Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation 207 (2023) 189–208
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Original articles

Numerical solution of kinetic SPDEs via stochastic Magnus expansion☆,☆☆

Kevin Kamm*, Stefano Pagliarani, Andrea Pascucci

Dipartimento di Matematica, Universita di Bologna, Piazza di Porta S. Donato 5, Bologna, 40126, Italy

Received 13 September 2022; received in revised form 28 December 2022; accepted 29 December 2022
Available online 3 January 2023

Abstract

In this paper, we show how the Itô-stochastic Magnus expansion can be used to efficiently solve stochastic partial differential equations (SPDE) with two space variables numerically. To this end, we will first discretize the SPDE in space only by utilizing finite difference methods and vectorize the resulting equation exploiting its sparsity.

As a benchmark, we will apply it to the case of the stochastic Langevin equation with constant coefficients, where an explicit solution is available, and compare the Magnus scheme with the Euler–Maruyama scheme. We will see that the Magnus expansion is superior in terms of both accuracy and especially computational time by using a single GPU and verify it in a variable coefficient case. Notably, we will see speed-ups of order ranging form 20 to 200 compared to the Euler–Maruyama scheme, depending on the accuracy target and the spatial resolution.

Keywords: Magnus expansion; Stochastic Langevin equation; Numerical solutions for SPDE; GPU computing

1. Introduction

In the recent paper [11] we have introduced a stochastic version of the Magnus expansion [13] (hereafter abbreviated in ME), namely an exponential representation for solutions of matrix-valued stochastic differential equations (SDEs), in the Itô sense, of the form

$$dX_t = B_t X_t dt + A_t X_t dW_t, \qquad X_0 = I_d, \tag{1}$$

for non-commuting bounded progressively measurable random matrices A_t and B_t. These results have recently found applications in the study of so-called signature cumulants [7], semi-linear Itô SDEs [19], linear SDEs on matrix Lie groups [14], stochastic modeling of motion in turbulent flows [3], stability of multi-variate geometric Brownian motion [2] and modeling rating transition matrices in quantitative finance [10].

☆ This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 813261 and is part of the ABC-EU-XVA project.

☆☆ All data generated or analyzed during this study are included in this published article. In particular, the code to produce the numerical experiments is available at https://github.com/kevinkamm/MagnusSPDE2D.

* Corresponding author.
E-mail addresses: kevin.kamm@unibo.it (K. Kamm), stefano.pagliarani9@unibo.it (S. Pagliarani), andrea.pascucci@unibo.it (A. Pascucci).

https://doi.org/10.1016/j.matcom.2022.12.029

In this note we investigate the application of the ME to the numerical resolution of a class of hypoelliptic stochastic partial differential equations (SPDEs) that naturally arises in physics and mathematical finance. The deterministic prototype of such SPDEs is the classical Langevin equation

$$\frac{1}{2}\partial_{vv}u_t + v\partial_x u - \partial_t u_t = 0, \tag{2}$$

where the variables $t \geq 0$, $x \in \mathbb{R}$ and $v \in \mathbb{R}$ respectively stand for time, position and velocity, and the unknown $u_t = u_t(x, v)$ stands for the density of a particle in the phase space. Notice that (2) is a degenerate, non-uniformly parabolic PDE. Perturbations of (2) with variable coefficients appear in linear and non-linear form in several applications in kinetic theory (see, for instance, [4,5,12]); also, (2) describes path-dependent financial derivatives such as Asian options and volatility contracts (see, for instance, [6,15]).

We consider here the stochastic version of (2), which is the kinetic SPDE

$$du_t = \left(\frac{a_t}{2}\partial_{vv} + v\partial_x + b_t\partial_v + c_t\right)u_t dt + (\sigma_t\partial_v + \beta_t)\,u_t dW_t, \tag{3}$$

where a_t, b_t, c_t, σ_t and β_t are non-constant (i.e., for example, $a_t = a_t(x, v)$) and possibly random coefficients. Here W denotes a Wiener process defined on a complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$ endowed with a filtration $(\mathcal{F}_t)_{t\geq 0}$ satisfying the usual conditions. SPDE (3) naturally appears in stochastic filtering theory: as shown in [16,17], the fundamental solution of (3) is the conditional transition density of a two-dimensional stochastic process representing the position and the velocity of a particle under partial observation.

The numerical solution of (3) is a challenging issue, as standard techniques, such as Euler methods, can be cumbersome and excessively time-consuming. In this paper, we want to demonstrate how the Itô-stochastic ME can be used as an efficient numerical tool to solve SPDEs meanwhile exploiting modern computer architectures such as GPUs and multiple CPUs. The theoretical groundwork for the Itô-stochastic ME was established in [11] alongside some numerical experiments using a GPU to fully utilize the parallel-in-time and parallel-in-simulation features of the ME. Here, we show how the Magnus expansion can be used iteratively to overcome the constraint of a small convergence radius in time. Indeed, by Kamm et al. [11, Theorem 1], the representation $X_t = e^{Y_t}$ for the unique strong solution to (1) is valid only up to a strictly positive stopping time. More precisely, we have the following:

Theorem 1.1 (*[11]*). *Let A_t and B_t be bounded progressively measurable matrices in $\mathbb{R}^{d\times d}$ and let $(\Omega, \mathcal{F}, \mathbb{P}, (\mathcal{F}_t)_{t\geq 0})$ be a filtered probability space equipped with a standard Brownian motion W. For $T > 0$ let also $X = (X_t)_{t\in[0,T]}$ be the unique strong solution to (1). There exists a strictly positive stopping time $\tau \leq T$ such that:*

1. *X_t has a real logarithm $Y_t \in \mathbb{R}^{d\times d}$ up to time τ, i.e.*

$$X_t = e^{Y_t}, \qquad 0 \leq t < \tau; \tag{4}$$

2. *the following representation holds $\mathbb{P}$-almost surely:*

$$Y_t = \sum_{n=1}^{\infty} Y_t^{(n)}, \qquad 0 \leq t < \tau, \tag{5}$$

   where $Y^{(n)}$ is the nth term in the stochastic ME (see also formulas (7) below, in the case of constant matrices A and B);
3. *there exists a positive constant C, only dependent A, B, T and d, such that*

$$\mathbb{P}(\tau \leq t) \leq Ct, \qquad t \in [0, T]. \tag{6}$$

The first point of Theorem 1.1 tells us that the ME only converges up to a stopping time: to overcome this restriction, the numerical implementation of the ME requires to apply it iteratively in time. Clearly, by (6) the convergence of the ME is problem-dependent, meaning that there is no universal best time step-size for the ME. Point (ii) of Theorem 1.1 actually yields the numerical scheme by truncating the infinite series (5): we will see that in practice it is sufficient to consider only two or three terms to obtain a good degree of accuracy. For this reason, we will not recall the entire general expression for the terms $Y_t^{(n)}$, but rather provide a user guide in Appendix A

on how to easily derive the following expansion formulas, up to order 3, in the case of constant A and B:

$$\begin{aligned} Y_t^1 &= Bt + AW_t, \\ Y_t^2 &= Y_t^1 - \frac{1}{2}A^2 t + [B,A]\int_0^t W_s ds - \frac{1}{2}[B,A]\, tW_t, \\ Y_t^3 &= Y_t^2 + [[B,A],A]\left(\frac{1}{2}\int_0^t W_s^2 ds - \frac{1}{2}W_t\int_0^t W_s ds + \frac{1}{12}tW_t^2\right) \\ &\quad + [[B,A],B]\left(\int_0^t sW_s ds - \frac{1}{2}t\int_0^t W_s ds - \frac{1}{12}t^2 W_t\right), \end{aligned} \tag{7}$$

where $[A,B] := AB - BA$.

The paper is organized as follows. In Section 2, we show how to discretize a parabolic-type SPDE by an SDE of the type (1). In Section 3, we show the numerical experiments in the special case of the stochastic Langevin equation with constant coefficients where the exact solution is available in closed form. First, the implementation and notations for the error-analysis are introduced and afterwards several tests concerning the parameters of the ME are discussed. Then we compare the performance of the iterated ME with that of standard Euler–Maruyama schemes. In Section 3.2, we consider the more general case of SPDE (3) with variable coefficients. In Appendix A, we provide a heuristic derivation of the expansion formulas up to order three.

2. SPDEs with two space-variables

In this section, we show how to derive a numerical scheme for a general parabolic SPDE by combining space-discretization and ME. We will perform some formal computations, which hold even for a fairly general class of SPDEs. Since the computations are understood in a formal manner, we do not impose any further conditions on the coefficients of the following type of SPDE

$$\begin{cases} du_t(x,v) = \Big(h(x,v)u_t(x,v) + f^x(x,v)\partial_x u_t(x,v) + f^v(x,v)\partial_v u_t(x,v) \\ \qquad\qquad + \frac{1}{2}g^{xx}(x,v)\partial_{xx}u_t(x,v) + g^{xv}(x,v)\partial_{xv}u_t(x,v) + \frac{1}{2}g^{vv}(x,v)\partial_{vv}u_t(x,v)\Big)\, dt \\ \qquad\qquad + \big(\sigma(x,v)u_t(x,v) + \sigma^x(x,v)\partial_x u_t(x,v) + \sigma^v(x,v)\partial_v u_t(x,v)\big) dW_t \\ u_0(x,v) \;\; = \phi(x,v). \end{cases} \tag{8}$$

For simplicity, we will only consider the case of time-independent and non-random coefficients but it is straightforward to include them in the formal computations.

In the following subsections, we will demonstrate how to derive an SDE of the form (1) to approximate the SPDE (8) and apply the Magnus expansion to it. In the end, we will recall the Euler–Maruyama scheme for the approximating SDE.

Space discretization and magnus expansion. Following Kamm et al. [11, Section 3.2], we will discretize the space variables but not time. After that we will vectorize the equation to derive a matrix-valued equation. We introduce the following two homogeneous grids for position and velocity respectively: $\mathbb{X}^{n_x}_{a_x,b_x}$ with n_x+2 points on the subset $[a_x,b_x]\subset\mathbb{R}$, $\mathbb{V}^{n_v}_{a_v,b_v}$ with n_v+2 points on the subset $[a_v,b_v]\subset\mathbb{R}$:

$$\mathbb{X}^{n_x}_{a_x,b_x} := \left\{x_i^{n_x} \in [a_v,b_v] : x_i^{n_v} = a_x + i\Delta x,\;\; i=0,\dots,n_v+1\right\}, \qquad \Delta x := \frac{b_x-a_x}{n_x+1}, \tag{9}$$

$$\mathbb{V}^{n_v}_{a_v,b_v} := \left\{v_j^{n_v} \in [a_v,b_v] : v_j^{n_v} = a_v + j\Delta v,\;\; j=0,\dots,n_v+1\right\}, \qquad \Delta v := \frac{b_v-a_v}{n_v+1}. \tag{10}$$

Let us denote by vec the isomorphism of transforming a matrix to a larger column-vector by stacking each column in the matrix below each other, i.e.

$$\begin{aligned} &\mathrm{vec} : \mathbb{R}^{n_x\times n_v} \to \mathbb{R}^{n_x\cdot n_v\times 1}, \\ &\quad A=[a_{ij}] \mapsto \mathrm{vec}(A) := [a_{1,1},\dots,a_{n_x,1},a_{1,2},\dots,a_{n_x,2},\dots,a_{1,n_v},\dots,a_{n_x,n_v}]^T. \end{aligned}$$

Moreover, let us define

$$u_t^{n_x,n_v} := \left(u_t(x_i, v_j)\right)_{\substack{i=1,\dots,n_x \\ j=1,\dots,n_v}}, \quad \phi^{n_x,n_v} := \left(\phi(x_i, v_j)\right)_{\substack{i=1,\dots,n_x \\ j=1,\dots,n_v}}, \quad \Phi^{n_x n_v} := \operatorname{vec}\left(\phi^{n_x,n_v}\right).$$

We consider the $n_x n_v$-dimensional SDE

$$dU_t^{n_x n_v} = BU_t^{n_x n_v} dt + AU_t^{n_x n_v} dW_t, \quad U_0^{n_x n_v} = \Phi^{n_x n_v}, \tag{11}$$

where A and B are $(n_x n_v \times n_x n_v)$-matrices, which will be defined via space-finite differences in the next paragraph in a way that $U_t^{n_x n_v} \approx \operatorname{vec}\left(u_t^{n_x,n_v}\right)$ with respect to a suitable norm. This equation can be solved by first computing its fundamental solution, then by multiplying it with the initial datum, i.e.

$$\begin{aligned} &dX_t = BX_t dt + AX_t dW_t, \quad X_0 = I \in \mathbb{R}^{n_x n_v \times n_x n_v}, \\ &U_t^{n_x n_v} = X_t \Phi^{n_x n_v}. \end{aligned}$$

The fundamental solution now can be approximated with the Magnus expansion, i.e. $X_t \approx \exp(Y_t)$. Since A and B will be very large in this case, we will utilize the sparsity of A and B, as well as using a special algorithm specifically designed to compute the matrix-exponential times a vector denoted by `expmvtay2`, which does not need to compute the whole matrix-exponential first. This is crucial for the implementation and explained in further detail in [9]. The approximation formulas for X_t, Y_t and their derivation are deferred to Appendix A.

Computation of A and B. The idea is to discretize the first and second-order derivatives we will use central differences with zero-boundary conditions. Therefore, let us introduce the following matrices corresponding to the finite differences

$$\begin{aligned} D^x &:= \frac{1}{2\Delta x}\operatorname{tridiag}^{n_x,n_x}(-1, 0, 1), & D^v &:= \frac{1}{2\Delta v}\operatorname{tridiag}^{n_v,n_v}(-1, 0, 1), \\ D^{xx} &:= \frac{1}{(\Delta x)^2}\operatorname{tridiag}^{n_x,n_x}(1, -2, 1), & D^{vv} &:= \frac{1}{(\Delta v)^2}\operatorname{tridiag}^{n_v,n_v}(1, -2, 1). \end{aligned}$$

Remark 2.1. In this paper, we will use central differences as our benchmark for the space derivatives, because we will see later on that the accuracy for our examples is already very good. However, the reader can choose at this point any finite-difference approximation while keeping the sparsity as low as possible. In Remark 2.2, we will discuss the importance of sparsity for the Magnus schemes further and conjecture that all methods will benefit similarly from high-order schemes. Conclusively, compact finite-difference schemes may be well-suited for higher-order approximations in space.

Additionally, we will introduce the following matrices corresponding to the coefficient functions on the discretized spatial grid

$$Z^w := \left(z^w(x_i, v_j)\right)_{\substack{i=1,\dots,n_x \\ j=1,\dots,n_v}}, \qquad \Sigma^w := \left(\sigma^w(x_i, v_j)\right)_{\substack{i=1,\dots,n_x \\ j=1,\dots,n_v}}$$

for $Z = F, G, H$, $z = f, g, h$, respectively, and $w \in \{x, v, xx, xv, vv\}$.

Let us start with discretizing the first-order derivative with respect to x. First, we replace the partial derivative by the first-order central differences and assume zero-boundary conditions, leading to

$$f^x(x_i, v_j)\partial_x u_t(x_i, v_j) \approx f^x(x_i, v_j)\frac{u_t(x_{i+1}, v_j) - u_t(x_{i-1}, v_j)}{2\Delta x}$$

for all $i = 1, \dots, n_x$ and $j = 1, \dots, n_v$. As aforementioned, we need to extract the correct coefficient matrix for the vectorized Eq. (11).

In our notations, a derivative in x is a multiplication of the corresponding finite-difference matrix from the left to $u_t^{n_x,n_v}$, i.e.

$$\left(\frac{u_t(x_{i+1}, v_j) - u_t(x_{i-1}, v_j)}{2\Delta x}\right)_{\substack{i=1,\dots,n_x \\ j=1,\dots,n_v}} = D^x u_t^{n_x,n_v}.$$

Using the Kronecker product, it is well-known for compatible matrices $D_1 U D_2 = C$ that

$$\operatorname{vec}(C) = \operatorname{vec}(D_1 U D_2) = \left(D_2^T \otimes D_1\right)\operatorname{vec}(U).$$

In our case, this leads to

$$\operatorname{vec}\left(D^x u_t^{n_x,n_v}\right) = \left(I_{n_v} \otimes D^x\right) U_t^{n_x n_v}.$$

Now, we need to deal with the coefficients as well. Denoting by $\odot$ the Hadamard, or elementwise, product, it is easy to see that

$$\operatorname{vec}\left(F^x \odot \left(D^x u_t^{n_x,n_v}\right)\right) = \operatorname{vec}\left(F^x\right) \odot \operatorname{vec}\left(\left(D^x u_t^{n_x,n_v}\right)\right) = \operatorname{diag}\left(\operatorname{vec}\left(F^x\right)\right) \operatorname{vec}\left(\left(D^x u_t^{n_x,n_v}\right)\right).$$

Using these two observations together yields

$$\left(f^x(x_i, v_j)\partial_x u_t(x_i, v_j)\right)_{\substack{i=1,\dots,n_x \\ j=1,\dots,n_v}} \approx \operatorname{diag}\left(\operatorname{vec}\left(F^x\right)\right)\left(I_{n_v} \otimes D^x\right) U_t^{n_x n_v}.$$

This reasoning holds true for all other derivatives as well, i.e. an operation in x is a matrix multiplication from the left and in v it is the matrix multiplication from the right with the transposed matrix.

Conclusively, we have

$$\begin{aligned} B :=\ & \operatorname{diag}(\operatorname{vec}(H)) + \operatorname{diag}\left(\operatorname{vec}\left(F^x\right)\right)\left(I_{n_v} \otimes D^x\right) + \operatorname{diag}(\operatorname{vec}(F^v))\left(D^v \otimes I_{n_x}\right) \\ & + \frac{1}{2}\operatorname{diag}\left(\operatorname{vec}\left(G^{xx}\right)\right)\left(I_{n_v} \otimes D^{xx}\right) + \operatorname{diag}\left(\operatorname{vec}\left(G^{xv}\right)\right)\left(D^v \otimes D^x\right) \\ & + \frac{1}{2}\operatorname{diag}(\operatorname{vec}(G^{vv}))\left(D^{vv} \otimes I_{n_x}\right), \\ A :=\ & \operatorname{diag}(\operatorname{vec}(\Sigma)) + \operatorname{diag}\left(\operatorname{vec}\left(\Sigma^x\right)\right)\left(I_{n_v} \otimes D^x\right) + \operatorname{diag}(\operatorname{vec}(\Sigma^v))\left(D^v \otimes I_{n_x}\right). \end{aligned}$$

Remark 2.2. We can see that the sparsity of B and A is mostly determined by the finite-differences matrices due to the elementwise product of the coefficient functions. The coefficient functions can further reduce or increase the sparsity by zero entries or cancellations. Additionally, this implies that the number of non-zero diagonals is independent of the size of B and A respectively even though the density of non-zero elements will decline for increasing dimensions.

An illustration of the sparsity pattern can be seen in Fig. 1: In the upper left corner is the pattern for A, followed by B, $[B, A]$, $[[B, A], A]$ and $[[B, A], B]$. The blue lines represent non-zero entries. The pictures are ordered by the number of non-zero diagonals, i.e. a diagonal with at least one non-zero entry, which are 2, 5, 5, 8 and 10 respectively. We can see that the sparsity decreases with the order of the commutator. In Section 3.2 we will see that one consequence of the reduced sparsity is a decrease in computational efficiency of the Magnus expansion.

Iterated Magnus expansion. An advantage of the Magnus expansion over pure time-iterative methods is that, as long as it is convergent, it does not need a previous time-iteration to compute the next time step. However, as we discussed in the introduction, the convergence time-interval can be small for the Magnus expansion. Therefore, if $T > 0$ is larger than the convergence radius, it is necessary to split the initial time interval $[0, T]$ into smaller sub-intervals $(0, t_1], (t_1, t_2], \dots, (t_n, T]$. As illustrated in Fig. 2, we will evaluate the Magnus expansion consecutively on each of them, i.e. use the terminal evaluation as the initial point of the next sub-interval.

We will call this method *iterated Magnus expansion*. On each sub-interval, the Magnus expansion still has the usual parallel-in-time features. Furthermore, due to its relatively large convergence region in time, we can use fewer iterations compared to other iterative methods, e.g. for the Euler–Maruyama scheme.

Euler–Maruyama. In this case, we do not need to vectorize the equation but we have to discretize the time-derivative as well. With the same notation and reasoning from above we obtain

$$\begin{aligned} u_{t_{k+1}}^{n_x,n_v} \approx u_{t_k}^{n_x,n_v} + \Bigg(& H \odot u_{t_k}^{n_x,n_v} + F^x \odot \left(D^x u_{t_k}^{n_x,n_v}\right) + F^v \odot \left(u_{t_k}^{n_x,n_v} (D^v)^T\right) \\ & + \frac{1}{2} G^{xx} \odot \left(D^{xx} u_{t_k}^{n_x,n_v}\right) + G^{xv} \odot \left(D^x u_{t_k}^{n_x,n_v} (D^v)^T\right) + \frac{1}{2} G^{vv} \odot \left(u_{t_k}^{n_x,n_v} (D^{vv})^T\right) \Bigg) \Delta t \\ & + \left(\Sigma \odot u_{t_k}^{n_x,n_v} + \Sigma^x \odot \left(D^x u_{t_k}^{n_x,n_v}\right) + \Sigma^v \odot \left(u_{t_k}^{n_x,n_v} (D^v)^T\right)\right) \Delta W_{t_k}, \end{aligned} \tag{12}$$

where $\Delta t := t_{k+1} - t_k > 0$ for any k and $\Delta W_{t_k} := W_{t_{k+1}} - W_{t_k}$. In the case of separable coefficients the Hadamard product can be replaced by matrix products with diagonal matrices from the left and right.

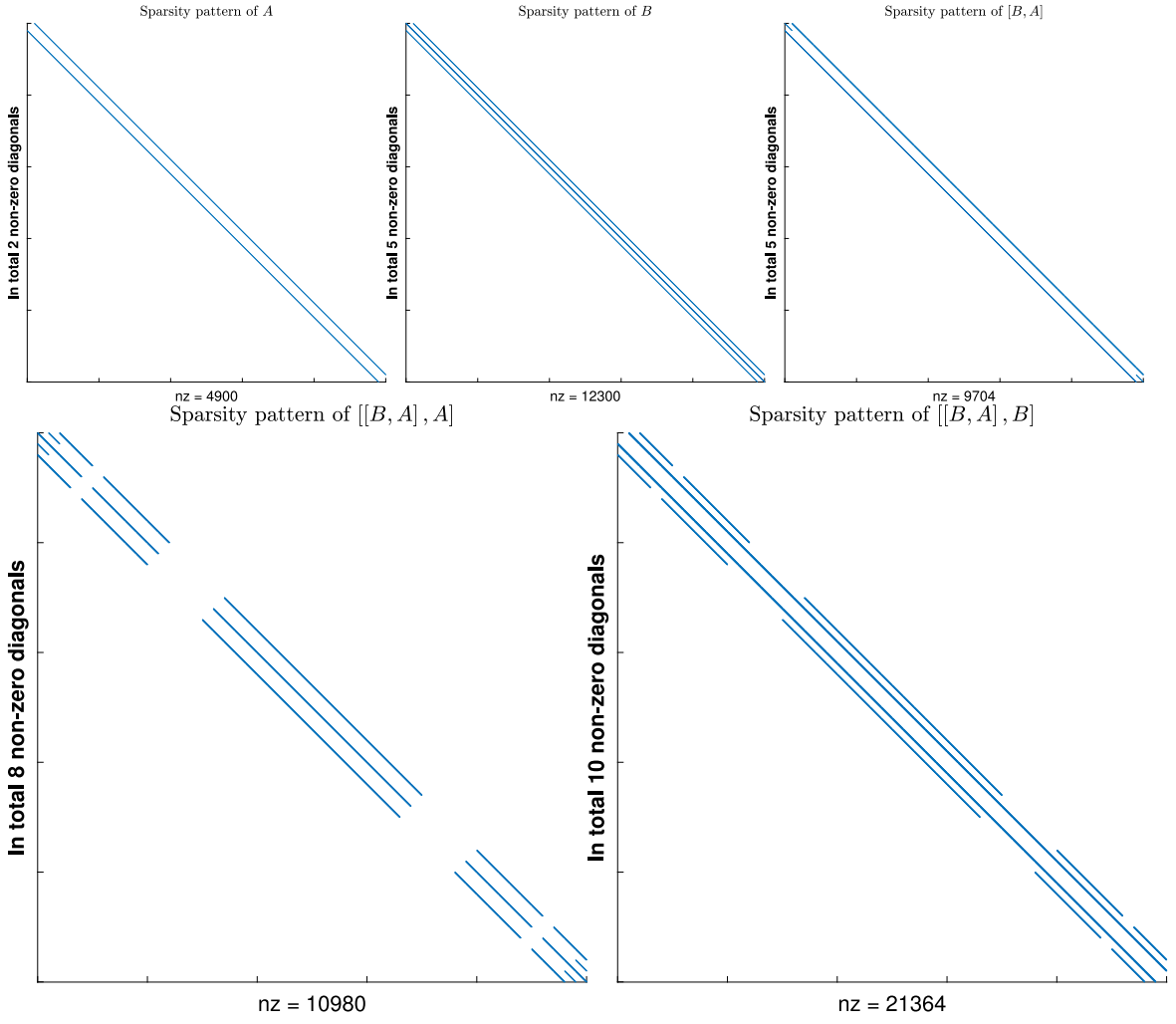

Fig. 1. Sparsity patterns of A, B and commutators of the SPDE from Section 3.1 with coefficients (16) and $d = 50$. nz stands for the number of non-zero entries.

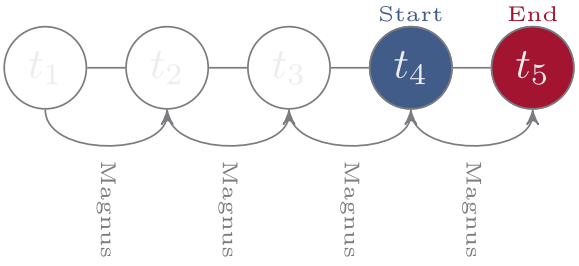

Fig. 2. Schematic of the iterative Magnus scheme.

3. Numerical experiments

We present here some numerical tests that demonstrate how the Magnus expansion can be efficiently applied to approximate the solutions of stochastic partial differential equations (SPDEs) with two spatial variables.

In this paper, we will focus on the stochastic Langevin equation. First, we consider the case of constant coefficients. This enables us to use an exact solution as a benchmark. We will perform experiments to test the

accuracy of the approximate solutions, elaborate on the computational times and discuss effects of the spatial boundaries. Afterwards, we will consider the stochastic Langevin equation with non-constant coefficients.

To this end, we will first elaborate on the implementation of the code and introduce some notation and norms for the error-analysis.

Implementation. As mentioned beforehand, we implement the algorithm iteratively as illustrated in Fig. 2. Now, we explain how to implement the algorithm for one time interval in `Matlab 2022a`. For this, let us have a close look at (7). We can see that, in this particular case, the computations of the Lebesgue-integrals are decoupled from the coefficients A and B. This makes it possible to evaluate the Lebesgue-integrals for all trajectories in parallel by using vectorization. In particular, we chose simple Riemann-sums to compute the integrals.

Now, we need to compute the commutators of the scheme. We found that for larger problems, e.g. $d \geq 100$ it is faster to copy A and B to the device and compute the commutators there. As it turns out, `Matlab 2022a` as of now does not support three-dimensional sparse matrices. Therefore, we decided to loop over all desired time evaluations (for our experiments only the terminal time) within the current sub-interval and over all simulations. Since, the Magnus expansion is parallel-in-time and parallel-in-simulations in each iteration, we use a threading environment to keep a single GPU busy with computing the expansion formulas and afterwards the matrix–vector exponential. This is also more efficient in terms of memory usage than using vectorization. As a side-note, it is straightforward to use multiple GPUs.

For $d \leq 100$ we use CPUs only because it is faster than copying to device in each iteration. For larger increasing spatial resolution the advantage of a single GPU increases more and more compared to CPUs only.

The choice of algorithm for evaluating the matrix–vector exponentials turned out to be crucial. We tested `expmv` by Al-Mohy and Higham [1] and `expmvtay2` by Ibáñez et al. [9], as well as a Krylov-subspace implementation called `expv`. For our purposes, `expmvtay2` was more accurate and significantly faster than the other methods. Their algorithm makes it possible to utilize the sparsity of the problem and is GPU-applicable without first computing the entire matrix exponential.

We tried to improve the performance of the Euler scheme as much as possible and tested different ways to compute the matrix multiplications within each iteration of the scheme and only save the values at the terminal points for a further speed-up. For the matrix multiplications we tested full and sparse matrix multiplications on GPU and CPUs. For computations using sparse matrices, one has to loop over the simulations, since there is no analogue to `pagemtimes` in the full case. Copying from host to device with our test cases was more expensive than a pure CPU implementation and the low-level multi-threading of `pagemtimes` surpassed the benefit of sparsity as well. Therefore, we use full matrix multiplication using all available CPU cores. On the machine we use, we could choose between 1 and 64 cores. There was no benefit after using 12 cores, which we use for all of our experiments.

We fix in our tests the number of simulations to $M = 100$. If we increase the number of simulations, both methods will benefit from more available computer hardware as expected. However, even for $M = 100$, the Magnus method would benefit immediately from another GPU, since the computations of matrix–vector exponentials are costly, which is the bottleneck of our implementation.

Error and notations. For the numerical error analysis we will make use of the following notations. We denote by U^{ref} and by U^{app} a benchmark and an approximate solution, respectively.

Henceforth, we will choose $\mathbb{V}^{n_v}_{a_v,b_v} = \mathbb{X}^{n_x}_{a_x,b_x}$ symmetric, centered around zero and set $d := n_x = n_v$ to make our analysis a bit easier. Moreover, we choose the cut-off region of $\mathbb{R}^2$ as $a_x = a_v = -4$ and $b_x = b_v = 4$ in all experiments.

Furthermore, we want to study the impact of the zero-boundary condition and therefore will use only a central part with varying size of the whole solution matrix at a given time, which is illustrated in Fig. 3. To vary the size we introduce a new parameter $\kappa = 0, 1, \dots$ indicating the $2^{-\kappa}$-th central part of the solution matrix U^d_t, which we will consider for our error analysis and denote the corresponding truncated matrix by $U^{\text{ref},d,\kappa}_t, U^{\text{app},d,\kappa}_t \in \mathbb{R}^{\left\lfloor \frac{d}{2^\kappa} \right\rfloor \times \left\lfloor \frac{d}{2^\kappa} \right\rfloor}$. Also we set

$$I^\kappa := \left\{ \left\lfloor \frac{d}{2} - \frac{d}{2^{\kappa+1}} \right\rfloor, \dots, \left\lfloor \frac{d}{2} + \frac{d}{2^{\kappa+1}} \right\rfloor \right\}$$

to collect the corresponding indices.

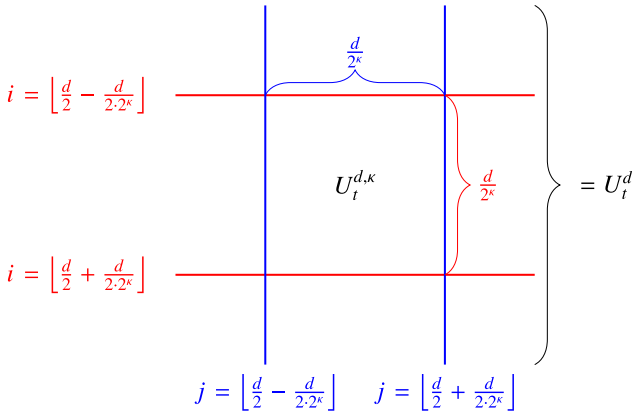

Fig. 3. Graphical representation of $U_t^{d,\kappa}$ compared to U_t^d for the error analysis to disregard boundary effects.

For our error analysis we will use the following three norms:

$$\mathrm{ME}_t^{d,\kappa} := \frac{1}{M}\sum_{m=1}^{M}\left|U_{t,m}^{\mathrm{ref},d,\kappa} - U_{t,m}^{\mathrm{app},d,\kappa}\right| \in \mathbb{R}^{\left\lfloor \frac{d}{2^\kappa}\right\rfloor \times \left\lfloor \frac{d}{2^\kappa}\right\rfloor} \tag{13}$$

$$\mathrm{AME}_t^{d,\kappa} := \frac{1}{|I^\kappa|^2}\sum_{i,j}\left(\mathrm{ME}_t^{d,\kappa}\right)_{ij} \in \mathbb{R} \tag{14}$$

$$\mathrm{Err}_t^{d,\kappa} := \frac{1}{M}\sum_{m=1}^{M}\frac{\left\|U_{t,m}^{\mathrm{ref},d,\kappa} - U_{t,m}^{\mathrm{app},d,\kappa}\right\|_F}{\left\|U_{t,m}^{\mathrm{ref},d,\kappa}\right\|_F} \in \mathbb{R}, \tag{15}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The first one is a matrix consisting of a mean absolute error between the reference solution and the approximation for each point in the grid $I^\kappa \times I^\kappa$ by taking the mean over all trajectories. The second error is the average of the first error for the corresponding grid indicated by κ. The larger κ the further away we are from the spatial boundary, as illustrated in Fig. 3. The third error is the mean of a relative error between the reference solution and the approximation taking all points in the region corresponding to κ into account. It will serve as our main error norm in this paper.

Regarding the Lebesgue discretization Δ_t^{Leb} we have observed that the computational times are more or less constant for $\Delta_t^{\mathrm{Leb}} \in \left[10^{-5}, 10^{-2}\right)$ and increase for lower Δ_t^{Leb} significantly. This can be explained by the fact that the discretization is only used to compute Lebesgue-integrals of the form $\int_0^t s^p W_s^q ds$. Therefore, the computation can be vectorized, which is very fast compared to the computation of the Magnus logarithm and the matrix–vector exponentials for $\Delta_t^{\mathrm{Leb}} \geq 10^{-5}$.

Since, there seems to be no computational drawback, we suggest to use a Lebesgue-discretization equal to 10^{-3} or 10^{-4} and fix it for all tests to $\Delta_t^{\mathrm{Leb}} = 10^{-4}$.

We have verified a linear behavior (with a slope less than one until the GPU is fully saturated) in the number of simulations M. In our experiments, we decided to use $M = 100$ simulations and display always the average computational times for one simulation.

Also the computational effort with respect to the finite time horizon T scales linearly for all methods. Thus, we use $T = 1$ as our terminal time.

Throughout the experiments we will use the notation in Table 1. We used for the calculations `Matlab 2022a` with the Parallel Computing Toolbox running on Debian GNU/Linux 10 (buster), on a machine with the following specifications: processor 2x AMD EPYC 7301 CPU @ 2.20 GHz, 256 GB RAM and a NVIDIA Tesla V100 PCIe (32 GB HBM2 RAM).

The next subsection is structured as follows: First, we derive the explicit solution of the Langevin equation. Then, we discuss the impact of the number of intervals Δ_t for the iterated Magnus scheme regarding computational times and errors. Next, we look at the boundary effects over time. This is followed by a comparison of the Magnus scheme with the Euler–Maruyama scheme with different sizes of the space grid.

Table 1
Notations for the numerical experiments.

`euler`	Euler–Maruyama scheme (12)
`m1`	Iterated Magnus scheme of order 1
`m2`	Iterated Magnus scheme of order 2
`m3`	Iterated Magnus scheme of order 3
M	Number of simulations
d	Number of grid points in $\mathbb{X}$ and $\mathbb{V}$
Δ_t	step-size of `euler` or Magnus
Δ_t^{Leb}	discretization of the Lebesgue-integrals for Magnus
M2, x	Magnus order 2 with step-size $\Delta_t = x$
M3, x	Magnus order 3 with step-size $\Delta_t = x$
E, x	`euler` with step-size $\Delta_t = x$

3.1. The Magnus expansion for the stochastic Langevin equation with constant coefficients

In this subsection, we apply the Magnus expansion to the stochastic Langevin equation. For further details and a solution theory in Hölder spaces under the weak Hörmander condition we refer the reader to [17].

In the constant coefficient case, the Langevin SPDE can be recovered from (8) setting

$$h \equiv f^v \equiv g^{xx} \equiv g^{xv} \equiv \sigma \equiv \sigma^x \equiv 0, \quad f_x(x, v) := -v, \quad g^{vv} \equiv a \in \mathbb{R}_{>0}, \quad \sigma^v(x, v) \equiv \sigma \in \mathbb{R}_{>0}. \tag{16}$$

In this special case, there exists an explicit fundamental solution Γ for $0 < \sigma < \sqrt{a}$ (cf. [17, p. 4 Proposition 1.1.]), which is given by

$$\Gamma(t, z; 0, \zeta) := \Gamma_0(t, z - m_t(\zeta)),$$
$$\Gamma_0\big(t, (x, v)\big) := \frac{\sqrt{3}}{\pi t^2 (a - \sigma^2)} \exp\left(-\frac{2}{a - \sigma^2}\left(\frac{v^2}{t} - \frac{3vx}{t^2} + \frac{3x^2}{t^3}\right)\right)$$

where $\zeta := (\xi, \eta)$ is the initial point and

$$m_t(\zeta) := \begin{pmatrix} \xi + t\eta - \sigma \int_0^t W_s ds \\ \eta - \sigma W_t \end{pmatrix}.$$

Having the fundamental solution, we can solve the Cauchy-problem by integrating against the initial datum, i.e.

$$u_t(z) = \int_{\mathbb{R}^2} \Gamma(t, z; 0, \zeta)\phi(\zeta) d\zeta, \qquad z = (x, v).$$

To get an explicit solution (up to the stochastic integral $\int_0^t W_s ds$) for the double integral we will choose ϕ to be Gaussian, i.e.

$$\phi(\xi, \eta) := \exp\left(-\frac{(\xi^2 + \eta^2)}{2}\right). \tag{17}$$

The formula for the exact solution is lengthy and its specific form is not instructive for the following experiments, therefore we decided to exclude it from this presentation. The interested reader can find it in the corresponding `Matlab 2022a` code, which is publicly available.

Having an exact benchmark solution we will now perform some numerical tests to judge the performance of the iterated Magnus scheme. Henceforth, the parameters for the stochastic Langevin equation will be $a = 1.1$ and $\sigma = \frac{1}{\sqrt{10}}$, so that $a - \sigma^2 = 1 > 0$.

Computational effort and errors with respect to the number of iterations. For this experiment, we fix the number of grid points in each space grid to $d = 200$ but vary step-size of the Magnus scheme Δ_t. In Fig. 4, we can see the corresponding results. The left y-axis shows the average computational times for one simulation in a log scale and the right y-axis the mean relative errors $\text{Err}_T^{d,4}$ also in a log scale. The computational times (in seconds) of `m2` are depicted in light blue and of `m3` in dark blue. Moreover, the mean relative errors for `m2` are orange and for `m3` red.

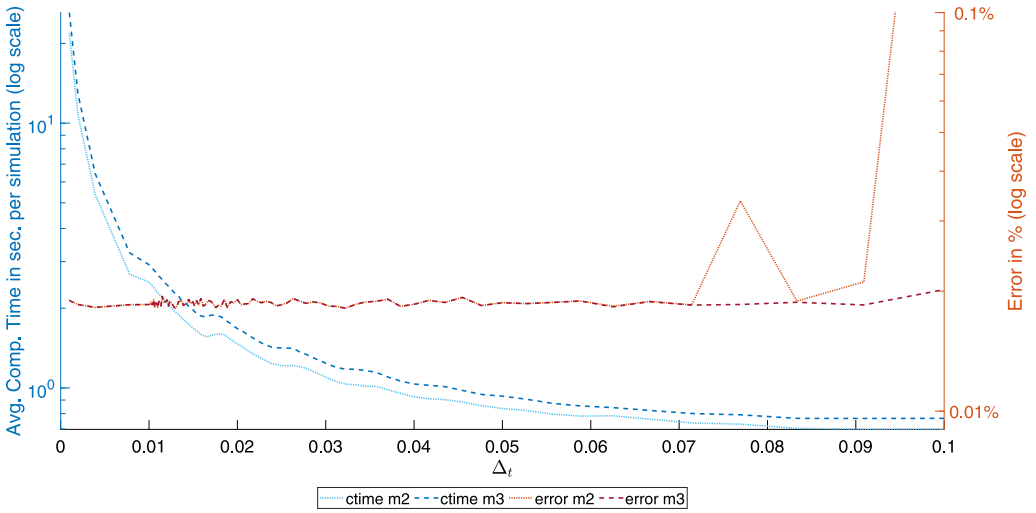

Fig. 4. Constant coefficients as in (16): Computational times and errors of the Magnus expansion for varying step-size Δ_t with fixed spatial dimension $d = 200$.

We can see that the mean relative errors start to fluctuate for a step-size larger than 0.065 for `m2` and see an explosion of `m2` around 0.085. For `m3` this phenomenon starts outside of the picture. The fluctuations begin right after $\Delta_t = 0.1$ and an explosion can be seen after $\Delta_t = 0.33$. The explosions for large step-sizes are not surprising, since the step-size is determined by the underlying stopping times for the convergence of the Magnus scheme. The fluctuations beforehand are most likely due to an interplay between error propagations due to larger step-sizes and the necessary Taylor-terms in `expmv`, and it indicates that `m3` is more stable than `m2`. Therefore, experiment indicates that any step-size less than 0.05, 0.1 for $d = 200$ is well within the convergence radius of `m2`, `m3`, respectively, and yields stable results.

For other spatial dimensions d, this breaking point might be different. Moreover, we can see that the computational time increases more and more for smaller step-size, while the error for both methods stays almost constant and close to each other.

This suggests that one should choose the step-size as large as possible for the iterated Magnus scheme to gain the maximal performance. However, being too greedy will lead to blow-ups of some trajectories.

Also, as a side note, usually with increasing spatial dimension d, one has to choose a smaller time step-size for the Magnus methods as well: this will be shown in Figs. 6–8.

Mean errors and boundary effects over time. For this experiment, we fix the grid points in each space grid to $d = 300$.

In Fig. 5, we can see the mean absolute errors of the entire spatial grid as a two-dimensional plot. A deep blue color indicates a small error and a bright yellow color an error up to 10^{-2}. The black rectangle is the corresponding region for $\kappa = 1$. The black number within the rectangle is the average mean absolute error of the corresponding region. The picture on the left-hand side is the area of errors at $t = 0.25$ and on the right-hand side at $t = 1$.

We can see that the errors in the upper right and lower left corners are significantly increasing over time. To explain this, one should note that the Langevin equation with this specific initial datum looks like a two-dimensional Gaussian at first and its shape changes on the diagonal from the lower left corner to the upper right corner over time more than on the other diagonal. Therefore, the cut-off region is getting too small for larger times leading to boundary effects in the error plots. This also explains why the upper left and lower right corner remain a stable small error. In the center of the error plots, we can see an increasing error over time. If this is due to the boundary effects, error propagation due to the iterated scheme, the error due to the order 3 truncation or the algorithm used for the matrix–vector exponential is not apparent in this illustration, we suspect a mixture of all of them.

Comparison to the Euler–Maruyama scheme. For this experiment, we will compare different choices of parameters for both the Magnus scheme and Euler–Maruyama scheme. There are essentially two major parameters contributing

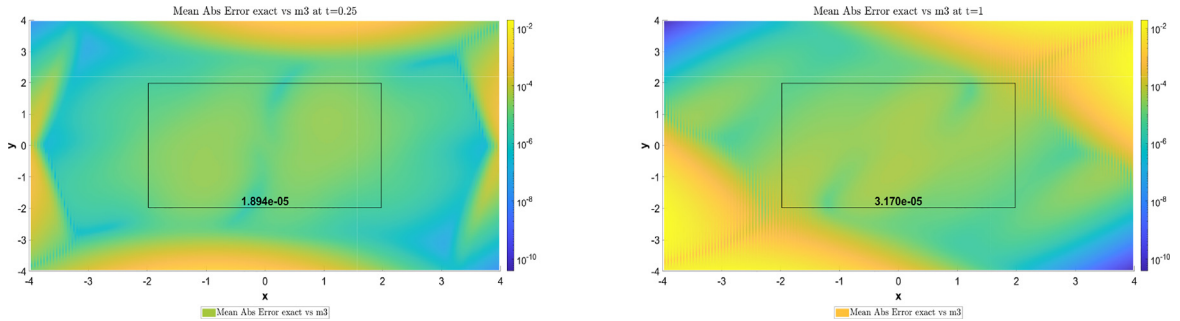

Fig. 5. Constant coefficients as in (16): Absolute Errors of `m3` compared to `exact` using $d = 300$ grid points at $t = 0.25$ (left) and $t = 1$ (right).

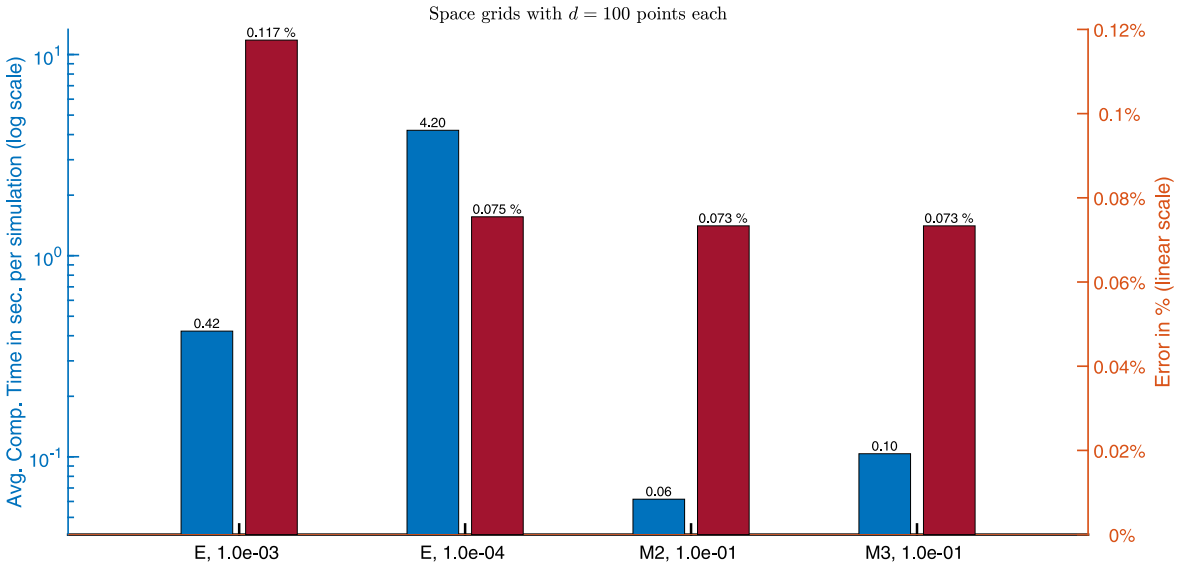

Fig. 6. Constant coefficients as in (16): Computational times and errors of the Magnus expansion and Euler scheme for $d = 100$.

to the possible accuracy. One is the time step-size of the individual schemes and the other one the space discretization. Hence, we compare Euler and Magnus methods with different time step-sizes for different space discretizations $d = 100, 200, 300$ to increase the level of accuracy. In Figs. 6–8, the left y-axis shows the average computational times in a log scale and the right y-axis the mean relative errors $\mathrm{Err}_T^{d,4}$ also in a linear scale. The computational times (in seconds) are depicted in the left blue columns and the mean relative errors in the red right columns for each method.

As mentioned in Table 1, "E, x" denotes Euler with step-size $\Delta_t = x$ and "M2, x", "M3, x" denotes Magnus with step-size $\Delta_t = x$ for order 2 and 3, respectively. In Fig. 6 we compare the errors and computational times of the methods with spatial dimension $d = 100$, in Fig. 7 with $d = 200$ and in Fig. 8 with $d = 300$.

Let us focus on Fig. 6 with $d = 100$. We can see that four different methods are compared: the Euler method with step-size $\Delta_t = 10^{-3}$ and $\Delta_t = 10^{-4}$, as well as the Magnus method with step-size $\Delta_t = 0.1$ of order 2 and order 3. It is notable that the Euler method with step-size $\Delta_t = 10^{-4}$ and the Magnus methods perform almost the same with respect to the error. The Euler method with step-size $\Delta_t = 10^{-3}$ has roughly double the error of the method with step-size $\Delta_t = 10^{-4}$ but is ten times faster. Overall, the Magnus methods were the fastest methods. The Magnus method of order two, three is 70, 42 times, respectively, faster than Euler method with step-size $\Delta_t = 10^{-4}$ and has a slightly better accuracy.

Now, let us consider Fig. 7 with $d = 200$. Again, we can see that two Euler methods and two Magnus methods are compared but this time we have a step-size $\Delta_t = 0.05$ for the Magnus methods. Similarly, to Fig. 6, we can

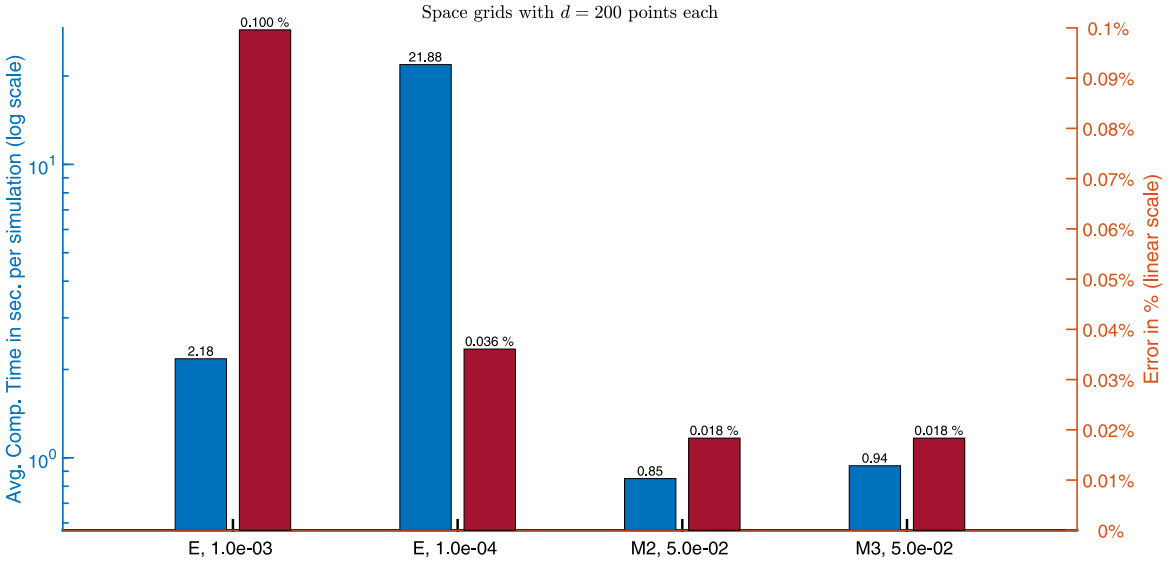

Fig. 7. Constant coefficients as in (16): Computational times and errors of the Magnus expansion and Euler scheme for $d = 200$.

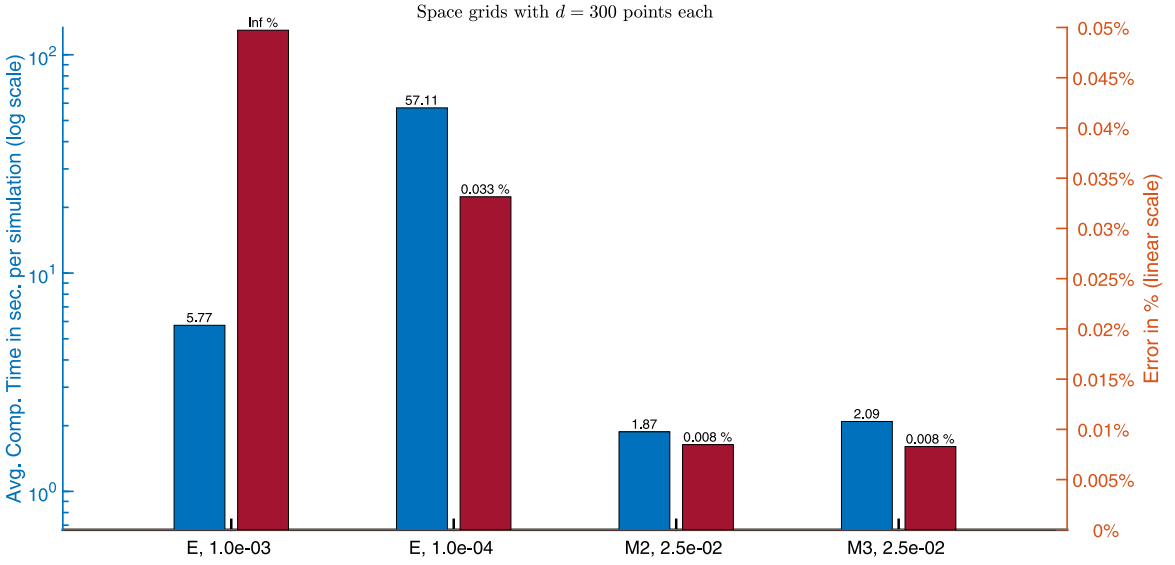

Fig. 8. Constant coefficients as in (16): Computational times and errors of the Magnus expansion and Euler scheme for $d = 300$.

see that the Euler method with step-size $\Delta_t = 10^{-3}$ performed worst and the Magnus methods best in terms of accuracy. However, this time the Euler method with step-size $\Delta_t = 10^{-4}$ has twice the error compared to the Magnus methods and is 25, 23 times slower than the Magnus method with order 2, 3, respectively.

In Fig. 8, with $d = 300$ the Euler method with step-size $\Delta_t = 10^{-3}$ is exploding, since its stability criterion is violated and its errors are ∞. Therefore, we compare the Euler method with step-size $\Delta_t = 10^{-4}$ to the Magnus method with step-size $\Delta_t = 0.025$ with order two and three. This time the Euler method is four times worse in terms of accuracy and 30, 27 times slower than Magnus with order two, three, respectively. We also performed tests with an Euler method using step-size $\Delta_t = 10^{-5}$. Its accuracy was still slightly worse compared to the Magnus methods and its computational time ten times slower than the Euler scheme in the figure. This results in a speed-up of order 250 of the Magnus method compared to an Euler scheme with similar accuracy.

Overall, from these observations it is clear that an Euler scheme with a fine time-discretization is essential to make it comparable to the iterated Magnus scheme in terms of accuracy. Moreover, increasing the number of grid points is leading to less accurate errors using the Euler method with step-size $\Delta_t = 10^{-4}$ compared to the Magnus schemes with corresponding step-sizes, while the Magnus methods remain roughly 30 times faster in all tests.

Remark 3.1. If we have a close look at all tests from above, we can see that all of them share a common feature, namely for reasonable parameters `m2` and `m3` were always close. Therefore, this leads to a natural step-size control in time by comparing the results of Magnus order 2 to order 3. If they are closer than a given tolerance then the step-size is small enough, otherwise make it smaller by a given factor.

With this method, the computation of the Magnus logarithms up to order 3 can be reused for Magnus order 2. However, two matrix–vector exponentials for each trajectory are necessary to determine if the time-step is rejected. For implementations with a lot of trajectories, one can think about using less randomly chosen trajectories to determine the correct step-size to increase the overall performance.

Remark 3.2. The Magnus expansion holds an advantage over all other finite-difference method in the deterministic case. Inspecting the approximation formulas in the case $A \equiv 0$ reveals immediately that the Magnus expansion is exact, at order 1, up to the initial space discretization for x and v, meaning that its accuracy is far more superior than e.g. explicit and implicit Euler-schemes.

Remark 3.3. We would like to point out, that it is straightforward to use the iterated Magnus scheme in the case of deterministic piecewise constant coefficients in time by partitioning the time interval according to the piecewise definition. Then on each of these intervals the Magnus expansion formulas for constant coefficients hold.

Remark 3.4. We also implemented an explicit Milstein scheme, which analogue to (12) reads as

$$\begin{aligned}
\tilde{u}_{t_k}^{n_x,n_v} &:= \Sigma \odot u_{t_k}^{n_x,n_v} + \Sigma^x \odot \left(D^x u_{t_k}^{n_x,n_v}\right) + \Sigma^v \odot \left(u_{t_k}^{n_x,n_v} (D^v)^T\right) \\
u_{t_{k+1}}^{n_x,n_v} &\approx u_{t_k}^{n_x,n_v} + \Bigg(H \odot u_{t_k}^{n_x,n_v} + F^x \odot \left(D^x u_{t_k}^{n_x,n_v}\right) + F^v \odot \left(u_{t_k}^{n_x,n_v} (D^v)^T\right) \\
&\quad + \frac{1}{2} G^{xx} \odot \left(D^{xx} u_{t_k}^{n_x,n_v}\right) + G^{xv} \odot \left(D^x u_{t_k}^{n_x,n_v} (D^v)^T\right) + \frac{1}{2} G^{vv} \odot \left(u_{t_k}^{n_x,n_v} (D^{vv})^T\right) \Bigg) \Delta t \\
&\quad + \tilde{u}_{t_k}^{n_x,n_v} \Delta W_{t_k} + \left(\Sigma \odot \tilde{u}_{t_k}^{n_x,n_v} + \Sigma^x \odot \left(D^x \tilde{u}_{t_k}^{n_x,n_v}\right) + \Sigma^v \odot \left(\tilde{u}_{t_k}^{n_x,n_v} (D^v)^T\right) \right) \frac{(\Delta W_{t_k})^2 - \Delta t}{2}.
\end{aligned} \tag{18}$$

The interested reader is referred to [8,18] for more details and further variants of the Milstein scheme.

In the case of constant coefficients (16), our tests showed that the stability of this Milstein scheme is the same as the Euler scheme, meaning that for $d = 200$ the Milstein scheme with time step-size $\Delta_t = 10^{-2}$ explodes and converges for $\Delta_t = 10^{-3}$. In the latter case, the Milstein scheme had almost the same accuracy as `m3` but was roughly 20 percent slower than the Euler-scheme with $\Delta_t = 10^{-3}$ due to the additional higher-order terms in (18). For $d = 300$ we saw the same: the Milstein scheme with $\Delta_t = 10^{-4}$ was roughly 30 percent slower than `euler` with $\Delta_t = 10^{-4}$ with similar accuracy to `m3`, and it also explodes for $\Delta_t = 10^{-3}$.

For brevity, we preferred not to add these numerical tests to the manuscript. However, our codes on https://github.com/kevinkamm/MagnusSPDE2D include the implementation of the Milstein scheme.

3.2. The Magnus expansion for the stochastic Langevin equation with variable coefficients

In this brief subsection, we will perform some tests in the case of variable coefficients. In particular, we choose bounded, smooth coefficients of the form

$$h \equiv f^v \equiv g^{xx} \equiv g^{xv} \equiv \sigma \equiv \sigma^x \equiv 0, \tag{19}$$

$$f_x(x,v) := -v, \quad g^{vv}(x,v) = a\left(1 + \frac{1}{x^2+1}\right), \quad \sigma^v(x,v) \equiv \sigma\sqrt{1 + \frac{1}{x^2+1}}, \tag{20}$$

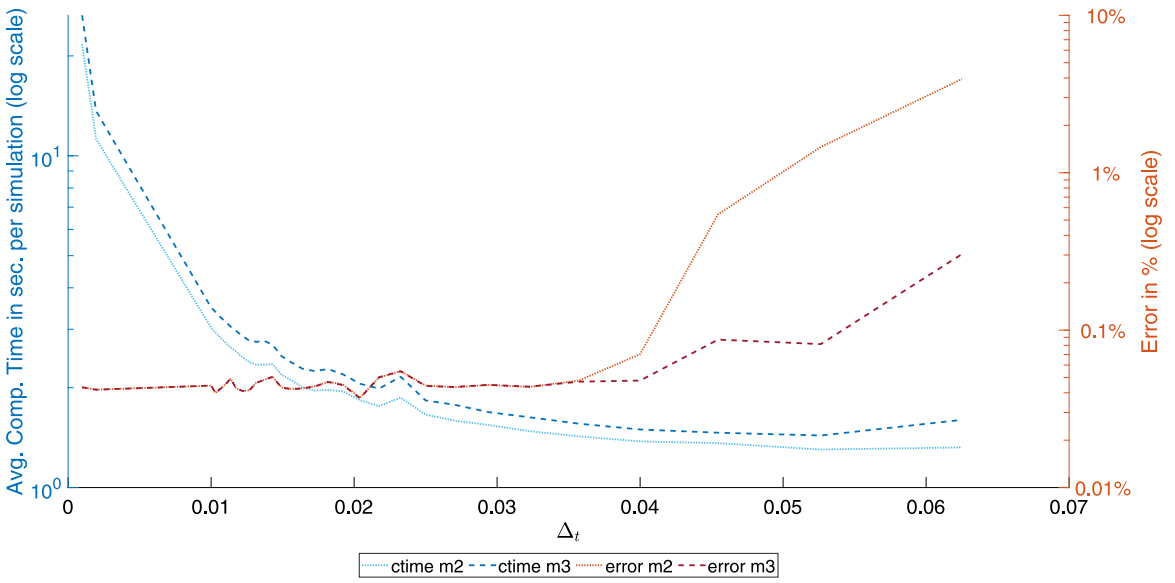

Fig. 9. Variable coefficients as in (20): Computational times and errors compared to Euler with $\Delta_t = 10^{-4}$ of the Magnus expansion for varying step-size Δ_t with fixed spatial dimension $d = 200$.

with $a, \sigma \in \mathbb{R}_{>0}$, as above, satisfying $g^{vv}(x, v) - (\sigma^v(x, v))^2 > 0$, as in the constant coefficient case. We will also use the same initial condition as in (17).

Analog to Fig. 4, we show in Fig. 9 the case of varying step-sizes for the Magnus method with fixed spatial discretization $d = 200$. The average computational times of `m2` and `m3` in seconds, per simulation, are again depicted in light blue and dark blue, respectively. The errors are this time with respect to the Euler method with $\Delta_t = 10^{-4}$, since an exact solution is not available, and again illustrated as orange for `m2` and red for `m3`.

The computational times look similar to the constant coefficient case and we can notice a slight increase of the computational times for `m3` at around 0.055. This behavior is the same in the constant coefficient case, but can only be seen at step-sizes larger than $\Delta_t = 0.1$. Therefore it is not part of the previous figure. This can be explained by the necessity to use more Taylor terms in the matrix–vector exponential to keep the errors small.

Moreover, we can see that the mean relative errors start to increase for a step-size larger than 0.035 for `m2` and 0.045 for `m3`. This happens earlier than in the constant coefficient case and is the expected behavior.

In our next experiment, similarly to Figs. 6–8, we show in Table 2 the mean relative errors and computational times for $d = 100, 200, 300$. This time we use an Euler method with $\Delta_t = 10^{-5}$ as our reference solution and compare an Euler method with $\Delta_t = 10^{-4}$, as well as `m2` and `m3` to it. The results are given in Table 2.

We can see that computational times are roughly twice higher for the Magnus scheme than in the constant coefficient case. The Euler scheme did not suffer from a performance decrease. As aforementioned this increase of computational effort can be attributed to the decreased sparsity. Therefore, the Magnus scheme is only 12 times faster than the Euler scheme. Regarding the accuracy, we can see in all cases that the Magnus expansion is more accurate than `euler` with $\Delta_t = 10^{-4}$.

All in all, we come to the same conclusion as in the constant coefficient case.

4. Conclusions

We have seen how to derive the Itô-stochastic Magnus expansion for SDEs with constant matrices and used it to solve two-dimensional SPDEs with a given initial datum numerically. We derived an approximation scheme for a generic parabolic SPDE with two space variables, then tested it on a special class of Langevin-type SPDEs. The scheme has an excellent accuracy and its advantage in terms of computational effort excels for higher spatial resolution. For instance, in the case of constant coefficients to have roughly the same accuracy for $d = 400$ we need an Euler scheme with $\Delta_t = 10^{-5}$ taking approximately 21 minutes, in average per trajectory, while Magnus

Table 2
Variable coefficients as in (20): Computational times and errors compared to Euler with $\Delta_t = 10^{-5}$ of the Magnus expansion for different spatial dimension $d = 100, 200, 300$.

Method	Mean Rel. Error (in %)	Comp. Time (in sec./simulation)
$d = 100$		
euler, $\Delta_t = 10^{-3}$	0.147%	0.43
euler, $\Delta_t = 10^{-4}$	0.047%	4.24
euler, $\Delta_t = 10^{-5}$	–	42.24
m2, $\Delta_t = 0.05$	0.015%	0.13
m3, $\Delta_t = 0.05$	0.014%	0.24
$d = 200$		
euler, $\Delta_t = 10^{-3}$	∞%	2.03
euler, $\Delta_t = 10^{-4}$	0.045%	21.02
euler, $\Delta_t = 10^{-5}$	–	209.02
m2, $\Delta_t = 0.025$	0.013%	1.66
m3, $\Delta_t = 0.025$	0.012%	1.83
$d = 300$		
euler, $\Delta_t = 10^{-3}$	∞%	5.79
euler, $\Delta_t = 10^{-4}$	0.041%	58.41
euler, $\Delta_t = 10^{-5}$	–	582.88
m2, $\Delta_t = 0.01$	0.014%	4.28
m3, $\Delta_t = 0.01$	0.014%	5.0

order 3 takes only 4.62 s using $\Delta_t = 0.01$, $\Delta_t^{\text{Leb}} = 10^{-4}$. This is a speed-up by a factor 280 just using one GPU while sparsity ensures an almost equal memory demand. Adding multiple GPUs will lead to further improvements with respect to computational times.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Magnus expansion formulas for constant coefficients

In order to make the paper self-contained, in this section, we demonstrate how to heuristically derive the Itô-stochastic Magnus expansion formula for the equation

$$dX_t = BX_t dt + AX_t dW_t, \quad X_0 = I_d, \tag{A.1}$$

where the coefficients $B, A \in \mathbb{R}^{d\times d}$ are constant matrices and W is a standard one-dimensional Brownian motion.

We make the ansatz $X_t = \exp(Y_t)$, where

$$Y_t = \int_0^t \mu(s, Y_s)\, ds + \int_0^t \sigma(s, Y_s)\, dW_s, \quad Y_0 = 0_{\mathbb{R}^{d\times d}}.$$

By Itô's formula (cf. [11, p. 8 Lemma 1 and p. 9 Proposition 1]) we have

$$\begin{aligned} dX_t &= BX_t + AX_t dW_t \\ &= d\exp(Y_t) \\ &= \left(\mathcal{L}_{Y_t}(\mu(s, Y_s)) + \frac{1}{2}\mathcal{Q}_{Y_t}(\sigma(t, Y_t), \sigma(t, Y_t))\right)\exp(Y_t)\, dt \\ &\quad + \mathcal{L}_{Y_t}(\sigma(t, Y_t))\exp(Y_t)\, dW_t, \end{aligned}$$

where the operators $\mathcal{L}$ and $\mathcal{Q}$ associated to the first and second order derivative of the exponential map are given by

$$\mathcal{L}_Y(M) := \sum_{n=0}^{\infty} \frac{1}{(n+1)!} \mathrm{ad}_Y^n(M),$$
$$\mathcal{Q}_Y(M,N) := \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} \frac{\mathrm{ad}_Y^n(M)}{(n+1)!}\frac{\mathrm{ad}_Y^m(N)}{(m+1)!} + \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} \frac{\left[\mathrm{ad}_Y^n(N), \mathrm{ad}_Y^m(M)\right]}{(n+m+2)(n+1)!m!}.$$

A comparison of coefficients yields

$$B \overset{!}{=} \mathcal{L}_{Y_t}\left(\mu\left(t, Y_s\right)\right) + \frac{1}{2}\mathcal{Q}_{Y_t}\left(\sigma\left(t, Y_t\right), \sigma\left(t, Y_t\right)\right)$$
$$A \overset{!}{=} \mathcal{L}_{Y_t}\left(\sigma\left(t, Y_t\right)\right).$$

Now, Baker's lemma (cf. [11, p. 10 Lemma 2]) provides us with suitable conditions and a series representation for the inverse of the first-order derivative operator $\mathcal{L}$, i.e.

$$\sigma\left(t, Y_t\right) \equiv \sigma\left(Y_t\right) = \sum_{n=0}^{\infty} \frac{\beta_n}{n!} \mathrm{ad}_{Y_t}^n(A) \tag{A.2}$$

$$\mu\left(t, Y_t\right) \equiv \mu\left(Y_t\right) = \sum_{k=0}^{\infty} \frac{\beta_k}{k!} \mathrm{ad}_{Y_t}^k \left(B - \frac{1}{2}\sum_{n=0}^{\infty}\sum_{m=0}^{\infty} \frac{\mathrm{ad}_{Y_t}^n(\sigma(Y_t))}{(n+1)!}\frac{\mathrm{ad}_{Y_t}^m(\sigma(Y_t))}{(m+1)!} + \frac{\left[\mathrm{ad}_{Y_t}^n\left(\sigma\left(Y_t\right)\right), \mathrm{ad}_{Y_t}^m\left(\sigma\left(Y_t\right)\right)\right]}{(n+m+2)(n+1)!m!} \right), \tag{A.3}$$

The Bernoulli numbers are denoted by β_k and we recall that $\beta_0 = 1$, $\beta_1 = -\frac{1}{2}$, $\beta_2 = \frac{1}{6}$, $\beta_3 = 0$ and $\beta_4 = -\frac{1}{30}$.

Now, we know the coefficients of Y_t and solve the Itô-SDE by means of a stochastic Picard iteration, i.e. we start at $n = 0$ with $Y_t^0 = 0_{\mathbb{R}^{d\times d}}$ and iterate for $n \geq 1$

$$Y_t^n = \int_0^t \mu\left(s, Y_s^{n-1}\right) ds + \int_0^t \sigma\left(s, Y_s^{n-1}\right) dW_s. \tag{A.4}$$

In order to derive the Magnus expansion formulas we will introduce some bookkeeping parameters $\epsilon, \delta > 0$ and substitute A by ϵA, as well as B by δB. Henceforth, we will denote the nth order Picard iteration with the substitution by $Y_t^{n,\epsilon,\delta}$.

Order 1. Let us derive the first-order Magnus expansion, meaning we are interested in all terms with the first power of ϵ and δ in the first-order Picard iteration. Thus, we insert $Y_t^{0,\epsilon,\delta} \equiv 0$ into (A.4). Notice, that the zero matrix commutes with all matrices and therefore by definition of $\mathrm{ad}_Y^0(A) = A$ we have

$$\sigma\left(Y_t^{0,\epsilon,\delta}\right) = \epsilon A.$$

Inserting this into the formula for μ yields

$$\mu\left(Y_t^{0,\epsilon,\delta}\right) = \delta B - \frac{1}{2}\epsilon^2 A^2,$$

because A commutes with itself as well.

Since, the Itô-correction term is of order ϵ^2 it will not be part of the first-order Magnus expansion and we have

$$Y_t^1 = \int_0^t B ds + \int_0^t A dW_s = Bt + AW_t.$$

Order 2. To derive the second-order Magnus expansion let us first think about how many terms we need at most in the infinite sums of (A.2) and (A.3) to disregard the majority of the high-order terms.

Let us consider (A.2) first and let $N = 2$ denote the desired order of the Magnus expansion. The operator ad^0 will at least result in a first order term in ϵ or δ. The commutator $\mathrm{ad}_Y^1(X) = [Y, X] = YX - XY$ will at least have second order terms, because X and Y will at least have a first-order ϵ or δ coefficient. Now, the nested commutator $\mathrm{ad}_Y^2(X) = [Y, [Y, X]]$ will have at least a third-order term for the same reason. Therefore, to compute the Nth order

of the Magnus expansion we only need to consider the infinite sum in (A.2) up to $N-1$. Now, let us consider (A.3). We will split our consideration into two parts by linearity and focus for the moment on

$$\sum_{k=0}^{\infty} \mathrm{ad}_{Y_t}^k (B).$$

With exactly the same arguments as for (A.2) we only need to consider $N-1$ terms of this infinite sum for the Nth order Magnus expansion.

Let us turn now to the rest, the Itô-correction term, of (A.3), and split it again in two parts, i.e. we consider

$$\sum_{k=0}^{\infty} \mathrm{ad}_{Y_t}^k \left(\sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{\mathrm{ad}_{Y_t}^n (\sigma (Y_t))}{(n+1)!} \frac{\mathrm{ad}_{Y_t}^m (\sigma (Y_t))}{(m+1)!} \right).$$

In the case $0 = k = n = m$ we will get σ^2, which has at least order 2 terms. Hence, we will only need $k = 1, \ldots, N-2$. This also means, that n and m cannot exceed $N-2$.

Increasing k and keeping $n = m = 0$ will also increase the order due to the nested commutators as in the case for (A.2). Therefore, the lowest k can have the most terms resulting from n and m. Vice versa, the higher n and m the lower k must be to have small enough orders.

Taking all these considerations into account, we can use the following formula to cover all necessary terms up until order N:

$$\sigma_N (Y_t) := \sum_{n=0}^{N-1} \frac{\beta_n}{n!} \mathrm{ad}_{Y_t}^n (A), \tag{A.5}$$

$$\mu_N (Y_t) := \sum_{k=0}^{N-1} \frac{\beta_k}{k!} \mathrm{ad}_{Y_t}^k (B) \tag{A.6}$$

$$-\frac{1}{2} \sum_{k=0}^{N-2} \frac{\beta_k}{k!} \mathrm{ad}_{Y_t}^k \left(\sum_{n=0}^{N-2-k} \sum_{m=0}^{N-2-k} \frac{\mathrm{ad}_{Y_t}^n (\sigma_{N-k} (Y_t))}{(n+1)!} \frac{\mathrm{ad}_{Y_t}^m (\sigma_{N-k} (Y_t))}{(m+1)!} + \frac{\left[\mathrm{ad}_{Y_t}^n (\sigma_{N-k} (Y_t)), \mathrm{ad}_{Y_t}^m (\sigma_{N-k} (Y_t)) \right]}{(n+m+2)(n+1)!m!} \right). \tag{A.7}$$

There will still be a lot of higher-order terms in (A.5) and (A.7) but it will make our derivation of the formulas easier.

Now, we compute $\sigma_2 \left(Y_t^{1,\epsilon,\delta} \right)$:

$$\begin{aligned} \sigma_2 \left(Y_t^{1,\epsilon,\delta} \right) &= \frac{1}{0!} \mathrm{ad}_{Y_t^{1,\epsilon,\delta}}^0 (\epsilon A) - \frac{1}{2} \mathrm{ad}_{Y_t^{1,\epsilon,\delta}}^1 (\epsilon A) \\ &= \epsilon A - \frac{1}{2} \left[Y_t^{1,\epsilon,\delta}, \epsilon A \right] \\ &= \epsilon A - \frac{1}{2} [\delta B, \epsilon A] t - \frac{1}{2} [\epsilon A, \epsilon A] W_t \\ &= \epsilon A - \frac{1}{2} [\delta B, \epsilon A] t. \end{aligned}$$

Next, we compute $\mu_2 \left(Y_t^{1,\epsilon,\delta} \right)$:

$$\begin{aligned} \mu_2 \left(Y_t^{1,\epsilon,\delta} \right) = {} & \frac{1}{0!} \mathrm{ad}_{Y_t^{1,\epsilon,\delta}}^0 (\delta B) - \frac{1}{2} \mathrm{ad}_{Y_t^{1,\epsilon,\delta}}^1 (\delta B) \\ & - \frac{1}{2} \left(\frac{1}{0!} \left(\frac{\mathrm{ad}_{Y_t^{1,\epsilon,\delta}}^0 \left(\sigma_2 \left(Y_t^{1,\epsilon,\delta} \right) \right)}{1!} \frac{\mathrm{ad}_{Y_t^{1,\epsilon,\delta}}^0 \left(\sigma_2 \left(Y_t^{1,\epsilon,\delta} \right) \right)}{1!} \right. \right. \\ & \left. \left. + \frac{\left[\mathrm{ad}_{Y_t^{1,\epsilon,\delta}}^0 \left(\sigma_2 \left(Y_t^{1,\epsilon,\delta} \right) \right), \mathrm{ad}_{Y_t^{1,\epsilon,\delta}}^0 \left(\sigma_2 \left(Y_t^{1,\epsilon,\delta} \right) \right) \right]}{2} \right) \right) \end{aligned}$$

$$= \delta B - \frac{1}{2}\left[Y_t^{1,\epsilon,\delta}, \delta B\right] - \frac{1}{2}\left(\left(\epsilon A - \frac{1}{2}\left[\delta B, \epsilon A\right] t\right)^2 + \frac{\left[\epsilon A - \frac{1}{2}\left[\delta B, \epsilon A\right] t, \epsilon A - \frac{1}{2}\left[\delta B, \epsilon A\right] t\right]}{2}\right).$$

Counting ϵ and δ we can already see that

$$\left(\epsilon A - \frac{1}{2}\left[\delta B, \epsilon A\right] t\right)^2 \approx \epsilon^2 A^2,$$
$$\left[\epsilon A - \frac{1}{2}\left[\delta B, \epsilon A\right] t, \epsilon A - \frac{1}{2}\left[\delta B, \epsilon A\right] t\right] \approx \left[\epsilon A, \epsilon A\right] = 0.$$

Thus,

$$\mu\left(Y_t^{1,\epsilon,\delta}; 2\right) = \delta B - \frac{1}{2}\left[\epsilon A, \delta B\right] W_t - \frac{1}{2}\epsilon^2 A^2.$$

In total, we have

$$Y_t^2 = \int_0^t B - \frac{1}{2}\left[A, B\right] W_s - \frac{1}{2}A^2 ds + \int_0^t A - \frac{1}{2}\left[B, A\right] s dW_s.$$

In a next step, we can apply Itô's formula to replace the stochastic integral by a Lebesgue integral like follows:

Lemma A.1. *For $p, p_1, p_2, q, q_1, q_2 \in \mathbb{N}_0$ we have*

$$\int_0^t s^p W_s^q dW_s = \frac{1}{q+1}\left(t^p W_t^{q+1} - \int_0^t \left[\frac{q(q+1)}{2} s^p W_s^{q-1} + p W_s^{q+1} s^{p-1}\right] ds\right) \tag{A.8}$$

$$\int_0^t s^{p_1} \int_0^s r^{p_2} W_r^q dr ds = \frac{1}{1+p_1}\left(t^{1+p_1} \int_0^t s^{p_2} W_s^q ds - \int_0^t s^{1+p_1+p_2} W_s^q ds\right) \tag{A.9}$$

$$\begin{aligned}\int_0^t s^{p_1} W_s^{q_1} \int_0^s r^{p_2} W_r^{q_2} dr dW_s = \frac{1}{q_1+1}\Bigg(& t^{p_1} W_t^{q_1+1} \int_0^t s^{p_2} W_s^{q_2} ds \\ & - \int_0^t s^{p_1+p_2} W_s^{q_1+q_2+1} ds \\ & - \frac{q_1(q_1+1)}{2} \int_0^t s^{p_1} W_s^{q_1-1} \int_0^s r^{p_2} W_r^{q_2} dr ds \\ & - p_1 \int_0^t s^{p_1-1} W_s^{q_1+1} \int_0^s r^{p_2} W_r^{q_2} dr ds \Bigg).\end{aligned} \tag{A.10}$$

Proof. Note that (A.8) is a special case of (A.10) by setting $p = p_1 - 1$, $q = q_1$ and $p_2 = q_2 = 0$.

Now, we show (A.9). With Itô's product rule we get

$$\begin{aligned} d\left(s^{p_1} \int_0^s r^{p_2} W_r^q dr\ s\right) &= s^{p_1} \int_0^s r^{p_2} W_r^q dr ds + s\ d\left(s^{p_1} \int_0^s r^{p_2} W_r^q dr\right) + 0 \\ &= s^{p_1} \int_0^s r^{p_2} W_r^q dr ds + s\ d\left(s^{p_1} s^{p_2} W_s^q ds + \int_0^s r^{p_2} W_r^q dr\, p_1 s^{p_1-1} ds\right) \\ &= (1+p_1)\, s^{p_1} \int_0^s r^{p_2} W_r^q dr ds + s^{1+p_1+p_2} W_s^q ds. \end{aligned}$$

Rearranging the equation yields the claim.

Next, we show (A.10). With Itô's product rule we get

$$\begin{aligned} d\left(s^{p_1} W_s^{q_1} \int_0^s r^{p_2} W_r^{q_2} dr W_s\right) = s^{p_1} W_s^{q_1} \int_0^s r^{p_2} W_r^{q_2} dr dW_s + W_s d\left(s^{p_1} W_s^{q_1} \int_0^s r^{p_2} W_r^{q_2} dr\right) \\ + d\left\langle \cdot^{p_1} W_\cdot^{q_1} \int_0^\cdot r^{p_2} W_r^{q_2} dr, W_\cdot \right\rangle_s. \end{aligned}$$

Now, use Itô's product rule and Itô's formula on the following term

$$\begin{aligned}
d\left(s^{p_1}W_s^{q_1}\int_0^s r^{p_2}W_r^{q_2}dr\right) &= s^{p_1}W_s^{q_1}d\left(\int_0^s r^{p_2}W_r^{q_2}dr\right)+\int_0^s r^{p_2}W_r^{q_2}drd\left(s^{p_1}W_s^{q_1}\right)+0\\
&= s^{p_1+p_2}W_s^{q_1+q_2}ds+\int_0^s r^{p_2}W_r^{q_2}dr\left(s^{p_1}\left(q_1W_s^{q_1-1}dW_s+\frac{q_1(q_1-1)}{2}W_s^{q_1-2}ds\right)\right.\\
&\quad\left.+W_s^{q_1}\left(p_1s^{p_1-1}ds\right)+0\right)\\
&= \int_0^s r^{p_2}W_r^{q_2}drs^{p_1}q_1W_s^{q_1-1}dW_s+\left[s^{p_1+p_2}W_s^{q_1+q_2}+\int_0^s r^{p_2}W_r^{q_2}drs^{p_1}\frac{q_1(q_1-1)}{2}W_s^{q_1-2}\right.\\
&\quad\left.+\int_0^s r^{p_2}W_r^{q_2}drW_s^{q_1}p_1s^{p_1-1}\right]ds
\end{aligned}$$

For the quadratic variation from above we have

$$d\left\langle .^{p_1}W_.^{q_1}\int_0^. r^{p_2}W_r^{q_2}dr,\,W_.\right\rangle_s=\int_0^s r^{p_2}W_r^{q_2}drs^{p_1}q_1W_s^{q_1-1}ds.$$

In total, we have

$$\begin{aligned}
&d\left(s^{p_1}W_s^{q_1}\int_0^s r^{p_2}W_r^{q_2}drW_s\right)\\
&\quad=(1+q_1)s^{p_1}W_s^{q_1}\int_0^s r^{p_2}W_r^{q_2}drdW_s+\left[s^{p_1+p_2}W_s^{q_1+q_2+1}+\frac{q_1(q_1+1)}{2}s^{p_1}W_s^{q_1-1}\int_0^s r^{p_2}W_r^{q_2}dr\right.\\
&\quad\left.+p_1\int_0^s r^{p_2}W_r^{q_2}drW_s^{q_1+1}s^{p_1-1}\right]ds.
\end{aligned}$$

Rearranging the equation yields the claim. □

Using Lemma A.1 (A.8) in the case $p=1$ and $q=0$ and the skew-symmetry of the commutator we have finally

$$\begin{aligned}
Y_t^2 &= Bt-\frac{1}{2}A^2t+\frac{1}{2}[B,A]\int_0^t W_sds+AW_t-\frac{1}{2}[B,A]\left(tW_t-\int_0^t W_sds\right)\\
&= Y_t^1-\frac{1}{2}A^2t+[B,A]\int_0^t W_sds-\frac{1}{2}[B,A]tW_t.
\end{aligned}$$

Order 3. From now on we will always repeat the steps seen from the derivation of order 2. First, identify all terms using (A.5) and (A.7) and secondly apply Lemma A.1 to remove the iterated (stochastic) integrals by expressions with single Lebesgue integrals.

After using (A.5) and (A.7) we get

$$\begin{aligned}
\sigma_3\left(Y_t^{2,\epsilon,\delta}\right) &= \epsilon A-\frac{1}{2}[\delta B,\epsilon A]t-\frac{1}{12}[[\delta B,\epsilon A],B]t^2-\frac{1}{2}[[\delta B,\epsilon A],\epsilon A]\int_0^t W_sds\\
&\quad+\frac{1}{6}[[\delta B,\epsilon A],\epsilon A]W_t\\
\mu_3\left(Y_t^{2,\epsilon,\delta}\right) &= \delta B-\frac{1}{2}\epsilon^2A^2+\frac{1}{12}[[\delta B,\epsilon A],\epsilon A]t-\frac{1}{2}[[\delta B,\epsilon A],\delta B]\int_0^t W_sds+\frac{1}{2}[\delta B,\epsilon A]W_t\\
&\quad+\frac{1}{3}[[\delta B,\epsilon A],\delta B]tW_t+\frac{1}{12}[[\delta B,\epsilon A],\epsilon A]W_t^2
\end{aligned}$$

Applying Lemma A.1 and collecting all terms yields

$$\begin{aligned}
Y_t^3 &= Y_t^2+[[B,A],A]\left(\frac{1}{2}\int_0^t W_s^2ds-\frac{1}{2}W_t\int_0^t W_sds+\frac{1}{12}tW_t^2\right)\\
&\quad+[[B,A],B]\left(\int_0^t sW_sds-\frac{1}{2}t\int_0^t W_sds-\frac{1}{12}t^2W_t\right).
\end{aligned}$$

References

[1] A.H. Al-Mohy, N.J. Higham, Computing the action of the matrix exponential, with an application to exponential integrators, SIAM J. Sci. Comput. 33 (2) (2011) 488–511, http://dx.doi.org/10.1137/100788860.

[2] G. Barrera, M. Högele, J. Pardo, Cutoff stability of multivariate geometric Brownian motion, 2022, arXiv:2207.01666.

[3] L. Campana, Stochastic Modelling of Non-Spherical Particles in Turbulence (Ph.D. thesis), Université Côte d'Azur et de INRIA, 2022.

[4] C. Cercignani, The Boltzmann Equation and Its Applications, Springer-Verlag, New York, 1988, p. xii+455.

[5] L. Desvillettes, C. Villani, On the trend to global equilibrium in spatially inhomogeneous entropy-dissipating systems: The linear Fokker-Planck equation, Comm. Pure Appl. Math. 54 (1) (2001) 1–42.

[6] M. Di Francesco, A. Pascucci, On the complete model with stochastic volatility by Hobson and Rogers, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 460 (2051) (2004) 3327–3338, http://dx.doi.org/10.1098/rspa.2004.1370.

[7] P.K. Friz, P.P. Hager, N. Tapia, Unified signature cumulants and generalized Magnus expansions, Forum Math. Sigma 10 (2022) http://dx.doi.org/10.1017/fms.2022.20, Paper No. e42, 60.

[8] M.B. Giles, C. Reisinger, Stochastic finite differences and multilevel Monte Carlo for a class of SPDEs in finance, 2012, http://dx.doi.org/10.48550/ARXIV.1204.1442, arXiv.

[9] J. Ibáñez, J.M. Alonso, P. Alonso-Jordá, E. Defez, J. Sastre, Two taylor algorithms for computing the action of the matrix exponential on a vector, Algorithms 15 (2) (2022) http://dx.doi.org/10.3390/a15020048.

[10] K. Kamm, M. Muniz, A novel approach to rating transition modelling via machine learning and SDEs on Lie groups, 2022, arXiv:2205.15699.

[11] K. Kamm, S. Pagliarani, A. Pascucci, On the stochastic Magnus expansion and its application to SPDEs, J. Sci. Comput. 89 (3) (2021) http://dx.doi.org/10.1007/s10915-021-01633-6, Paper No. 56, 31.

[12] P.-L. Lions, On Boltzmann and Landau equations, Philos. Trans. R. Soc. Lond. Ser. A 346 (1679) (1994) 191–204.

[13] W. Magnus, On the exponential solution of differential equations for a linear operator, Comm. Pure Appl. Math. 7 (1954) 649–673, http://dx.doi.org/10.1002/cpa.3160070404.

[14] M. Muniz, M. Ehrhardt, M. Günther, R. Winkler, Higher strong order methods for linear itô sdes on matrix lie groups, BIT Numerical Mathematics 62 (3) (2022) http://dx.doi.org/10.1007/s10543-022-00911-5, 1093-1093.

[15] A. Pascucci, PDE and Martingale Methods in Option Pricing, in: Bocconi & Springer Series, Springer-Verlag, New York, 2011, p. xiv+720.

[16] A. Pascucci, A. Pesce, Backward and forward filtering under the weak hörmander condition, Stochastics and Partial Differential Equations: Analysis and Computations (2022a) http://dx.doi.org/10.1007/s40072-021-00225-7.

[17] A. Pascucci, A. Pesce, On stochastic Langevin and Fokker-Planck equations: The two-dimensional case, J. Differential Equations 310 (2022b) 443–483, http://dx.doi.org/10.1016/j.jde.2021.11.004.

[18] C. Reisinger, Z. Wang, Stability and error analysis of an implicit Milstein finite difference scheme for a two-dimensional Zakai SPDE, 2018, http://dx.doi.org/10.48550/ARXIV.1802.07682, arXiv.

[19] G. Yang, K. Burrage, Y. Komori, P. Burrage, X. Ding, A class of new Magnus-type methods for semi-linear non-commutative Itô stochastic differential equations, Numer. Algorithms 88 (4) (2021) 1641–1665, http://dx.doi.org/10.1007/s11075-021-01089-7.