^{n-1} \frac{B_j}{j!} \int_0^t \boldsymbol{S}_n^{(j)}(\tau) d\tau; \quad n \ge 2, \\ \Omega_{(0,t)} &= \sum_{n=1}^{\infty} \Omega_n \end{aligned} \tag{5}$$

where B_j are Bernoulli numbers and the adjoint representation is defined as

$$\mathrm{ad}_{\boldsymbol{A}}^j(\boldsymbol{B}) = [\boldsymbol{A}, \mathrm{ad}_{\boldsymbol{A}}^{j-1}(\boldsymbol{B})], \quad \mathrm{ad}_{\boldsymbol{A}}^0(\boldsymbol{B}) = \boldsymbol{B}$$

The Magnus integration method then follows:

$$\begin{aligned} t_{k+1} &= t_k + h_k, \\ \boldsymbol{p}_{k+1} &= \exp(\sigma_{(t_k,h_k)}^{[q]}) \cdot \boldsymbol{p}_k \end{aligned} \tag{6}$$

where $\boldsymbol{\sigma}_{(t_k,h_k)}^{[q]}$ is an approximation of $\boldsymbol{\Omega}_{(t_k,h_k)}$ to the qth order. To derive $\boldsymbol{\sigma}_{(t_k,h_k)}^{[q]}$, the Magnus expansion (5) is truncated to the first q terms, and the integrals are approximated by a quadrature rule.

The Magnus expansion has been employed extensively in physics where it is sometimes referred to as time-dependent exponential perturbation theory.[11] It has the appeal that its approximation preserves important qualitative properties of the exact solution.[13,23] It has also been used in the field of geometric numerical integration to reproduce important geometric structures in the solutions, a goal not straightforwardly possible with general-purpose integration methods.

We implement the 4th-order Magnus integration method using the Gauss-Legendre quadrature rule:[10,12,24]

$$\boldsymbol{A}_1 = \boldsymbol{A}\left(t_k + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) h_k\right),$$
$$\boldsymbol{A}_2 = \boldsymbol{A}\left(t_k + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right) h_k\right),$$
$$\sigma_{(t_k,h_k)}^{[4]} = \frac{h_k}{2}(\boldsymbol{A}_1 + \boldsymbol{A}_2) + \frac{h_k^2 \sqrt{3}}{12} [\boldsymbol{A}_2, \boldsymbol{A}_1]$$

5.2. Krylov Subspace Technique

The term $\exp(\boldsymbol{\sigma}_{(t_k,h_k)}^{[4]}) \cdot \boldsymbol{p}_k$ in (6) is computed by Expokit, which implements a Krylov-based algorithm that seeks to approximate $\exp(\boldsymbol{\sigma}) \cdot \boldsymbol{p}$, the action of the matrix exponential on a vector, as a projection in the Krylov subspace of order m

$$\mathcal{K}_m(\sigma, \boldsymbol{p}) = \mathrm{span}\{\boldsymbol{p}, \sigma \cdot \boldsymbol{p}, \ldots, \sigma^{m-1} \cdot \boldsymbol{p}\}$$

The Arnoldi process is employed to compute an orthonormal basis $\boldsymbol{V}_m$ of this subspace, and an associated Hessenberg matrix $\boldsymbol{H}_m$. The Krylov approximation is then

$$\exp(\boldsymbol{\sigma}) \cdot \boldsymbol{p} \approx \beta \boldsymbol{V}_m \exp(\boldsymbol{H}_m) \boldsymbol{e}_1$$

where $\beta = \|\boldsymbol{p}\|_2$ and $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^T$. The Padé approximation,[25] together with scaling and squaring, is employed to compute $\exp(\boldsymbol{H}_m)$. Other variants with an incomplete orthogonalization in the Arnoldi process could also be attempted,[26] but this was not our focus here.

Krylov-based methods have proved efficient when the matrix is sparse, as is the case in many biological problems. They also have the appeal of being matrix-free by only requiring the matrix-vector product $\sigma^{[4]}_{(t_k,h_k)} \cdot \boldsymbol{v}$ to compute $\exp(\sigma^{[4]}_{(t_k,h_k)}) \cdot \boldsymbol{p}_k$. Our numerical tests use the vanilla Expokit with Krylov order $m = 30$, producing approximations well within the tolerance $\text{tol} = 10^{-5}$ even for problems with large sizes.

5.3. Magnus with an Adaptive SSA-Based State Space

During the integration time of any ODE solver for (2), most of the values in $\boldsymbol{p}(t)$ will be extremely small and therefore computing the full distribution can be expensive without gaining much accuracy. For CME problems with time-independent rates, the FSP-SSA method[27] confines the state space $\boldsymbol{X}$ at each step to only the most probable states at that step. To apply this strategy in our context, we proceed as follows: knowing the current $\boldsymbol{p}_k \approx \boldsymbol{p}(t_k)$ and having chosen a step-size h_k, we wish to advance to the next approximation $\boldsymbol{p}_{k+1} \approx \boldsymbol{p}(t_k + h_k)$ using (6) with $\sigma^{[4]}_{(t_k,h_k)}$ restricted to only the subset of states that the system is likely to occupy during the time interval $[t_k, t_k + h_k]$. This is implemented by first dropping current states with low probabilities in $\boldsymbol{p}_k$, then sampling SSA trajectories from the remaining states and updating it to include all states that the SSA paths travel through. The 'roughness' in the paths is then smoothed out by the r-step reachability,[7] which seeks all states that can be connected to the state space by r reactions or less, and expands the state space to include those. Below is the corresponding pseudocode.

```
1. $\boldsymbol{X}$ is reduced to states with probability $>10^{-16}$
2. $\boldsymbol{X}$ is expanded by SSA runs over $[t_k, t_k + h_k]$ and $r$-step reachability with $r = 5$
3. $\boldsymbol{A}(t)$ is updated to only states in $\boldsymbol{X}$
4. Compute $\boldsymbol{p}_{k+1} = \exp(\sigma^{[4]}_{(t_k,h_k)}) \cdot \boldsymbol{p}_k$.
```

The implementation uses a hash table to keep track of the state space and also note that $\boldsymbol{A}(t)$ is in functional form for the matrix-free Krylov stage, with the action of the matrix exponential in the last step done by using Expokit with the matrix-vector operator $\sigma^{[4]}_{(t_k,h_k)} \cdot \boldsymbol{v}$ defined through:[24]

$$\boldsymbol{A}_1 = \boldsymbol{A}\left(t_k + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) h_k\right),$$

$$\boldsymbol{A}_2 = \boldsymbol{A}\left(t_k + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right) h_k\right),$$

$$\boldsymbol{w}_1 = \boldsymbol{A}_1 \boldsymbol{v},$$

$$\boldsymbol{w}_2 = \boldsymbol{A}_2 \boldsymbol{v},$$

$$\boldsymbol{w}_3 = \boldsymbol{A}_2 \boldsymbol{w}_1,$$

$$\boldsymbol{w}_4 = \boldsymbol{A}_1 \boldsymbol{w}_2,$$

$$\sigma^{[4]}_{(t_k,h_k)} \cdot \boldsymbol{v} = \frac{h_k}{2}(\boldsymbol{w}_1 + \boldsymbol{w}_2) + \frac{h_k^2\sqrt{3}}{12}(\boldsymbol{w}_3 - \boldsymbol{w}_4)$$

It is important to mention again that SSA is considered inexact for the CME with time-dependent rates, because reaction rates are kept constant during each time step. However, the SSA is used only to expand the state space. The probability distribution is computed using the Magnus method instead, therefore the results are not compromised. In the cases where the reaction rates change dramatically, the state space can be expanded using the FRM. This will be more time-consuming, but the state spaces will not be distorted.

The adaptive *MAGNUS-SSA* method requires a scheme for computing the step-size h_k. This will be completed in the next section.

6. ADAPTIVE TIME-STEPPING SCHEMES

As can be seen from the previous section, the error in *MAGNUS-SSA* comes from a combination of four different error sources:

- The FSP error: from truncating the state space
- The Magnus truncation error: from truncating the infinite Magnus series
- The quadrature error: from approximating the integrals involved with quadrature rules
- The Krylov error: from approximating $\boldsymbol{p}_{k+1} = \exp(\sigma) \cdot \boldsymbol{p}_k$ using Krylov subspace techniques.

The FSP error exists for all ODE solvers and *MAGNUS-SSA*. However, in our numerical tests, for the ODE solvers, a sufficiently big state space was truncated at the beginning of each algorithm according to a large number of SSA trajectories, and is fixed during the entire integration. The FSP error for these schemes is therefore minimal. In *MAGNUS-SSA*, the state space is changed at each time step, but the large number of SSA trajectories and r-step reachability with large r required to build the state space assure that the FSP error is also insignificant. Therefore we assume that the FSP error is negligible for all algorithms.

The Krylov error, on the other hand, is automatically managed by Expokit,[28] which is the most extensive software for computing the matrix exponential and has been reported to be well suited for large sparse matrices.[25] Expokit allows an error tolerance, so we can also leave out the Krylov error for simplicity.

The dominant error of the Magnus-based methods, therefore, comes from truncating and interpolating the Magnus series. We seek to approximate this error and use it in an adaptive time-stepping Magnus-based scheme.

6.1. Adaptive Time-Stepping Scheme for Magnus-SSA

At any time interval $[t_k, t_k + h_k]$, the local error is defined to be

$$\text{error}(t_{k+1}) = \|\boldsymbol{p}_{k+1} - \bar{\boldsymbol{p}}_{k+1}\|_1$$

where

$$\boldsymbol{p}_{k+1} = \exp(\sigma^{[4]}_{(t_k,h_k)}) \cdot \boldsymbol{p}_k$$

is the *MAGNUS-SSA* approximation to the exact solution

$$\bar{\boldsymbol{p}}_{k+1} = \exp(\Omega_{(t_k,h_k)}) \cdot \boldsymbol{p}_k$$

The following pseudocode is a template for the overall step-by-step integration process using a traditional step-size control.

Algorithm (MAGNUS-SSA).

```
1: Initialize state space and $\boldsymbol{p}_0$ and time $t_0 = 0$
2: Initialize step-size $h_0 = 0.5$ and step $k = 0$
3: While $t_k < t_f$ do
4:     Update state space and get $\boldsymbol{p}_{k+1} = \exp(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)})\boldsymbol{p}_k$
           as discussed in the previous section
5:     Compute the error estimate error$(t_{k+1})$
6:     if error$(t_{k+1}) > \epsilon$ then
7:         $h_k \leftarrow 0.5 \cdot h_k$
8:         go to Step 3
9:     end if
10:    $t_{k+1} \leftarrow t_k + h_k$
11:    $h_{k+1} \leftarrow \delta\left(\frac{\epsilon}{\text{error}}\right)^{1/p} h_k$
12:    $k \leftarrow k+1$
13: end While
```

The safety factor δ is 0.5 in our code. The error tolerance is $\epsilon = 10^{-5}$, and Expokit is implemented with the same tolerance. The parameter p in Step 11 is set as the order of the error estimator, to be discussed next. The discussed variants use the same Magnus approximation in Step 3 and only differ in how they perform the error estimation in Step 5 of the template. We will see later that Step 3 is moved to be after the if-block in the case of *MAGNUS-SSA*-4 because it has an *a priori* error control.

6.2. Error Approximation

A traditional error estimating technique is to compute the results of one ODE solver with two different orders, and compute the difference. MacNamara and Burrage[24] developed a local error estimate based on this embedding scheme:

$$\begin{aligned}\text{error}(t_{k+1}) &\approx \|\exp(\sigma^{[4]}_{(t_k,h_k)}) \cdot \boldsymbol{p}_k - \exp(\sigma^{[2]}_{(t_k,h_k)}) \cdot \boldsymbol{p}_k\|_1 \\ &= \|\boldsymbol{p}_{k+1} - \exp(\sigma^{[2]}_{(t_k,h_k)}) \cdot \boldsymbol{p}_k\|_1\end{aligned}$$

where $\exp(\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}) \cdot \boldsymbol{p}_k$ is the 2nd-order Magnus with Gauss-Legendre quadrature rule, computed with Expokit where the matrix-vector product operator $\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)} \cdot \boldsymbol{v}$ defined as

$$\boldsymbol{A}_1 = \boldsymbol{A}\left(t_k + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) h_k\right),$$

$$\boldsymbol{A}_2 = \boldsymbol{A}\left(t_k + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right) h_k\right),$$

$$\boldsymbol{w}_1 = \boldsymbol{A}_1 \boldsymbol{v},$$

$$\boldsymbol{w}_2 = \boldsymbol{A}_2 \boldsymbol{v},$$

$$\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)} \cdot \boldsymbol{v} = \frac{h_k}{2}(\boldsymbol{w}_1 + \boldsymbol{w}_2)$$

Since this error estimate is based on the 2nd-order Magnus approximation, it is of order $p = 2$. The algorithm will proceed with the 4th-order Magnus approximation instead. This Magnus implementation is denoted *MAGNUS-SSA*-1 in our numerical tests.

At each time step, *MAGNUS-SSA*-1 requires two Expokit runs, one for the 2nd-order solution and one for the 4th-order solution, where only the latter is required to proceed. This can be time-consuming, therefore we propose a new cheaper local error estimate that removes the need to explicitly compute the 2nd-order approximation. The starting point is the observation that the inverse of a matrix exponential is

$$[\exp(\boldsymbol{\sigma})]^{-1} = \exp(-\sigma)$$

which allows us to rewrite the terms in error(t_{k+1}) as

$$\begin{aligned}&[\exp(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}) - \exp(\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)})] \cdot \boldsymbol{p}_k \\ &\quad = [\boldsymbol{I} - \exp(\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)})\exp(-\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)})] \cdot \boldsymbol{p}_{k+1} \qquad (7)\end{aligned}$$

From this, we use the fact that the product of two matrix exponentials can be approximated using the Baker-Campbell-Hausdorff formula, and the important fact that the matrix-vector product for the 2nd-order Magnus algorithm is embedded in that for the 4th-order Magnus algorithm. This reduces (7) into

$$\begin{aligned}&\left[\boldsymbol{I} - \exp(\sigma^{[2]}_{(t_k,h_k)} - \sigma^{[4]}_{(t_k,h_k)} - \frac{1}{2}[\sigma^{[2]}_{(t_k,h_k)}, \sigma^{[4]}_{(t_k,h_k)}] + \mathcal{O}(h_k^4))\right] \\ &\cdot \boldsymbol{p}_{k+1} \\ &\quad = \left[-\sigma^{[2]}_{(t_k,h_k)} + \sigma^{[4]}_{(t_k,h_k)} + \frac{1}{2}[\sigma^{[2]}_{(t_k,h_k)}, \sigma^{[4]}_{(t_k,h_k)}] + \mathcal{O}(h_k^4))\right] \\ &\quad\cdot \boldsymbol{p}_{k+1} \\ &\quad = \left[\frac{h_k^2\sqrt{3}}{12}[\boldsymbol{A}_2, \boldsymbol{A}_1] + \frac{h_k^3\sqrt{3}}{48}[\boldsymbol{A}_1 + \boldsymbol{A}_2, [\boldsymbol{A}_2, \boldsymbol{A}_1]] \right. \\ &\qquad \left. + \mathcal{O}(h_k^4))\right] \cdot \boldsymbol{p}_{k+1}\end{aligned}$$

where $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are defined as in the *MAGNUS-SSA* algorithm.

We can derive from this a cheap approximation of the local error estimate in *MAGNUS-SSA*-1:

$$\begin{aligned}&\boldsymbol{u}_1 = \boldsymbol{A}_1 \cdot \boldsymbol{p}_{k+1}; \quad \boldsymbol{u}_2 = \boldsymbol{A}_2 \cdot \boldsymbol{p}_{k+1}; \quad \boldsymbol{u}_3 = \boldsymbol{A}_2 \cdot \boldsymbol{u}_1; \\ &\boldsymbol{u}_4 = \boldsymbol{A}_1 \cdot \boldsymbol{u}_2; \quad \boldsymbol{u}_5 = \boldsymbol{A}_1 \cdot \boldsymbol{u}_1; \quad \boldsymbol{u}_6 = \boldsymbol{A}_2 \cdot \boldsymbol{u}_2; \\ &\boldsymbol{u}_7 = \boldsymbol{A}_1 \cdot \boldsymbol{u}_3; \quad \boldsymbol{u}_8 = \boldsymbol{A}_2 \cdot \boldsymbol{u}_4; \quad \boldsymbol{u}_9 = \boldsymbol{A}_2 \cdot \boldsymbol{u}_3; \\ &\boldsymbol{u}_{10} = \boldsymbol{A}_1 \cdot \boldsymbol{u}_4; \quad \boldsymbol{u}_{11} = \boldsymbol{A}_1 \cdot \boldsymbol{u}_6; \quad \boldsymbol{u}_{12} = \boldsymbol{A}_2 \cdot \boldsymbol{u}_5;\end{aligned}$$

$$\text{error}(t_{k+1}) = \left\| \frac{h_k^2\sqrt{3}}{12}(\boldsymbol{u}_3 - \boldsymbol{u}_4) + \frac{h_k^3\sqrt{3}}{48}(2\boldsymbol{u}_7 - 2\boldsymbol{u}_8 + \boldsymbol{u}_9 - \boldsymbol{u}_{10} + \boldsymbol{u}_{11} - \boldsymbol{u}_{12}) \right\|_1$$

The Magnus implementation using this error estimate is denoted *MAGNUS-SSA*-2 in the numerical tests. Because it is based on *MAGNUS-SSA*-1, the error is also of order $p = 2$.

Another well-known error approximating approach is computing the difference between the result of the ODE solver with the result of the same solver with halved time-step. Using this technique, the local error is then:

$$\begin{aligned}\text{error}(t_{k+1}) &\approx \|\exp(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)})\cdot \boldsymbol{p}_k - \exp(\boldsymbol{\sigma}^{[4]}_{(t_k+h_k/2,h_k/2)}) \cdot \exp(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k/2)})\cdot \boldsymbol{p}_k\|_1 \\ &= \|\boldsymbol{p}_{k+1} - \exp(\boldsymbol{\sigma}^{[4]}_{(t_k+h_k/2,h_k/2)})\cdot \exp(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k/2)})\cdot \boldsymbol{p}_k\|_1\end{aligned}$$

The Magnus implementation using this error estimate is called *MAGNUS-SSA*-3 in the numerical tests. A disadvantage of this approach is that Expokit has to be run three times for every time step, one for the normal 4th-order Magnus approximation and two for the 'corrector.' On the other hand, the error estimate is of order $p = 4$, therefore the time steps will be less conservative than *MAGNUS-SSA*-1 and *MAGNUS-SSA*-2.

The final approach of approximating the local error in consideration is based on the leading term of the error in truncating the Magnus expansion. Iserles, Marthinsen and Nørsett[12] derived

$$\begin{aligned}&\sigma^{[4]}_{(t_k,h_k)} \\ &= \Omega_{(t_k,h_k)} + \frac{h_k^4}{720} \\ &\quad \times [\boldsymbol{A}(t_k+h_k),[\boldsymbol{A}(t_k+h_k),[\boldsymbol{A}(t_k),\boldsymbol{A}(t_k+h_k)]]] + \mathcal{O}(h_k^5)\end{aligned}$$

They used the Baker-Campbell-Hausdorff formula to obtain a local error estimate of order $p = 4$ from this equation:

$$\begin{aligned}&\text{error}(t_{k+1}) \\ &= \left\| \frac{h_k^4}{720}[\boldsymbol{A}(t_k+h_k),[\boldsymbol{A}(t_k+h_k),[\boldsymbol{A}(t_k),\boldsymbol{A}(t_k+h_k)]]]\cdot \boldsymbol{p}_k \right\|_1\end{aligned}$$

Expanding the nested commutators in the equation and collecting like-terms, we get a commutator-free form of this error estimate, used in *MAGNUS-SSA*-4:

$$\begin{aligned}\boldsymbol{A}_0 &= \boldsymbol{A}(t_k), \\ \boldsymbol{A}_3 &= \boldsymbol{A}(t_k+h_k), \\ \boldsymbol{u}_1 &= \boldsymbol{A}_3\cdot \boldsymbol{A}_3 \cdot \boldsymbol{A}_0 \cdot \boldsymbol{A}_3 \cdot \boldsymbol{p}_k, \\ \boldsymbol{u}_2 &= \boldsymbol{A}_3\cdot \boldsymbol{A}_0 \cdot \boldsymbol{A}_3 \cdot \boldsymbol{A}_3 \cdot \boldsymbol{p}_k, \\ \boldsymbol{u}_3 &= \boldsymbol{A}_3\cdot \boldsymbol{A}_3 \cdot \boldsymbol{A}_3 \cdot \boldsymbol{A}_0 \cdot \boldsymbol{p}_k, \\ \boldsymbol{u}_4 &= \boldsymbol{A}_0\cdot \boldsymbol{A}_3 \cdot \boldsymbol{A}_3 \cdot \boldsymbol{A}_3 \cdot \boldsymbol{p}_k, \\ \text{error}(t_{k+1}) &= \left\| \frac{h_k^4}{720}(3\boldsymbol{u}_1 - 3\boldsymbol{u}_2 - \boldsymbol{u}_3 + \boldsymbol{u}_4) \right\|_1\end{aligned}$$

A great advantage of *MAGNUS-SSA*-4 over the other Magnus variants in our comparison is that the error estimate is *a priori*. Its implementation moves Step 3 to be after the if-block in the template given earlier so that for each time step, it calculates the error before committing to proceed, and therefore does not run Expokit unnecessarily. There is also only one Expokit run per time step, keeping the execution time minimal.

As implemented here, this error estimate has the shortcoming that it takes into account only the Magnus truncation error and not the quadrature error, unlike the other Magnus variants. The quadrature error is notoriously difficult to estimate,[29,30] and it depends on the behavior of $\boldsymbol{A}(t)$.

7. NUMERICAL TESTS

All numerical tests were done using resources of the Alabama Supercomputer, which houses two supercomputers called SGI UV and DMC. The user can request a job to be executed on either of them, or can simply let the operating system select the more suitable system depending on the workload and availability. All codes were written in FORTRAN 77 and were run on the large queue of the SGI UV with 1 processor core (Xeon E5-4640 CPU operating at 2.4 GHz), 360 hr time limit and 120 GB memory limit.

The models in our numerical tests arise from different fields of biology. In each numerical test, the distributions from solving (2) by the ODE solvers are compared with the frequencies from 10,000 FRM trajectories, computed by (3). The fixed FSP state space for the ODE solvers is found by finding the maximum and minimum of each species count during the 10,000 FRM trajectories, except for *MAGNUS-SSA*, which does not require *a priori* fixed FSP bounds and changes the state space adaptively instead. These FSP bounds are reported for each numerical test.

The error of each ODE solver is defined to be the maximum of 1-norm differences between the marginal distributions from that ODE solver and the FRM.

7.1. Test 1—The Model of Two Competing T Cell Clonotypes

The first problem in our comparison models the competition between T cell clonotypes.[24,31] We consider two species:

$$V_1: \text{T cell clonotype 1,}$$
$$V_2: \text{T cell clonotype 2}$$

Table I. Results from the ODE solvers for the model of two competing T cell clonotypes (Test 1). The error is computed as the maximum 1-norm difference between the marginal distributions from each ODE solver and the FRM.

ODE solver	Time cost and note	Error
Adams-PECE	Incorrect solver (flag 4)	N/A
Runge-Kutta	1 s	0.0481
BDF-GM-LU0	1 s	0.1624
BDF-LU0	5 s	1.3152
MAGNUS-SSA-1	10 s	0.0478
MAGNUS-SSA-2	8 s	0.0476
MAGNUS-SSA-3	4 s	0.0479
MAGNUS-SSA-4	3 s	0.0491

which can interact through a multivariate birth-death process:

$$\varnothing \xrightarrow{c_1} V_1,$$
$$V_1 \xrightarrow{c_2} \varnothing,$$
$$\varnothing \xrightarrow{c_3} V_2,$$
$$V_2 \xrightarrow{c_4} \varnothing$$

The reaction rates are:[24]

$$c_1 = \phi(t)\cdot\left(\frac{0.5[V_1]}{[V_1]+[V_2]} + \frac{0.5[V_1]}{[V_1]+1000}\right)\ (\text{year}^{-1}),$$
$$c_2 = 1\ (\text{year}^{-1}),$$
$$c_3 = \phi(t)\cdot\left(\frac{0.5[V_2]}{[V_1]+[V_2]} + \frac{0.5[V_2]}{[V_2]+1000}\right)\ (\text{year}^{-1}),$$
$$c_4 = 1\ (\text{year}^{-1})$$

where $\phi(t) = 60/(1+(t/15)^5)$ models the decreasing stimulation that the clonotypes receive.

We start from the initial values

$$V_1 = 10$$
$$V_2 = 10$$

until the end time $t_f = 5$ (years). The FRM trajectories during this time interval result in the bounds for the FSP state space:

$$0 \le V_1 \le 57,$$
$$0 \le V_2 \le 51$$

The FSP state space contains $n = 3016$ states and the CME matrix contains $nz = 14860$ nonzero elements. The norm of the CME matrix at $t = 0$ is 513603.

The results from the ODE solvers are summarized in Table I. Figure 1 shows the probability distributions from these ODE solvers, and Figure 2 shows the local error estimates from all *MAGNUS-SSA* algorithms. Figure 1 shows that the solutions from *Runge-Kutta* and the *MAGNUS-SSA* implementations agree with the FRM frequencies. *Adams-PECE* did not finish, while the solutions of both *BDF-GM-LU*0 and *BDF-LU*0 fail to follow the shape of the correct distributions, resulting in noticeable errors, as seen in Table I. Figure 2 shows that the error estimates from all *MAGNUS-SSA* variants are very similar.

7.2. Test 2—The Epidemic Model with Periodic Contact Rate

The second problem in our comparison is an epidemic model,[32–35] consisting of three species:

$$S: \text{susceptible population},$$
$$I: \text{infected population},$$
$$R: \text{recovered population}$$

These populations can interact in six different reactions:

$$S + I \xrightarrow{c_1} 2I,$$
$$I \xrightarrow{c_2} R,$$
$$R \xrightarrow{c_3} S,$$
$$S \xrightarrow{c_4} \varnothing,$$
$$I \xrightarrow{c_5} \varnothing,$$
$$R \xrightarrow{c_6} \varnothing$$

with reaction rates

$$c_1 = 0.003 f(t)\ (\text{day}^{-1}),$$
$$c_2 = 0.02\ (\text{day}^{-1}),$$
$$c_3 = 0.007\ (\text{day}^{-1}),$$
$$c_4 = 0.002\ (\text{day}^{-1}),$$

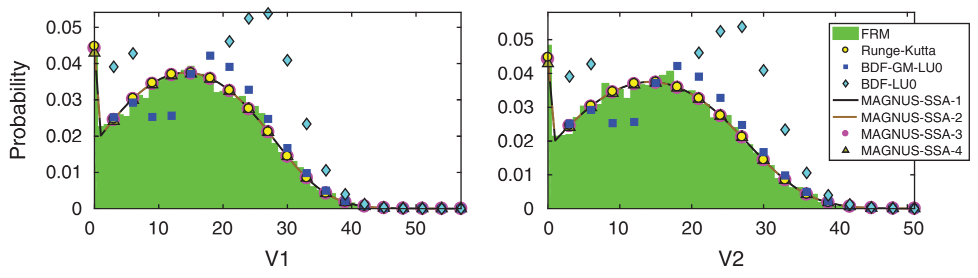

Fig. 1. Probability distributions from the model of two competing T cell clonotypes (Test 1).

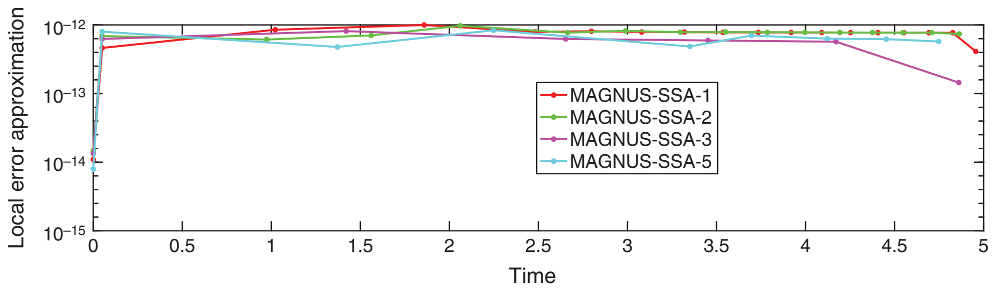

Fig. 2. Error estimates in the *MAGNUS-SSA* variants from the model of two competing T cell clonotypes (Test 1).

$$c_5 = 0.05\ (\text{day}^{-1}),$$
$$c_6 = 0.002\ (\text{day}^{-1})$$

where $f(t) = (1 + 0.6 \sin(2\pi \cdot t/6))$ describes the periodic contact rate between the infected population and the susceptible population.

The initial values for the populations are:

$$S = 200,$$
$$I = 10,$$
$$R = 0$$

and the end time is $t_f = 10$ (years). The bounds for the FSP state space are:

$$0 \le S \le 200,$$
$$0 \le I \le 17,$$
$$0 \le R \le 35$$

There are $n = 1237356$ states in the FSP state space, and $nz = 8518611$ nonzero elements in the CME matrix. The norm of the CME matrix is 78286181 at $t = 0$.

Table II summarizes the results from the ODE solvers. Figures 3 and 4 show the probability distributions from the ODE solvers and the error estimates in the *MAGNUS-SSA* variants. Because the solutions from *BDF-GM-LU*0 and *BDF-LU*0 are very far from the correct probability distributions and even contain negative values (see Table II),

Table II. Results from the ODE solvers for the epidemic model with periodic contact rate (Test 2). The error is computed as the maximum 1-norm difference between the marginal distributions from each ODE solver and the FRM.

ODE solver	Time cost and note	Error
Adams-PECE	243 s	0.0550
Runge-Kutta	172 s	0.0550
BDF-GM-LU0	189 s	136.2503
BDF-LU0	17 s	71.4624
MAGNUS-SSA-1	3924 s	0.0550
MAGNUS-SSA-2	2211 s	0.0550
MAGNUS-SSA-3	2165 s	0.0572
MAGNUS-SSA-4	509 s	0.0552

their approximations are not included in Figure 3. The solutions from *Adams-PECE*, *Runge-Kutta* and all *MAGNUS-SSA* variants are shown in agreement with the FRM frequencies. Figure 4 shows that the error estimates are periodic, with slight disturbances when $f(t)$ reaches maximum or minimum.

7.3. Test 3—The Transcriptional Regulatory Model

The final biological problem for comparing the ODE solvers depicts a transcriptional regulatory system.(36) The problem consists of six species:

M:	protein (monomer),
D:	transcription factor (dimer),
DNA:	DNA template, free of dimers,
DNA.D:	DNA template, bound at one binding site,
DNA.2D:	DNA template, bound at both binding sites,
RNA:	mRNA produced by transcription

which can interact through ten reactions:

$$\text{RNA} \xrightarrow{c_1} \text{RNA} + \text{M},$$
$$\text{M} \xrightarrow{c_2} \varnothing,$$
$$\text{DNA.D} \xrightarrow{c_3} \text{RNA} + \text{DNA.D},$$
$$\text{RNA} \xrightarrow{c_4} \varnothing,$$
$$\text{DNA} + \text{D} \xrightarrow{c_5} \text{DNA.D},$$
$$\text{DNA.D} \xrightarrow{c_6} \text{DNA} + \text{D},$$
$$\text{DNA.D} + \text{D} \xrightarrow{c_7} \text{DNA.2D},$$
$$\text{DNA.2D} \xrightarrow{c_8} \text{DNA.D} + \text{D},$$
$$\text{M} + \text{M} \xrightarrow{c_9} \text{D},$$
$$\text{D} \xrightarrow{c_{10}} \text{M} + \text{M}$$

The reaction rates are:

$$c_1 = 0.043\ (\text{s}^{-1}),$$
$$c_2 = 0.0007\ (\text{s}^{-1}),$$
$$c_3 = 0.078\ (\text{s}^{-1}),$$

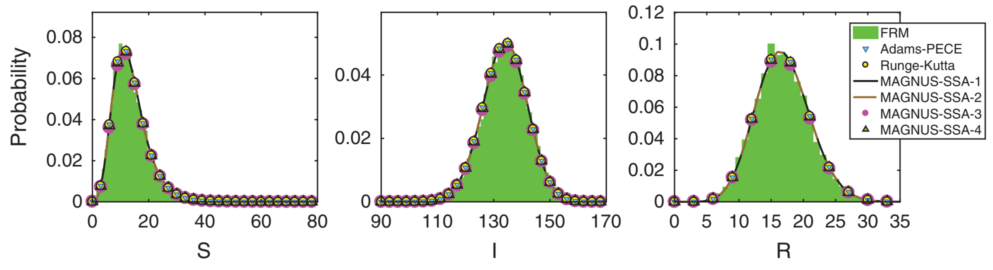

Fig. 3. Probability distributions from the epidemic model with periodic contact rate (Test 2).

$$c_4 = 0.0039\ (\mathrm{s}^{-1}),$$
$$c_5 = \frac{0.012 \cdot 10^9}{A \cdot V(t)}\ (\mathrm{s}^{-1}),$$
$$c_6 = 0.4791\ (\mathrm{s}^{-1}),$$
$$c_7 = \frac{0.00012 \cdot 10^9}{A \cdot V(t)}\ (\mathrm{s}^{-1}),$$
$$c_8 = 0.8765 \cdot 10^{-11}\ (\mathrm{s}^{-1}),$$
$$c_9 = \frac{0.05 \cdot 10^9}{A \cdot V(t)}\ (\mathrm{s}^{-1}),$$
$$c_{10} = 0.5\ (\mathrm{s}^{-1})$$

where A is the Avogado's constant, and $V(t)$ is the cell volume at time t, which increases from the initial value $V(0) = 10^{-15}$ in accordance to

$$V(t) = V(0)e^{\ln(2)t/\tau}$$

during the entire cell cycle time period $\tau = 35$ minutes until the cell divides.

We wish to follow the distributions of the count of each species from the initial state where the cell has 2 monomers, 6 dimers, and two unbound DNAs:

$$\mathrm{M} = 2, \quad \mathrm{D} = 6,$$
$$\mathrm{DNA} = 2, \quad \mathrm{DNA.D} = 0,$$
$$\mathrm{DNA.2D} = 0, \quad \mathrm{RNA} = 0$$

until the final time $t_f = 300$ (s).

The FSP bounds are

$$0 \le \mathrm{M} \le 50, \quad 0 \le \mathrm{D} \le 81,$$
$$0 \le \mathrm{DNA} \le 2, \quad 0 \le \mathrm{DNA.D} \le 2,$$
$$0 \le \mathrm{DNA.2D} \le 2, \quad 0 \le \mathrm{RNA} \le 24$$

There are $n = 2822850$ states in the FSP state space, and the CME matrix contains $nz = 24093072$ nonzero elements. The norm of the CME matrix at $t = 0$ is 311973491. Table III summarizes the results from the ODE solvers. The probability distributions from the ODE solvers and the error estimates are plotted in Figures 5 and 6. Similarly to Test 2, *BDF-GM-LU*0 and *BDF-LU*0 produce inaccurate results (see Table III). Furthermore, *Adams-PECE* detected that the problem is stiff and therefore did not finish. Their solutions are therefore not included in Figure 5. *Runge-Kutta* and all *MAGNUS-SSA* variants finished and their solutions followed the FRM frequencies faithfully. Figure 6 shows that the *MAGNUS-SSA* error estimates are stable, with clear agreement between the 2nd-order and the 4th-order approximations, respectively.

7.4. Observations

The three biological problems in the numerical tests were chosen because they exhibit distinct features. The model of two competing T cell clonotypes is symmetrical. Over time, the probability mass concentrates on states with low V_1 counts and high V_2 counts, or those with high V_1 counts and low V_2 counts, because the model is also bimodal. The epidemic model with periodic contact rate, on the other hand, is monomodal with periodic time-dependency.

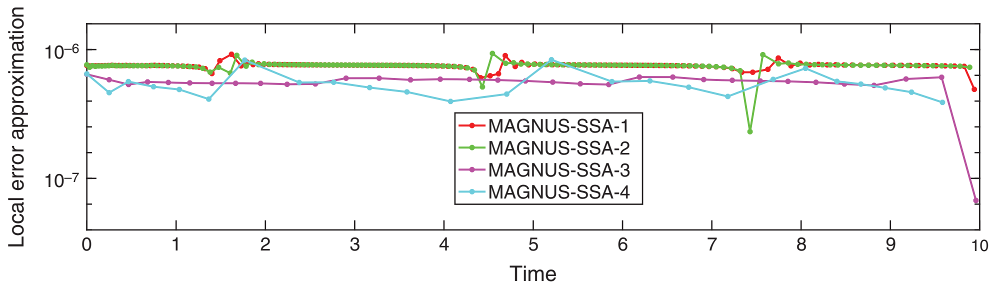

Fig. 4. Error estimates in the *MAGNUS-SSA* variants from the epidemic model with periodic contact rate (Test 2).

Table III. Results from the ODE solvers for the transcriptional regulatory model (Test 3). The error is computed as the maximum 1-norm difference between the marginal distributions from each ODE solver and the FRM.

ODE solver	Time cost and note	Error
Adams-PECE	Incorrect solver (flag 5)	N/A
Runge-Kutta	82415 s	0.0465
BDF-GM-LU0	1203 s	78.2631
BDF-LU0	88 s	273.3214
MAGNUS-SSA-1	137304 s	0.0465
MAGNUS-SSA-2	52764 s	0.0465
MAGNUS-SSA-3	78505 s	0.0465
MAGNUS-SSA-4	49139 s	0.0465

Finally, the transcriptional regulatory model is monomodal and very stiff, evidenced in the differences in magnitude of the reaction rates and the big norm of the CME matrix.

The initial conditions for the problems in the three numerical tests were chosen so that they are biologically relevant. Specifically, the initial state in Test 3 is from the original problem.[36] The initial condition for Test 1 follows the same structure as in Ref. [24], and the initial values for Test 2 is the same as in Ref. [32], except the lower value for S, which is dictated by storage requirements. We also performed tests for the same biological problems with slightly different initial conditions, and found the results to be qualitatively similar to those presented here. This means that the algorithms are sensitive to a similar extent to the initial conditions.

*BDF-GM-LU*0 and *BDF-LU*0 finished in all three numerical tests. They escape the storage requirement of the complete LU decomposition by applying a fast and cheap incomplete LU decomposition instead where, in the course of the decomposition, the elements not in the sparsity pattern of $\mathbf{A}(t)$ are discarded. The price for the low storage requirement and cheap computation time cost is that the incomplete LU decomposition might lose valuable information during the integration time, evidenced by the large error of the marginal distributions. Because of this, the probability distributions computed by *BDF-GM-LU*0 and *BDF-LU*0 are wrong in all three tests. Tables I–III confirm that their results always have the largest errors. For instance, in Test 1, which is the smallest problem, they fail to follow the shape of the marginal probability distributions. This is in contrast to results in Ref. [37], in which *BDF-GM-LU*0 and *BDF-LU*0 performed better than their complete LU variants across a range of ODE problems. This is because for those problems, the matrices are sparse with small norms, so the elements discarded by the incomplete LU decomposition are indeed insignificant. Performing the complete LU decomposition in that case would be time-consuming without substantially improved accuracy.

Adams-PECE did not finish in Test 1, because the maximum number of steps allowed in the program was exceeded before reaching t_f. In Test 2, it produced correct distributions in competitive time, only surpassed by *Runge-Kutta* (*RK*). In Test 3 *Adams-PECE* did not finish because it detected that the problem is stiff.

Quite surprisingly, *RK* was the only solver besides the *MAGNUS-SSA* algorithms to complete for all three numerical tests. This is interesting, given that among the traditional ODE solvers in this comparison, it is the only explicit algorithm. Even more, *RK* finished first in Tests 1 and 2, because its explicit formula requires less computation time. *RK* produced the correct probability distribution in Test 3, while the other traditional ODE solvers failed because the problem is stiff. Note that in our previous report,[38] all ODE solvers were tested for the transcriptional regulatory model with the same set-up as Test 3, but with initial values

$$\mathrm{M}=0, \quad \mathrm{D}=2,$$
$$\mathrm{DNA}=1, \quad \mathrm{DNA.D}=0,$$
$$\mathrm{DNA.2D}=0, \quad \mathrm{RNA}=0$$

final time $t_f = 600$ s, FSP bounds

$$0 \le \mathrm{M} \le 46, \quad 0 \le \mathrm{D} \le 59,$$
$$0 \le \mathrm{DNA} \le 1, \quad 0 \le \mathrm{DNA.D} \le 1,$$
$$0 \le \mathrm{DNA.2D} \le 1, \quad 0 \le \mathrm{RNA} \le 12$$

and $n = 293280$, $nz = 2091993$. *RK* did not finish in that test, possibly because of the longer end time.

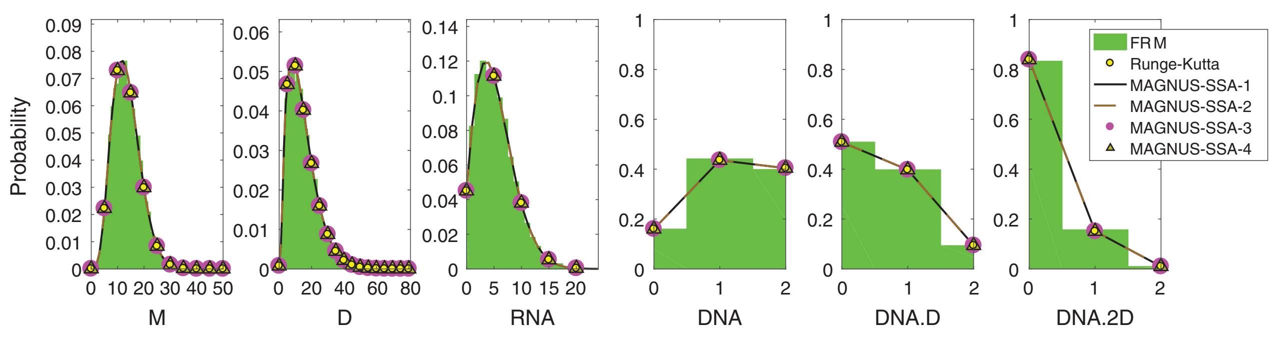

Fig. 5. Probability distributions from the transcriptional regulatory model (Test 3).

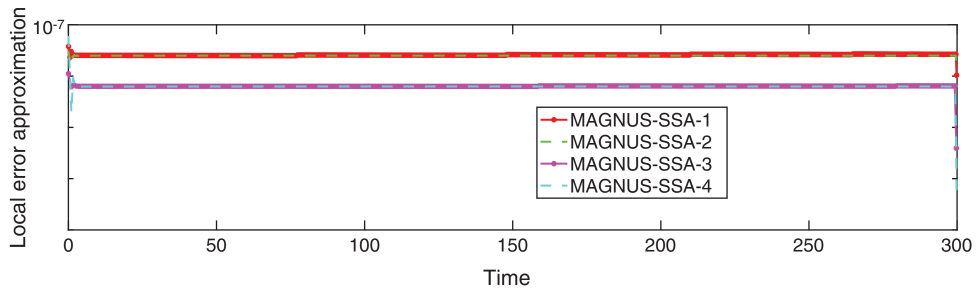

Fig. 6. Error estimates in the *MAGNUS-SSA* variants from the transcriptional regulatory model (Test 3).

All *MAGNUS-SSA* algorithms produced exact probability distributions in all numerical tests, proving that Magnus-based integration methods are suitable for solving CME problems with time-dependent rates. On the other hand, *MAGNUS-SSA* is time-consuming, because there is at least one exponential-matrix-vector product to be computed every time step. Therefore all *MAGNUS-SSA* implementations were exceeded by *RK* in Test 1, and *RK* and *Adams-PECE* in Test 2. However, all *MAGNUS-SSA* implementations except for *MAGNUS-SSA*-4 outpaced *RK* in Test 3, because the Magnus-based methods have been known to excel in solving highly oscillatory or stiff problems.(13)

We now compare the different *MAGNUS-SSA* algorithms. They all produced good results in the tests, and it is not surprising that *MAGNUS-SSA*-1 always finished last. It requires two Expokit runs per time step, which is time-consuming, and the error estimate is of order 2, implying the time steps are taken conservatively. *MAGNUS-SSA*-2 employs our cheap approximation of the local error estimate used in *MAGNUS-SSA*-1, and it is faster in all numerical tests, even more than halving the time cost in Test 3. Figures 2, 4 and 6 reveal that the error estimates in *MAGNUS-SSA*-2 follow closely those in *MAGNUS-SSA*-1, explaining why its solutions are very dependable.

MAGNUS-SSA-3 applies the common halved time-step scheme. Because both the normal Magnus solution and its 'corrector' are of the 4th order, so is its error. This, in combination with the fact that its error estimates tend to be lower than those in *MAGNUS-SSA*-1 and *MAGNUS-SSA*-2, result in more relaxed time steps and ultimately small time costs, even though there are three Expokit runs for every time step.

MAGNUS-SSA-4 is the fastest among the Magnus implementations in all tests, and there are several reasons for this. Firstly, and most importantly, its error estimate is *a priori*, allowing the algorithm to check the time step before committing to the time-demanding task of finding the next probability distribution. Moreover, there is only one Expokit run for each time step. It is possible that for the problems where $\boldsymbol{A}(t)$ changes dramatically, *MAGNUS-SSA*-4 might fail, because its error estimate does not take the quadrature error into account. However, many biological problems with time-dependent rates are modeled using continuous functions for the reaction rates, in which case *MAGNUS-SSA*-4 is a good candidate as an ODE solver.

8. CONCLUSIONS

In this paper, we developed the framework of Magnus-SSA, which allows the FSP state space to be adaptively changed during the Magnus integration time. This decreases the time costs of the Magnus-based methods. We also implemented two local error approximating schemes from MacNamara and Burrage,(24) and Iserles, Marthinsen and Nørsett,(12) as well as two other error estimation techniques. The resulting Magnus-SSA algorithms were then tested against Adams-PECE, Runge-Kutta, and BDF, across several problems in biology where the CME arises. The numerical results show that Magnus-based methods are serious candidates for solving stiff CME problems with time-dependent rates.

It is important to point out that Adams-PECE, Runge-Kutta, and BDF are 'all-purpose' ODE solvers, where the Magnus-based methods in consideration here are only suitable for solving the linear ODE in the form of (2). However, such problems arise in many biological fields and even in other sciences where Markovian reaction networks occur.(1) Numerical Magnus-based methods, therefore, can be an important choice for solving these problems. On the other hand, there is already interest in expanding the Magnus method to solving non-autonomous linear ODE problems.(14) This may be of interest to the biology community, where such problems can emerge.(39)

Acknowledgments: The work was supported by the NSF grant DMS-1320849. Resources of the Alabama Supercomputer Authority were used to conduct the tests.

References and Notes

1. J. Goutsias and G. Jenkinson; Markovian dynamics on complex reaction networks; *Physics Reports* 529, 199 (**2013**).
2. D. Gillespie; Exact stochastic simulation of coupled chemical reactions; *J. Phys. Chem.* 81, 2340 (**1977**).
3. D. Gillespie; A general method for numerically simulating the stochastic time evolution of coupled chemical reactions; *Journal of Computational Physics* 22, 403 (**1976**).

4. R. Purtan and A. Udrea; A modified stochastic simulation algorithm for time-dependent intensity rates; *19th International Conference on Control Systems and Computer Science* (**2013**).
5. Y. Niu, K. Burrage, and L. Chen; Modelling biochemical reaction systems by stochastic differential equations with reflection; *Journal of Theoretical Biology* 396, 90 (**2016**).
6. K. Burrage, P. Burrage, S. Leier, and T. Marquez-Lago; Stochastic processes, multiscale modeling, and numerical methods for computational cellular biology, A Review of Stochastic and Delay Simulation Approaches in Both Time and Space in Computational Cell Biology, Springer (**2017**).
7. B. Munsky and M. Khammash; The finite state projection algorithm for the solution of the chemical master equation; *J. Chem. Phys.* 124, 044104 (**2006**).
8. K. N. Dinh and R. B. Sidje; Understanding the finite state projection and related methods for solving the chemical master equation; *Physical Biology* 13 (**2016**).
9. W. Magnus; On the exponential solution of differential equations for a linear operator; *Communications on Pure and Applied Mathematics* 7, 649 (**1954**).
10. A. Iserles, S. P. Nørsett, and A. F. Rasmussen; Time symmetry and high-order Magnus methods; *Applied Numerical Mathematics* 39, 379 (**2001**).
11. S. Blanes, F. Casas, J. A. Oteo, and J. Ros; The Magnus expansion and some of its applications, *Physics Reports* 470, 151 (**2009**).
12. A. Iserles, A. Marthinsen, and S. P. Nørsett; On the implementation of the method of Magnus series for linear differential equations; *BIT Numerical Mathematics* 39, 281 (**1999**).
13. N. Aparicio, S. Malham, and M. Oliver; Numerical evaluation of the evans function by Magnus integration; *BIT Numerical Mathematics* 45, 219 (**2005**).
14. P. Bader, S. Blanes, F. Casas, and E. Ponsoda; Efficient numerical integration of Nth-order non-autonomous linear differential equations; *Journal of Computational and Applied Mathematics* 291, 380 (**2016**).
15. D. Gillespie; A rigorous derivation of the chemical master equation; *Physica A: Statistical Mechanics and Its Applications* 188, 404 (**1992**).
16. K. Burrage; Parallel and Sequential Methods for Ordinary Differential Equations, Clarendon Press (**1995**).
17. E. Hairer, S. P. Nørsett, and G. Wanner; Solving Ordinary Differential Equations I, Nonstiff Problems, Springer (**1987**).
18. G. Wanner and E. Hairer, Solving Ordinary Differential Equations II, Stiff and Differential Algebraic Problems, Springer (**1991**).
19. G. Hall and A. Usman; Modified order and stepsize strategies in Adams codes; *Journal of Computational and Applied Mathematics* 11, 113 (**1999**).
20. L. F. Shampine and M. K. Gordon; Computer Solution of Ordinary Differential Equations: The Initial Value Problem, W.H. Freeman and Co. (**1975**).
21. R. W. Brankin, I. Gladwell, and L. F. Shampine; RKSUITE: A suite of Runge-Kutta codes for the initial value problem for ODEs, Softreport 91-1, Technical Report, Math. Dept., Southern Methodist University, Dallas, TX, USA (**1991**).
22. P. N. Brown, G. D. Byrne, and A. C. Hindmarsh; VODE: A variable-coefficient ODE solver; *SIAM Journal on Scientific and Statistical Computing* 10, 1038 (**1989**).
23. Y. Cai, X. Peng, Q. Li, and K. Wang; A numerical solution to the nonlinear point kinetics equations using Magnus expansion; *Annals of Nuclear Energy* 89, 84 (**2016**).
24. S. MacNamara and K. Burrage; Stochastic modeling of naive T cell homeostasis for competing clonotypes via the master equation; *Multiscale Modeling and Simulation* 8, 1325 (**2010**).
25. C. Moler and C. Van Loan; Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later; *SIAM Review* 45, 1 (**2003**).
26. H. Vo and R. B. Sidje; Approximating the large sparse matrix exponential using incomplete orthogonalization and Krylov subspaces of variable dimension. *Numerical Linear Algebra with Applications* 24, e2090 (**2017**).
27. R. B. Sidje and H. Vo; Solving the chemical master equation by a fast adaptive finite state projection based on the stochastic simulation algorithm; *Mathematical Biosciences* 269, 10 (**2015**).
28. R. B. Sidje; Expokit: A software package for computing matrix exponentials; *ACM Transactions on Mathematical Software (TOMS)* 24, 130 (**1998**).
29. R. Cools; Constructing cubature formulae: The science behind the art; *Acta Numerica* 6, 1 (**1997**).
30. P. Davis and P. Rabinowitz; Methods of Numerical Integration, Dover Publications, Inc. (**1984**).
31. E. Stirk, C. Molina-Paris, and H. van den Berg; Stochastic niche structure and diversity maintenance in the T cell repertoire; *Journal of Theoretical Biology* 255, 237 (**2008**).
32. T. Vo and C. Priami; Simulation of biochemical reactions with time-dependent rates by the rejection-based algorithm; *The Journal of Chemical Physics* 143 (**2015**).
33. R. M. Anderson; Population Dynamics of Infectious Diseases: Theory and Applications, Chapman and Hall, London, New York (**1982**).
34. D. J. Daley and J. Gani; Epidemic Modeling: An Introduction, Cambridge University Press (**2005**).
35. N. Bacaër; On the stochastic SIS epidemic model in a periodic environment; *Journal of Mathematical Biology* 71, 491 (**2015**).
36. J. Goutsias; Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems; *The Journal of Chemical Physics* 122, 184102 (**2005**).
37. R. B. Sidje and W. J. Stewart; A numerical study of large sparse matrix exponentials arising in Markov chains; *Computational Statistics and Data Analysis* 29, 345 (**1999**).
38. K. N. Dinh and R. B. Sidje; A comparison of the Magnus expansion and other solvers for the chemical master equation with time-dependent propensities (**2017**).
39. R. Johnson and B. Munsky; The finite state projection approach to analyze dynamics of heterogeneous populations; *Physical Biology* 14 (**2017**).