Parallel optimized sampling for stochastic equations

Bogdan Opanchuk, Simon Kiesewetter and Peter D. Drummond

Centre for Quantum and Optical Science, Swinburne University of Technology, Melbourne, Australia.

Abstract

Stochastic equations play an important role in computational science, due to their ability to treat a wide variety of complex statistical problems. However, current algorithms are strongly limited by their sampling error, which scales proportionate to $1/\sqrt{N_S}$ for $N_S$ samples. In this paper, we obtain a new class of parallel optimized sampling methods for treating stochastic equations. The objective of parallel optimized sampling is to reduce the sampling error in the observables of an ensemble of stochastic trajectories. This is achieved through calculating a finite set of observables — typically statistical moments — in parallel, and minimizing the errors compared to known target values. The algorithm is both numerically efficient and unbiased, in the sense that it converges for large sample number. Importantly, it does not increase the errors in higher order moments, and generally reduces such errors as well. The same procedure is applied both to initial ensembles and to changes in a finite time-step. Results of these methods show that errors in initially optimized moments can be reduced to the machine precision level, typically around $10^{-16}$ in current hardware. For nonlinear stochastic equations, sampled moment errors during time-evolution are larger than this, due to error propagation effects. Even so, we provide evidence for error reductions of up to two orders of magnitude in a nonlinear equation example, for low order moments, which is a large practical benefit. The sampling error reduction scales as $1/\sqrt{N_S}$ for non-optimized moments, as expected, but with the advantage of a very much smaller prefactor than for standard, non-optimized methods.

Keywords: stochastic, optimization, parallel

1. Introduction

Stochastic differential equations (SDEs) play a universal and important role in many disciplines requiring quantitative modeling [1–9]. Their practical advantage is that, when used to solve large statistical problems, the ability to randomly sample greatly reduces the complexity of treating the full distribution function. They are employed for treating problems ranging from statistical physics, chemistry and engineering through to economics, biology and financial modeling. As a result of this large field of applications, there is substantial literature on the algorithms used to solve them [10–15].

This utility is not without a price. Such numerical algorithms typically utilize many independent trajectories. Any statistical result will therefore have a sampling error $\epsilon_{S}$ that scales as $1/\sqrt{N_S}$ for $N_S$ independent samples, giving an error that only decreases slowly with the total computation time. Numerical algorithms generally aim to reduce the truncation error $\epsilon_T$ due to the discretization in time, giving an error of order $(\Delta t)^{-p}$ for methods of order $p$. Yet reducing the truncation error cannot reduce the total error to less than the sampling error. Since the sampling error is often a large part of the total error one has to deal with, reducing the truncation error has a limited effectiveness.

Here we propose a new class of SDE algorithms called parallel optimized sampling or POS algorithms for short. This approach can provide extremely useful improvements over independent random sampling methods. The method locally eliminates sampling errors over a finite set of moments, up to numerical precision. Higher order moment errors are reduced also. Global sampling error improvement over a finite time interval is not as large as this, due to error propagation effects, but is still very substantial. Other techniques that can also improve sampling errors include the use of low discrepancy (or quasi-random) sequences [16], and an extrapolated hierarchy of simulations [17], which we do not treat here.

Preprint submitted to Journal of Computational Physics

February 26, 2015
To achieve our results, we introduce a modification of the standard independent stochastic trajectory approach. A set of \( N_s \) trajectories is solved \textit{in parallel} using an optimized algorithm. This gives an improvement that depends on the order of the statistical moment being calculated. The present method is a highly practical one, since the run time is still only linear in \( N_s \), which means that the error reductions are of real value. If the run time scaled as \( N_s^3 \), and errors were reduced to \( 1/N_s \), the reductions would not be so useful, since the total computational overhead at fixed error would not change.

There is a close relationship between the present approach and methods of moment hierarchies [5] or cumulant expansions. These existing cumulant based methods have a major drawback. One must close the infinite hierarchy with an arbitrary ansatz at some finite order. If this is incorrect, the method will fail [18]. In fact, such cumulant hierarchies cannot in general be truncated consistently [19]. By contrast, our method requires no ansatz; it gracefully reduces in a consistent way to the usual, unbiased stochastic method at large orders. It can be thought of as a way to unify moment hierarchy and stochastic methods.

We focus on simple one-dimensional cases here for simplicity. Extensions of the POS method to cases of higher-order convergence in time will be treated elsewhere. However, as one example of potential applications, sampling errors can play a major role in quantum simulations using phase-space representations. The difficulty is that distribution tails may only be weakly bounded [20]. In such cases, sampling error can determine whether a quantum simulation is feasible or not. One approach to solving this problem is using weighted trajectory methods [21]. Another method, which was successfully used in treating long-range order in the fermionic Hubble model, is to impose global conservation laws [22]. The present approach has a more general applicability than either of these methods.

The organization of the paper is as follows. In Section 2, we analyze the stochastic errors that can arise in computational methods for stochastic differential equations. In Section 3, sampling errors are reduced in initial observables by introducing a new type of stochastic noise generator. In Section 4, this approach is extended to dynamical optimization of stochastic equation algorithms. In Section 5, we demonstrate the performance of these methods for a number of one-dimensional linear and nonlinear cases, showing that both local and global sampling error performance is substantially improved. Section 6 gives our summary and conclusions.

2. Stochastic integration error

Stochastic integration involves both truncation errors due to finite step-size, and errors due to finite samples. In this section we analyze the performance of independent SDE methods by considering the overall resources needed to solve the equations for some given error. We show that increasing the convergence order is a less effective way to reduce errors than one might expect, because of a trade-off between the resources needed to reduce both truncation and sampling errors. As large ensembles are required to reduce sampling errors, we find that higher-order methods for SDEs are not as useful as in ODE integration.

In the rest of this paper, we show how to overcome this by using algorithms that reduce sampling errors.

2.1. Distributions and observables

The observables of a probability distribution function \( P(t,x) \) are the fundamental objects of interest in stochastic calculations. These are probability-weighted integrals of functions \( o_m (x) \) over \( x \):

\[
\langle o_m \rangle = \int dx o_m (x) P(t,x).
\]

In general, \( dx \) is a Euclidean measure over a \( d \)-dimensional real or complex space of variables \( x \). The numerical examples given here will focus on moments of one-dimensional distributions with \( o_m(x) = x^m \), giving a vector whose error should be minimized:

\[
o = \{o_1, \ldots, o_M \}.
\]
With sampling methods, the time-evolution is solved by generating a number \( N_S \) of independent sample paths \( \mathbf{x}^{(n)}(t) \). This procedure is equivalent to approximating the true distribution by the sampled distribution defined as

\[
P_S(t, \mathbf{x}) = \frac{1}{N_S} \sum_{n=1}^{N_S} \delta \left( \mathbf{x} - \mathbf{x}^{(n)}(t) \right).
\]

(3)

We note that the initial value problem consists of defining an initial static distribution \( P(x) \) at \( t = t_0 \), and then sampling it with initial points \( \mathbf{x}^{(m)}(t_0) \). For dynamics, the sampled paths are defined at \( N_T \) discrete times \( t_i \), with a spacing \( \Delta t \), such that exact ensemble average values are recovered in the double limit of infinite sample and zero step-size:

\[
\langle o_m(t_i) \rangle = \lim_{\Delta t \to 0} \lim_{N_S \to \infty} \frac{1}{N_S} \sum_{n=1}^{N_S} o_m \left( \mathbf{x}^{(n)}(t_i) \right).
\]

(4)

These paths are obtained by numerical algorithms which propagate \( \mathbf{x}^{(n)}(t_i) \) to \( \mathbf{x}^{(n)}(t_{i+1}) \), with a truncation error vanishing as \( \Delta t \to 0 \). If we call \( N_S \) the sample number, the phase-space is made up of a set of distinct samples or trajectories. These form an extended vector of parallel trajectories,

\[
\mathbf{X} = \left( \mathbf{x}^{(1)T} \cdots \mathbf{x}^{(N_S)T} \right)^T,
\]

(5)
of size \( N_S d \). It is convenient to define the sampled observables as a function \( \mathcal{O} \) of the extended vector of sampled trajectories,

\[
\mathcal{O}_m(\mathbf{X}) = \frac{1}{N_S} \sum_{n=1}^{N_S} o_m \left( \mathbf{x}^{(n)} \right).
\]

(6)

Throughout this paper, we will use the standard notation that all extended vectors or matrices involving a dimension of the order of the sample size \( N_S \) are written in upper case like \( \mathbf{X} \), while objects that do not depend on the sample size are written in lower case.

### 2.2. Stochastic equations and distributions

To explain the problem of interest here, we give a detailed analysis of a stochastic differential equation or SDE. The underlying distribution function \( P(t, \mathbf{x}) \) satisfies the Fokker-Planck equation [6],

\[
\frac{\partial P(t, \mathbf{x})}{\partial t} = -\sum_i \frac{\partial}{\partial x_i} a_i(\mathbf{x}) + \frac{1}{2} \sum_{ij} \frac{\partial^2}{\partial x_i \partial x_j} d_{ij}(\mathbf{x}) P(t, \mathbf{x}) .
\]

(7)

We wish to minimize the errors for solving the corresponding Itô SDE [4][6], given in a standard form by:

\[
d\mathbf{x} = \mathbf{a}(\mathbf{x}) \, dt + \mathbf{b}(\mathbf{x}) \, dw,
\]

(8)

where \( \mathbf{d} = \mathbf{b} \mathbf{b}^T \), and the Gaussian distributed real noise \( dw \) has correlations given by:

\[
\langle dw_i dw_j \rangle_\infty = \delta_{ij} dt
\]

\[
\langle dw_i \rangle_\infty = 0.
\]

Here the \( \langle \ldots \rangle \) notation means ensemble average, so that

\[
\langle \mathbf{x} \rangle_{N_S} = \frac{1}{N_S} \sum_{n=1}^{N_S} \mathbf{x}^{(n)} ,
\]

(10)

with \( \langle \ldots \rangle_\infty \) being the infinite ensemble limit.
2.3. Sampling and error criteria

To evaluate the computational error, we must define an error criterion relevant to the entire sample ensemble $X$. In practical terms, this means one must calculate errors in each required observable, rather than just the errors in a single trajectory. By minimizing the total error over a finite number of samples, one should be able to arrive at the most efficient that utilizes a given computational resource.

To quantify the error we will use $L^2$ norm $\|P\|_W = \sqrt{(P, P)} \geq 0$. The integration error of the sampled, calculated stochastic distribution $P_S$ relative to the exact distribution $P_E$ is then defined as

$$\epsilon = \|P_S - P_E\|_W,$$  \hfill (11)

which depends on the choice of distribution norm.

The computed averages, $\langle o_m \rangle_{P_S}$, are obtained through sampling, which means that typically one is interested in calculating specific observables with minimum error. To evaluate the computational accuracy relevant to these observables of interest, we choose a particular set of observable quantities, $o_m$, for $m = 1, \ldots, M$, to define a distribution norm:

$$\|P\|^2_W = \sum_{m=1}^{M} W_m |\langle o_m \rangle_{P_S}|^2.$$  \hfill (12)

Here $W_m$ is the relative weight assigned to observable $m$. We assume the error is evaluated at a fixed time $t$, otherwise one may wish to minimize the errors at each of a set of sample times. In this paper, the observables are a finite set of one-dimensional moments, $o_m \equiv x^m$, although other measures are certainly possible.

With this definition, the total error is

$$\epsilon_P = \sqrt{\sum_{m=1}^{M} W_m |\langle o_m \rangle_{P_S} - \langle o_m \rangle_{P_E}|^2},$$  \hfill (13)

The integration error $\epsilon$ clearly must be calculated from the entire vector of stochastic trajectories $x$. The task of defining an optimal stochastic integration method is to obtain $x(t)$ with a procedure that minimizes the total error $\epsilon$ relative to some computational resource. Using the definition that

$$\mu_m = \langle o_m \rangle_{P_E},$$  \hfill (14)

we see that:

$$\epsilon_P = \sqrt{\sum_{m=1}^{M} W_m |\bar{o}_m (X) - \mu_m|^2}.$$  \hfill (15)

Although different to the usual probability norms, we will use the above definition throughout this paper, since it corresponds to the operational requirements of a norm for a sampled distribution.

2.4. Optimizing the sample number

What is the optimal strategy to solve this SDE, given fixed total computational resources and a goal of minimizing the total error? With independent paths, the computational resource utilized is proportionally to $N = N_S N_T$. One can invest total processing time either in reducing the time-step, or in increasing the number of samples, but reducing the step-size means using less samples, and vice-versa.

To understand these issues quantitatively, we first consider the optimal strategy for error-reduction with traditional methods that involve using independent sample paths. We will show that it is only important to reduce the step-size or discretization error to the point where it is some fraction of the sampling error.

Consider the total error for $N_T$ time-steps with a stochastic integration method of global order $p$, that utilizes $N_S$ independent sample paths. From the central limit theorem, the global sampling error $\epsilon_S$ due to the use of a finite sample, is

$$\epsilon_S = \sigma N_S^{-1/2},$$  \hfill (16)
where $\sigma$ is the standard deviation of the measured quantity, and the same type of scaling holds for sums over errors in multiple observables.

There are additional errors $\epsilon_T$ over a fixed time interval $T = N_T \Delta t$, due to the finite time-step $\Delta t$. If the local error of one time-step is $\epsilon_{\Delta t}$, then at best $\epsilon_T \approx N_T \epsilon_{\Delta t}$. The discretization order $p$ is therefore defined so that the global time-step truncation error $\epsilon_T$ scales as

$$\epsilon_T = c N_T^{-p} \propto \Delta t^p,$$

where $c$ is an algorithm-dependent constant that depends on the set of computed observables.

Since the two error sources are independent, the overall error is given approximately by a sum of two terms:

$$\epsilon = \epsilon_T + \epsilon_S = c N_T^{-p} + \sigma N_S^{-1/2}.$$  \hspace{1cm} (18)

We wish to constrain the total processor time — parallel or otherwise — so that $N_S N_T = N$ is bounded. The total CPU time is then $T_{CPU} = N \tau$, if one step in time takes a real time duration $\tau$, and other overheads are negligible.

Next, consider how to optimize the tradeoff between the truncation and sampling errors. Letting $\tilde{\epsilon} = c N^{-p}$, and keeping $N$ fixed, one obtains:

$$\epsilon = \tilde{\epsilon} N^p + \sigma N_S^{-1/2}.$$  \hspace{1cm} (19)

This is minimized by choosing a combination of time-step and sample number such that:

$$N_S = N \frac{2^p}{2 p c} \left[ \frac{\sigma}{2 p c} \right]^\frac{1}{p+1},$$

$$N_T = N \frac{2^p}{2 p c} \left[ \frac{\sigma}{2 p c} \right]^{-\frac{2}{p+1}}.$$  \hspace{1cm} (20)

The total error is then:

$$\epsilon = \left[ \frac{c \sigma^{2p}}{2 p N^p} \right] \frac{1}{p+1} \left[ 1 + \frac{1}{2p} \right].$$  \hspace{1cm} (21)

2.5. Effective integration order

From the analysis above, the best that one can do to reduce errors is to obtain a scaling of $\epsilon \propto N^{-p_{\text{eff}}}$, where the effective order $p_{\text{eff}}$ is

$$p_{\text{eff}} = \frac{p}{2p+1}.$$  \hspace{1cm} (22)

This is because a higher order integration technique has no effect on the sampling error, which eventually dominates the error calculation. The best effective order, even with $p = \infty$, is $p_{\text{eff}} = 1/2$. Given that stochastic higher order algorithms are complex and slow, higher $p$ values are not always an advantage.

This optimal effective order requires the use of an optimal ratio of samples to steps, which is:

$$\frac{N_S}{N_T} = N \frac{2^p}{2 p c} \left[ \frac{\sigma}{2 p c} \right]^{\frac{4}{p+1}}.$$  \hspace{1cm} (23)

This ratio is difficult to calculate a-priori, since these constants are not usually known in advance. One can estimate the sampling error numerically by measuring the sample standard deviation $\sigma$, which is known from the statistics of the simulation. The truncation error constant $c$ is also measurable by changing the step-size. Thus, both $c$ and $\sigma$ can be measured in software. An optimum point is reached when the ratio between discretization and sampling error is given by the extremely simple result that:

$$\frac{\epsilon_T}{\epsilon_S} = \frac{1}{2p}.$$  \hspace{1cm} (24)
In summary, it is only productive to reduce the step-size to the point where the discretization error of \(1/2p\) of the sampling error. After this point is reached, one is better off to use more samples. Another way to describe this is that the least computational cost for error \(\epsilon\) is

\[
N \propto \epsilon^{-1/p_{\text{eff}}} = \epsilon^{-\frac{2p+1}{p}}.
\]

(25)

There are multi-scale methods \[17\] that can achieve a cost of \(N \propto \epsilon^{-2 (\log \epsilon)^2}\), which is a useful further improvement.

In summary, from the traditional analyses of ODE integration, one might expect that the error would be reduced extremely rapidly with a higher order method. This is not the case with an SDE. Because the independent sampling error varies slowly with resource, this becomes the dominant error with increasing resources.

Our conclusion is that sampling error strongly limits the effectiveness of independent sampling methods for SDEs.

3. Parallel optimized sampling

We now describe a family of parallel optimized sampling (POS) methods which locally eliminate sampling errors for a finite set of observables \(o\), thus improving computational efficiency. As described above, for \(N_S\) samples, there is a \(N_Sd\)-dimensional extended vector of stochastic variables, \(X = (x^{(1)}^T \cdots x^{(N_S)}^T)^T\), which is optimized.

Both initial samples and the resulting stochastic trajectories are optimized in parallel, thus making use of increasingly parallel hardware capabilities. Our approach is to assume that a set of stochastic observables are known at a given time \(t\), and to choose a noise ensemble to give correct observables at time \(t + \Delta t\). Overall convergence is then achieved by taking a large \(N_S\) limit.

A POS algorithm must therefore meet three essential requirements:

1. Equations for a finite set of \(M\) observables are exactly satisfied both initially, and for local changes in time to a given order in \(\Delta t\).
2. Non-optimized observables still have their correct values to the same order in \(\Delta t\) in the limit of \(N_S \to \infty\).
3. The computational cost of the algorithm should be no worse than \(O(N_S)\).

3.1. Observables as moments

The goal of the algorithm is that the observable moments are obtained with minimal error. For efficiency, the resulting calculation complexity should be linear in \(N_S\).

If the observables chosen are moments, then, since higher order moments depend on lower order ones, these will experience a degree of optimization as well, even though not directly optimized. POS methods can also be combined with higher-order time-step methods, but here we focus on sampling error reduction for ease of explanation.

There is an analogy between this approach and the method of moment hierarchies in statistical physics \[5\]. The difference is that rather than an arbitrary truncation of the moment hierarchy, the higher order moments are estimated in an unbiased way via sampling.

3.2. Initial conditions

Before discussing the performance of POS algorithms, we need to address an issue arising from the known initial condition for the SDE, with \(x\) having a distribution \(P(x)\) at \(t = t_0\). When we derive the optimization conditions, one of the assumptions will be that the moments of \(x\) are already optimized at the start of each time step, that is \(\langle o_m(t) \rangle_{N_S} = \mu_m(t)\) for \(m = 1 \ldots M\). At every step of the SDE integration, if we obtain the optimized stochastic vector from the previous step, then, since we optimize the differentials of the moments, the resulting stochastic variables will be optimized as well as possible, apart from any time-step errors, which is a separate issue.
But how do we ensure the initial set of moments of the distribution $P(x)$ is optimized at the first integration step?

This is solved by optimizing the initial distribution of sampled variables. We call this static POS, as it does not involve dynamical time-evolution. In other words, we wish to sample the initial values for the trajectories so that $M$ chosen observables are equal to their known exact values, that is,

$$\bar{o}(X) = \mu \equiv \langle o \rangle_\infty,$$

where $\mu$ is an $M$-dimensional vector. That is, we wish to set $\epsilon_P = \epsilon_P(t_0) = 0$, where $\epsilon_P$ is the total error in the initial distribution, using an observable-based error measure.

This is a classic example of nonlinear, multidimensional root-finding. It is a challenge to solve these equations efficiently, with resources no greater than $O(N_S)$, and with minimal changes to the original sampled estimate for $X(t_0)$, which is labelled $X(0)$. The numerical problem can be accurately solved, thus numerical precision issues play a role. Here we use an iterative, modified Newton-Raphson method for the solution. Other techniques are available as well.

In an iterative Newton-Raphson approach, we use stochastic methods for the first estimate, $X(0)$. At each iteration, we obtain the next estimate $X(i+1)$, which has a better fit to the required moments, as $X(i+1) = X(i) + \Delta(i)$, where $\|\Delta(i)\| \ll \|X(i)\|$, and $\|\cdot\|$ denotes the Euclidean norm.

Expanding to first order in $\Delta(i)$, the conditions we require are:

$$\bar{o}(\text{opt}) = \mu = \bar{o}(X(i)) + J(X(i)) \Delta(i) + O(\Delta^2(i)),
$$

where $J$ is a Jacobian of $\bar{o}$:

$$J_{mn}(X) = \frac{\partial \bar{o}_m(X)}{\partial X_n}.$$

In matrix form, the iteration equation is therefore

$$J(i) \Delta(i) = \mu - \bar{o}(i),$$

where we use the obvious notation that $\bar{o}(i) \equiv \bar{o}(X(i))$ and $J(i) \equiv J(X(i))$.

Each step of the iterative solution is underdetermined, since the relevant matrices are not square, and hence not invertible. Such problems are relatively common in optimization, and here we analyze two different techniques to solve them.

### 3.2.1. One-dimensional example

We will use as an illustrative example the situation when the observables are moments of a one-dimensional stochastic equation. In this example $o_m = x^m$, so the conditions are:

$$[X^{(\text{opt})}^m] \cdot 1 = N_S \mu_m$$

$$= X^m(i) \cdot 1 + mX^{m-1}(i) \cdot \Delta(i) + O(\Delta^2(i)),$$

where the powers of matrices and vectors are understood in the element-wise (Hadamard) sense (e.g., $X^2 \equiv X \circ X$). Hence, in this case the observable vector and matrix of derivatives are:

$$\bar{o}(X) = \frac{1}{N_S} \sum_{n=1}^{N_S} \left( \begin{array}{c} x^{(n)} \\ \ldots \\ [x^{(n)}]^M \end{array} \right),$$

$$J(X) = \frac{1}{N_S} \left( \begin{array}{ccc} 1 & \cdots & 1 \\ \cdots & \cdots & \cdots \\ M [x^{(1)}]^{M-1} & \cdots & M [x^{(N_S)}]^{M-1} \end{array} \right).$$
3.3. Static pseudo-inverse iterations

The simplest method is to use a pseudo-inverse to solve the iteration equation (29). This has another advantage, since the pseudo-inverse has the property that it generates a least-squares solution, with minimal changes, as required for this algorithm.

Thus, we have the iterative procedure

\[ X_{(i+1)} = X_{(i)} + J_{(i)}^+ (\mu - \bar{o}_{(i)}) . \]

(32)

where \( J^+ \) indicates the pseudo-inverse of the \( M \times N_S \) Jacobian \( J \).

This pseudo-inverse method is simple, general and effective. It satisfies our requirements given above. However, standard pseudo-inverse software using singular value decomposition (SVD) methods has drawbacks. It is relatively slow, can have stability problems and may not be available on all programming platforms.

3.4. Static matrix inversion iterations

To improve efficiency and overcome problems with the use of general purpose pseudo-inverses, there is a faster alternative method for solving (29).

The pseudo-inverse is a limit. If \( J^\dagger \) denotes the conjugate transpose of the matrix \( J \), then:

\[ J^+ \equiv \lim_{\alpha \to 0} J^\dagger \left( JJ^\dagger + \alpha I \right)^{-1} . \]

(33)

Hence, provided \( u \equiv JJ^\dagger \) is invertible, one can directly obtain that \( J^+ = J^\dagger u^{-1} \). It is important to note here that \( u \) is only an \( M \times M \) matrix, regardless of how many samples were used originally.

This gives as many unknown coefficients as there are equations. As a result, we only need solve \( M \) linear equations in \( M \) unknowns, for which there are efficient techniques, resulting in the iteration procedure

\[ X_{(i+1)} = X_{(i)} + J_{(i)}^\dagger u_{(i)}^{-1} (\mu - \bar{o}_{(i)}) . \]

(34)

Here the matrix \( u \) being inverted is an \( M \times M \) positive-semidefinite matrix. We use the lower case for it to indicate that it does not take an extended dimensionality. It is generally invertible and relatively small compared to the sample size, resulting in a highly efficient optimization algorithm. Numerical investigations show that this simple method is convergent for large sample number \( N_S \), which corresponds to cases where the initial sample has moments close to their ideal values.

As the stopping condition in the numerical examples given later, we choose the condition that the changes are small relative to the modulus of the current \( X \) value, i.e. \( \| \Delta_{(i)} \|/\|X_{(i)}\| < \eta = 10^{-8} \), where \( \eta^2 \) is the numerical machine precision, as previously. An iteration limit of \( I_{\text{max}} = 50 \) was used, although as we see later, the typical iteration number is much lower than this.

We note that this simplified method is only useful when \( u \) is invertible, otherwise the full SVD techniques or other more robust algorithms should be used. As often the case with such numerical methods, performance can be improved through linear scaling or translation operations.

3.4.1. One-dimensional case

As an equivalent way to understand this approach in the one-dimensional case, we expand \( \Delta \), which is the change in \( X \), using the derivative matrix and a vector of changes in observables, \( \delta \). With observables as moments, this is a series in \( X^\dagger \) up to order \( M-1 \), with \( M \) coefficients written as a vector \( \delta \). The expansion is given by:

\[ \Delta = J^\dagger \delta = \frac{1}{N_S} \sum_{m=1}^{M} m (X^\dagger)^{m-1} \delta_m . \]

(35)
Defining a square, positive definite matrix:

\[ \mathbf{u}_{(i)} = \mathbf{J}_{(i)} \mathbf{J}_{(i)}^\dagger \]  

the basic iteration equation (29) becomes:

\[ \mathbf{u}_{(i)} \delta_{(i)} = \mu - \bar{o}_{(i)}. \]  

Since the solution for \( \delta_{(i)} \) requires the inversion of \( \mathbf{u}_{(i)} \), this leads to the same final algorithm that is defined above, in Eq (34).

4. Parallel dynamic optimization

We now describe how to extend this initial optimization algorithm to treat dynamical optimization of moments during stochastic time evolution of \( \mathbf{x} \). This involves stochastic noise terms and deterministic drift terms.

4.1. Euler-Maruyama algorithm

For simplicity, we only treat the optimization of the Euler-Maruyama integration scheme for an SDE in Itô form [13], which has truncation order \( p = 1 \) for convergence of moments. This is a discrete expression of the standard Ito SDE, so that for a finite step in time \( \Delta t \),

\[ \Delta \mathbf{x} = \mathbf{a} \Delta t + \mathbf{b} \Delta \mathbf{w}. \]  

Next, we introduce \( \mathbf{A} = (\mathbf{a}^T (\mathbf{x}^{(1)}) \cdots \mathbf{a}^T (\mathbf{x}^{(N_S)}))^T \) as an extended vector of drift coefficients, \( \mathbf{B} = \text{diag} (\mathbf{b} (\mathbf{x}^{(1)}) \cdots b (\mathbf{x}^{(N_S)})) \) (a block-diagonal matrix with \( \mathbf{b} (\mathbf{x}^{(i)}) \) elements) as an extended matrix of noise terms, and \( \Delta \mathbf{W} = \left( (\Delta \mathbf{w}^{(1)})^T \cdots (\Delta \mathbf{w}^{(N_S)})^T \right)^T \) as an extended vector of noises. The parallel sample vector \( \mathbf{X} \) will satisfy the following vector SDE, now of dimension \( N_S d \):

\[ \Delta \mathbf{X} = \mathbf{A} \Delta t + \mathbf{B} \Delta \mathbf{W}. \]  

One requires that the sampled observables are given as closely as possible by the known infinite ensemble results, that is,

\[ \bar{o} (\mathbf{X} + \Delta \mathbf{X}) - \bar{o} (\mathbf{X}) = \langle \Delta (\mathbf{o} (\mathbf{x})) \rangle_{\infty} + \mathcal{O} (\Delta t^2), \]  

where \( \langle \Delta (\mathbf{o} (\mathbf{x})) \rangle_{\infty} \) is the ideal observable change for an infinite ensemble to order \( \Delta t \). This is given either by an application of a single-step approximation to the Fokker-Planck equation (7) followed by partial integration, or equivalently by an application of Itô’s formula [6] to the stochastic equation (8):

\[ \langle \Delta o_m (\mathbf{x}) \rangle_{\infty} = \delta_m \left( \frac{\partial o_m}{\partial x_i} a_i + \frac{1}{2} \frac{\partial^2 o_m}{\partial x_i \partial x_j} d_{ij} \right) \Delta t + \mathcal{O} (\Delta t^2) \]  

In order to be able to calculate these moments in practice we must make another approximation, since we only know the required averages for a finite ensemble of trajectories. From the central limit theorem, if the individual variances are finite then:

\[ \delta_m \left( \frac{\partial o_m}{\partial x_i} a_i + \frac{1}{2} \frac{\partial^2 o_m}{\partial x_i \partial x_j} d_{ij} \right) \Delta t + \mathcal{O} \left( \frac{1}{\sqrt{N_S}} \right). \]  

In some cases the equality is exact; see the discussion of the error introduced by this approximation in Section 4.4.
4.1.1. One-dimensional case

In the one-dimensional moment-based case, this result reduces to:

\[
\langle \Delta (x^m) \rangle_\infty \approx m \left( x^{m-1} a + \frac{m-1}{2} x^{m-2} b^2 \right) N_S \Delta t.
\]

These methods can also be extended to higher orders in the step-size, but here we treat the equations up to the first order in \( \Delta t \), and truncate higher orders for simplicity.

At this stage, we point out the existence of more than one possible strategy for satisfying the moment equations. Since each has its own distinct advantages and disadvantages, two particular approaches will be treated here. They are described in the following two subsections.

Both of the resulting POS methods have the following useful features:

- Uniqueness — the matrix iteration equations have unique results.
- Parameter independence — only the observables that are optimized are specified, not an arbitrary parameter.
- Linear complexity — the time taken scales as \( N_S \), since the inner products only require \( N_S \) operations.

4.2. Combined optimization

In combined optimization, all time orders in the sampled moments are combined together, to give a moment estimate in an analogous form to that treated previously for the initial conditions. The advantage of this approach is its ease of implementation and simplicity.

However, unlike the case of the initial distribution, we typically do not know the infinite ensemble average moments exactly, that is, to all orders in \( \Delta t \). Hence, in this approach the generated probability distribution differs from the stochastic distribution by terms of order \( \Delta t^2 \), even in the infinite ensemble limit.

Hence, we use the parallel SDE to provide the initial estimate for the stochastic trajectory, prior to error optimization, of:

\[
X(0) (t + \Delta t) = X(t) + A(t) \Delta t + B(t) \Delta W.
\]

Obtaining the improved trajectory estimates is then essentially identical to the static requirement given already. We can write the moment requirement in the form (26):

\[
\bar{o} (X(t + \Delta t)) = c, \quad (45)
\]

where \( c \) is a vector of ideal observables, and \( X(t + \Delta t) = X(0) (t + \Delta t) + \Delta \). Since we do not know these “ideal” averages over infinite number of trajectories, we approximate \( c \) by assuming we know all the moments exactly at the start of the time interval based on knowing \( X(t) \). Changes in moments are estimated up to terms of order \( \Delta t \) using the approximations (41) and (42) so that, in general:

\[
c_m = \left\langle o_m + \left[ \frac{\partial o_m}{\partial x_i} a_i + \frac{1}{2} \frac{\partial^2 o_m}{\partial x_i \partial x_j} d_{ij} \right] \Delta t \right\rangle_{N_S}.
\]

In the one-dimensional example case, from Eq. (43),

\[
c_m = \left\langle x^m + m \left[ x^{m-1} a + \frac{m-1}{2} x^{m-2} b^2 \right] \Delta t \right\rangle_{N_S}.
\]

At the \( i \)-th step, we assume that \( X(i+1) (t + \Delta t) = X(i) (t + \Delta t) + \Delta(i) \), so that

\[
\bar{o} (X(i+1) (t + \Delta t)) = \bar{o} (X(i) (t + \Delta t)) + J(i) \Delta(i) + O(\Delta^2(i)).
\]
The iterative equations to be solved are:

\[ J^{(i)} \Delta^{(i)} = c - \bar{o}^{(i)}. \]  

(49)

Here we define the matrices \( J^{(i)} \) as in Eq. (31). The moment equation is then solved iteratively using the techniques outlined above, that is, either by pseudo-inverse iterations, or more efficiently by the matrix inversion method. With pseudo-inverses, the iteration equations are then simply:

\[ X^{(i+1)} = X^{(i)} + J^{(i)} \left( c - \bar{o}^{(i)} \right). \]  

(50)

For cases where \( u = JJ^\dagger \) is invertible, we can define, just as in the static case:

\[ X^{(i+1)} = X^{(i)} + J^{(i)} u^{-1} \left( c - \bar{o}^{(i)} \right). \]  

(51)

Numerical results using this method are given in Section 5.2.

4.3. Individual optimization

In the individual optimization approach, both the finite ensemble moments and the infinite ensemble moments are calculated to the same order in \( \Delta t \), which allows us to optimize each order in time individually. This strategy permits a clear separation of ensemble and time-step errors. For this purpose, it is convenient to define an effective noise term,

\[ \Delta V = B \Delta W. \]

This is optimized to give the final change in \( X \).

From the moment equations of Eq. (40) and (41), one has the required observable changes to order \( \Delta t \):

\[
\sum_n \frac{\partial \bar{O}_n}{\partial X_n} \Delta V_n (X) + \frac{1}{2} \sum_{n,p} \frac{\partial^2 \bar{O}_n}{\partial X_n \partial X_p} \left[ \Delta V_n (X) \Delta V_p (X) - \Delta t D_{np} (X) \right] = e_m = \mathcal{O} (\Delta t^2) \]

(52)

Here, since we wish to set every moment error to zero over a finite set of moments, it is convenient to define an error vector, \( \mathbf{e} = (e_1, \ldots, e_M)^T \). For example, in the one-dimensional case, from Eq. (40) the extended diffusion matrix is simply diagonal, and one has that:

\[
\frac{m}{N_S} \sum_n \left[ X_n^{m-1} \Delta V_n + \frac{m-1}{2} X_n^{m-2} (\Delta V_n^2 - D_{nn} \Delta t) \right] = e_m = \mathcal{O} (\Delta t^2),
\]

(53)

The error terms can be viewed as two distinct “mean” and “variance” conditions analogous to those in Eq. (9), and are equal to zero in the limit of an infinite ensemble. The error requirement is satisfied provided two individual conditions are met, representing terms of order \( \sqrt{\Delta t} \) and \( \Delta t \) respectively:

\[
\mathbf{e}^{(1)} = J \Delta \mathbf{V}^{\text{opt}} = 0
\]

\[
\mathbf{e}^{(2)} = \frac{1}{2} \mathbf{H} : \left( \Delta \mathbf{V}^{\text{opt}} \right)^T \Delta \mathbf{V}^{\text{opt}} - D \Delta t = 0,
\]

(54)

where "\( : \)" stands for double contraction, so \( (\mathbf{H} : \mathbf{V V}^T)_i \equiv \sum_{jk} \mathcal{H}_{ijk} V_j V_k \). Here, we define \( J \) as previously, and \( \mathbf{H} \) as a Hessian tensor:

\[
\mathcal{H}_{mn} (X) = \frac{\partial^2 \bar{O}_m (X)}{\partial X_m \partial X_n}.
\]

(55)

In the one-dimensional moment-based example, the results are simpler. Due to the absence of cross-derivative terms, the second derivative tensor becomes a matrix,

\[
\mathbf{H}_{mn} (X) = \frac{\partial^2 \bar{O}_m (X)}{\partial X_m \partial X_n} = m \frac{m-1}{N_S} X_n^{m-2}.
\]

(56)
The variance condition then becomes:

\[ e^{(2)} = \frac{1}{2} H \left( \Delta V^{(\text{opt})} \circ \Delta V^{(\text{opt})} - D \Delta t \right) = 0, \quad (57) \]

where \( D_n \equiv D_{nn} \).

These are nonlinear equations, and to solve them we implement an iterative approach. For a large ensemble, the optimizing equations are nearly satisfied to zero-th order, up to errors of order \( 1/\sqrt{N_S} \). Hence, it is efficient to iteratively solve the equations by linearization, giving a Newton-Raphson approach similar to the combined method given above.

Defining the difference \( \Delta (i) \equiv \Delta V^{(\text{opt})} - \Delta V^{(i)} \), the separate conditions are linearized by assuming that \( \| \Delta (i) \| \ll \| \Delta V \| \), to give

\[ \frac{1}{2} \mathcal{H} : \left( \Delta V^{(i)} \Delta V^{T}_{(i)} + \Delta (i) \Delta V^{T}_{(i)} + \Delta V^{(i)} \Delta^{T}_{(i)} - D \Delta t \right) = 0. \quad (58) \]

Noting that \( \mathcal{H} \) is symmetric with respect to its last two indices (\( \mathcal{H}_{mnp} = \mathcal{H}_{mpn} \)), we can rewrite the terms with \( \Delta (i) \) in the second equation as

\[ \frac{1}{2} \mathcal{H} : \left( \Delta (i) \Delta V^{T}_{(i)} + \Delta V^{(i)} \Delta^{T}_{(i)} \right) \equiv \left( \mathcal{H} \Delta V^{(i)} \right) \Delta (i), \quad (59) \]

which allows us to concatenate the two systems into a single matrix equation

\[ \tilde{J} (\Delta V^{(i)}) \Delta (i) = \tilde{R} (\Delta V^{(i)}). \quad (60) \]

Here the doubly extended matrix \( \tilde{J} \) is:

\[ \tilde{J} = \left( \begin{array}{c} J \\ \mathcal{H} \Delta V \end{array} \right). \quad (61) \]

and the remainder vector \( \tilde{R} (\Delta V) \) is defined as:

\[ \tilde{R} (\Delta V) = \left( \begin{array}{c} -J \Delta V \\ \frac{1}{2} \mathcal{H} : \left( D \Delta t - \Delta V^{(i)} \Delta V^{T}_{(i)} \right) \end{array} \right). \quad (62) \]

Starting with \( i = 0 \), we now take \( \Delta V^{(i)} \) and solve these equations iteratively to obtain \( \Delta V^{(i+1)} = \Delta (i) + \Delta V^{(i)} \). To obtain a linear solution for each iterative step, we can use a pseudo-inverse as before, so that:

\[ \Delta V^{(i+1)} = \Delta V^{(i)} + \tilde{J}^{(i)} \tilde{R} (\Delta V^{(i)}). \quad (63) \]

As previously, this reduces to ordinary matrix inversion in most cases of interest since, we can define a positive semi-definite matrix

\[ \tilde{u} (\Delta v) = \tilde{J} (\Delta V) \tilde{J}^{T} (\Delta V). \quad (64) \]

Provided this is invertible, this gives an iterative procedure for the individual POS algorithm, which is:

\[ \Delta V^{(i+1)} = \Delta V^{(i)} + \tilde{J}^{(i)} \tilde{u}^{T}_{(i)} \tilde{R}^{T}_{(i)}. \quad (65) \]

This method can be implemented to scale as \( \mathcal{O} (MN_S) \) by pre-calculating repeating matrix elements.
4.4. Error propagation

With either of the dynamical POS methods described above, the goal of the optimization is to remove any difference between the sampled observables and the infinite trajectory observables. This can be achieved exactly for static optimization, but there are error propagation effects to be considered in the dynamical case.

To understand this, we note that error propagation effects depend on both the choice of observables and the structure of the equations themselves. Ideally, the optimized set of observables form a closed set of equations, but this is rarely the case in practical nonlinear calculations.

We illustrate the effects of error propagation by considering the one-dimensional moment-based equations. Initially, the error in the \( p \)-th moment is:

\[
\epsilon_p = \left| \langle x^p + \Delta x^p(0) \rangle_{NS} - \langle x^p + \Delta x^p \rangle_\infty \right| \neq 0, \tag{66}
\]

which varies as \( 1/\sqrt{NS} \) for random noises. The aim of our optimization is to set the difference to zero, apart from time-step errors. Can this be achieved exactly, assuming that the stochastic variables themselves are initially optimized at the start of the simulation, that is, \( \langle x^p \rangle_{NS} = \langle x^p \rangle_\infty \) for \( p = 1, \ldots, M \)?

To answer this, note that the drift and the diffusion terms can be expanded in the powers of \( x \) as

\[
a(x,t) = \sum_{j=0}^\infty f_j(t)x^j, \quad b^2(x,t) = \sum_{j=0}^\infty g_j(t)x^j. \tag{67}
\]

After substituting the expansions into the condition expressions, and expanding to order \((\Delta t)^2\),

\[
\epsilon_p = p^2 \left| \left( x^{p-1} \Delta w + \frac{p-1}{2} x^{p-2} (\Delta w^2 - b^2 \Delta t) \right)_{NS} \right.
\]

\[
+ \sum_{j=M-p+2}^\infty f_j(t) \left( \langle x^{p+j-1} \rangle_{NS} - \langle x^{p+j-1} \rangle_\infty \right) \Delta t
\]

\[
+ \frac{p-1}{2} \sum_{j=M-p+3}^\infty g_j(t) \left( \langle x^{p+j-2} \rangle_{NS} - \langle x^{p+j-2} \rangle_\infty \right) \Delta t^2.
\tag{68}
\]

The last two terms depend on unoptimized and thus unknown differences \( \langle x^j \rangle_{NS} - \langle x^j \rangle_\infty \) for \( j > M \). In deriving the POS algorithm, we assume that all the moments of the sampled distribution are correct at the start of the step in time, which allows us to neglect these higher-order moment differences.

One possibility is that the coefficients \( f \) and \( g \) for the corresponding indices are in fact zero. This is equivalent to having \( a \) at most linear in \( x \), and \( b^2 \) at most quadratic in \( x \). If this condition is satisfied, then higher order moments remain uncoupled to lower order moments, and no error propagation occurs apart from the usual error propagation in an ODE method. If this condition is not satisfied, neglecting these terms will result in unoptimized high order moments “leaking in” to low order moments over time, with the effect more noticeable for orders closer to \( M \). We ignore this effect in deriving the algorithm, which means that we can only locally remove sampling errors exactly, as we show later. As a result, higher-order moment errors gradually degrade the optimized moments. This results in globally increased errors for nonlinear SDE solutions, which is demonstrated in the next section.

5. Numerical results

For definiteness, in the numerical examples of POS algorithms analyzed here, the \( m \)-th observable is a finite moment of a one-dimensional real distribution. Hence,

\[
\langle o_m \rangle_{NS} = \frac{1}{NS} \sum_{n=1}^{NS} \left[ x^{(n)} \right]^m = \frac{X^m \cdot 1}{NS}, \tag{69}
\]

13
where $\mathbf{1}$ is an $N_S$-dimensional unit column vector.

5.1. Static optimization

In this section we will test the performance of the iterative algorithm from Section 3.2 applied to the most common initial condition — a normal distribution. In this section we will use $\mu = \mu_1 = 0$ for all the tests and set the variance $\sigma^2 = \mu_2 = 1$. This helps to avoid precision loss when calculating central moments, which is important, since the method is extremely efficient and optimizes the moments almost to the limit of numerical precision, as seen in the results below. If maximum precision is required for cases when $\mu \neq 0$, one can shift the initial distribution so that $\mu = 0$.

With $\mu = 0$, the moments of the test distribution are given by

$$\mu_m = \begin{cases} 0, & m \text{ is odd} \\ \sigma^m (m-1)!!, & m \text{ is even} \end{cases}.$$  

(70)

The number of moments being optimized here is $M = 6$, but this is not essential.

We test the behavior of the algorithms in 10000 independent optimization attempts, each with $N_S = 1000$, in order to collect statistics. In each attempt, a $N_S$-dimensional vector $\mathbf{X}$ is generated using random sampling and the matrix inversion iterative method from Section 3.4 is applied to it. We then plot the distribution of measures of interest, which indicate the performance of the method. This demonstrates how well the algorithm performs when tested over a range of different initial random samples, each drawn from the same underlying normal distribution.

There are several quantities we are interested in. The first is the distribution of final central moments of the vector compared to the desired values:

$$\tilde{R}_m (\mathbf{X}) = \left| \frac{1}{N_S} \mathbf{X}^m \cdot \mathbf{1} - \mu_m \right|,$$

(71)

where we will compare this quantity for initial randomly sampled vectors and optimized vectors. It is convenient to scale this to the expectation of the error in the sampled moment (for random samples) for the
normal distribution:

\[ R_m = \frac{\tilde{R}_m}{\sigma^m \sqrt{m! / N_S}} \]  

so the average of \( R_m \) for the initial randomly sampled vectors will be close to 1 for all \( m \). Fig. 1 shows this quantity both for the moments we are optimizing (\( m = 1 \ldots M \)) and for two higher moments that are not optimized.

The optimization procedure was able to reduce errors almost to the limit of numerical precision of \( R_m \approx 10^{-15} \) in many cases. The moments that were not directly optimized (\( m = 7, 8 \)) show some improvement as well, but typically closer to a single order of magnitude. Since we use logarithmic graphs, and \( R_m \) can sometimes be zero, we excluded these zero results from the binning.

It is also useful to know the number of iterations performed in each of the algorithms. Fig. 2(a) demonstrates that only a small number of iterations — typically four — are necessary for the method to converge, given a reasonably large ensemble.

The third quantity of interest is the relative distance of the optimized vector from the initial randomly sampled one:

\[ D = \frac{\|X^{(\text{opt})} - X\|}{\|X\|}. \]  

We would like this to be much less than 1, which means that the optimized vector is still approximately "random". Fig. 2(b) shows that it is indeed the case, and the average deviation is \( D \approx 5 \times 10^{-3} \) for \( \sigma = 1 \). Fig. 2(c) shows that the results have the desirable property that the relative distance between the optimized and non-optimized trajectories scales with \( 1/\sqrt{N_S} \). This is expected from the fact that the initial sampled moment relative errors scale as \( 1/\sqrt{N_S} \), hence one expects the minimum required change to be of this order.

However, if there are too few trajectories, the simple matrix inversion iteration method diverges, producing unacceptably large changes \( D \) in the distribution. For these small \( N_S \) values, the iteration limit of \( I_{\text{max}} = 50 \) was reached, indicating a lack of convergence.

To investigate this more carefully, the dependence of the final error \( R \) on the number of trajectories \( N_S \), is plotted in Fig. 3. The results also display a sharp cut-off in the number of trajectories below which the iterative approach does not converge. More robust minimization algorithms could presumably solve this, but the issue was not investigated here for space reasons.

In conclusion, the expansion iterative method performs well for a wide range of parameters, while being fast and easy to implement. Convergence is extremely rapid when the number of stochastic trajectories is increased above a critical value. However, we note that many other methods of nonlinear root-finding exist, and we do not exclude the possibility of better techniques being available.

We use this approach in Section 5.3 to pre-optimize the starting conditions for the SDE integration tests.
5.2. Synthetic one-step benchmarks

Before we apply our POS method to solving an SDE with many successive minimization steps, it is informative to see how effective it is at finding a solution for the corresponding single step equation.

In order to evaluate the performance of the method at single-step error optimization, we implement the combined and individual optimization methods with 10000 separate initial ensembles of $X$ and initial stochastic noises $W$. In all the tests, the initial elements of $X$ are normally distributed with the mean 1 and standard deviation 0.1, $a_i = 0.5$ for all $i$ (constant drift), $b_i = 0.5$ for all $i$ (additive noise).

Results for more complicated synthetic tests show similar behavior, so we do not include them here.

5.2.1. Combined optimization

In this section we are interested in how well we can solve the target equation (45). The error measure in this case is essentially the same as in Section 5.1:

$$
\tilde{R}_m(X) = \frac{1}{N_S} X^m \cdot 1 - c_m,
$$

where the target moments $c_m$ are given by Eq. (47).

Similarly to the previous section, we will scale the errors as

$$
R_m = \frac{\tilde{R}_m}{\sqrt{\Delta t/N_S}}
$$

in order to bring the errors for the unoptimized, randomly sampled vectors close to 1.

The results in Fig. 5 show that a significant improvement is achieved for all 6 moments being optimized, with normalized sampling errors reduced to $10^{-12}$, or around twelve orders of magnitude in these examples. Even the non-optimized moments of orders $m = 7, 8$ have errors reduced to $10^{-5}$, or around five orders.

The method requires only a few iterations to converge, as Fig. 5(a) demonstrates. We also calculate the distance

$$
D = \frac{\|\Delta X^{(opt)} - \Delta X_{(0)}\|}{\|\Delta X_{(0)}\|} = \frac{\|X^{(opt)} (t + \Delta t) - X_{(0)} (t + \Delta t)\|}{\|X_{(0)} (t + \Delta t) - X (t)\|}
$$

(76)

to estimate how far the solution is from the initial approximation. Fig. 5(b) shows that the deviation of the optimized noise from the original is 0.07 on average, which is of order $1/\sqrt{N_S}$ as expected.
Figure 4: Distribution of normalized errors in the synthetic benchmark for the combined method with $\Delta t = 10^{-4}$, $N_S = 1000$. Blue solid lines show the distribution of $R_m$ for the optimized vector, grey dotted lines show the distribution of $R_m$ for the initial approximation $X_{(0)} (t + \Delta t)$.

Figure 5: (a) number of iterations taken in the synthetic test for the combined method, $\Delta t = 10^{-4}$, $N_S = 1000$, (b) Relative distance $D$ between the optimized vector $X^{(opt)} (t + \Delta t)$ and the initial approximation $X_{(0)} (t + \Delta t)$, and (c) average relative distance $\langle D \rangle$ depending on the number of trajectories $N_S$ in the synthetic optimization tests for the individual expansion method, $\Delta t = 10^{-4}$. The grey dashed line denotes the slope of a $1/\sqrt{N_S}$ dependence.
The difference in the orders of $\Delta t$ in the left and the right parts of the target equation (45) leads to the distance $D$ being bounded by a finite value, depending on either $\Delta t$ or $1/N_S$. This is unlike the behavior demonstrated by the static optimization in the previous section. Our tests show that for very small $\Delta t$, the asymptotic dependence of $\langle D \rangle$ on $N_S$ is $\langle D \rangle \propto 1/\sqrt{N_S}$, as shown in Fig. 6(a). Similarly, for very large $N_S$ the dependence is $\langle D \rangle \propto \Delta t^{3/2}$, as seen in Fig. 6(b). This asymptotic error is essentially due to the fact that the moment equations are themselves an expansion in $\Delta t$, leading to a residual truncation error.

The tests of the dependence on the noise scale ($\Delta t$) show that the method is stable for a wide range of values, but breaks down at larger values of $\Delta t$ as seen in Fig. 7. The dependence on the number of trajectories in Fig. 8 displays a similar cutoff as seen in the tests of the static POS, in the results of the previous subsection.

5.2.2. Individual optimization

In this subsection we will use the individual method from Section 4.3 for a similar type of synthetic benchmark. Since the target equations we try to satisfy are different from the previous section, the definition of the error changes to simply

$$\tilde{R}_m (X) = |e_m^{(1)}| + |e_m^{(2)}|,$$

(77)

where the vectors $e^{(1)}$ and $e^{(2)}$ are defined by Eq. (54).

Similarly to the approach of the previous subsection, we normalize the error as

$$R_m = \frac{\tilde{R}_m}{\sqrt{\Delta t/N_S}}.$$  (78)

The results in Fig. 9(a) show that a significant improvement is achieved for the first 6 moments being optimized, with results only limited by the numerical precision. To check that numerical precision was the limiting factor, tests were run with the same method in quadruple precision. This demonstrated that the improvement was indeed limited only by the numerical precision, with errors now as low as $10^{-30}$. The stopping condition must be changed to $\eta = 10^{-15}$ to reach this greater numerical precision limit.
Figure 7: Dependence of errors on the number of trajectories $N_S$ in the synthetic test for the combined method, $\Delta t = 10^{-4}$. Blue solid lines show the average of $R_m$ for the optimized vector, grey dotted lines show the average of $R_m$ for the initial approximation $X_{(0)} (t + \Delta t)$.

Figure 8: Dependence of average scaled errors on $\Delta t$ in the synthetic test for the combined method, $N_S = 1000$. Blue solid lines show the average of $R_m$ for the optimized vector, grey dotted lines show the average of $R_m$ for the initial approximation $X_{(0)} (t + \Delta t)$. 
Figure 9: Distribution of normalized errors in the synthetic benchmark for the individual method with $\Delta t = 0.1$, $N_S = 1000$. Blue solid lines show the distribution of $R_m$ for the optimized vector, grey dotted lines show the distribution of $R_m$ for the initial effective noise term $\Delta V$.

Figure 10: (a) The number of iterations taken in the synthetic test for the individual method, $\Delta t = 0.1$, $N_S = 1000$. (b) Relative distance $D$ between the optimized vector $\Delta V^{(opt)}$ and the initial approximation $\Delta V$. (c) Average relative distance $\langle D \rangle$ depending on the number of trajectories $N_S$ in the synthetic optimization tests for the individual method, $\Delta t = 0.1$. The grey dashed line denotes the slope of a $1/\sqrt{N_S}$ dependence.
We also calculate the distance

\[ D \equiv \frac{\|\Delta V^{(\text{opt})} - \Delta V\|}{\|\Delta V\|} \]  

and the number of iterations in our tests. The method also requires only a few iterations to converge, as Fig. 10(a) demonstrates. Fig. 10(b) shows that the deviation of the optimized noise from the original one is small. Unlike the combined method, there is no mismatch between the orders of \( \Delta t \) in Eq. (54), and the distance therefore converges as \( \langle D \rangle \propto 1/\sqrt{N_S} \), as shown in Fig. 10(c).

The tests of the dependence on the noise scale (\( \Delta t \)) show that this method is exceptionally stable over a wide range of step-size values, as seen in Fig. 11. The dependence on the number of trajectories in Fig. 12 displays a similar cutoff to those seen in the tests of the static POS.

5.3. SDE benchmarks

5.3.1. Linear drift, additive noise

Next, tests were performed of complete SDE solutions. It is instructive to first consider an “ideal” linear SDE for the application of POS, with the target observables being the moments of \( x \) as previously. An ideal linear case is one with a linear \( a \) and \( b \). This is preferred as a first test because the errors from non-optimized moments do not get mixed with the optimized ones, as explained in Section 4.4. In these benchmarks, both the combined and individual methods were used, giving very similar performance overall.

As an example, we used a one-dimensional Ornstein-Uhlenbeck process

\[ dx = (f - gx) dt + bw, \]  

where \( \langle dw \rangle = 0 \) and \( \langle dw^2 \rangle = dt \), and \( f, g \) and \( b \) are constants. It has well-known exact solutions for every moment. Namely, for Gaussian starting conditions the mean is given by:

\[ \langle x \rangle = \frac{f}{g} \left( 1 - e^{-gt} \right) + e^{-gt} \langle x \rangle_{t=0}, \]  

and for the variance,

\[ \langle (x - \langle x \rangle )^2 \rangle = e^{-2gt} \langle (x - \langle x \rangle )^2 \rangle_{t=0} + \frac{b^2}{2g} \left( 1 - e^{-2gt} \right), \]  

Figure 11: Dependence of average scaled errors on \( \Delta t \) in the synthetic test for the individual method, \( N_S = 1000 \). Blue solid lines show the average of \( R_m \) for the optimized vector, grey dotted lines show the average of \( R_m \) for the initial approximation \( \Delta V^{(0)} \).
We also use cumulants as benchmarks, since these are a commonly used alternative statistical measure to the ordinary moment. The cumulants are calculated from the moments recursively, as

$$\kappa_m = \langle x^m \rangle - \sum_{p=1}^{m-1} \binom{m-1}{p-1} \kappa_p \langle x^{m-p} \rangle.$$  

In our tests we took $f = 1$, $g = 0.2$, $b = 0.5$. At initial times, $x(0)$ is normally distributed with mean 0.5 and standard deviation 0.1. We pre-optimized the starting distributions using the static method from Section 3.4 in order to satisfy the requirements of POS methods.

For our tests, we compare the deviations from the exact solution in the first 8 moments and first 8 cumulants of $x$ at $t = 1$ for the POS integrator, using both combined and individual methods, with a reference SDE solver. The latter uses the same integration method (explicit Euler) and the same number of trajectories as POS. We set up POS to optimize the first 6 moments, so that we could look at the behavior of both optimized and non-optimized moments.

In Figs. 13 and 14 we plotted the error in the first moments

$$E[\langle x^m \rangle] = \left| \langle x^m \rangle - \langle x^m \rangle^{(exact)} \right|$$

for the reference and the two POS integrators, averaged over 8 independent tests. In case of the POS methods the error is dominated by the time step error $\epsilon_T$, while the error for the reference integrator
Figure 13: Linear SDE case, showing the difference of the first 8 moments from the exact solution at $t = 1$ for the POS integrator using the combined method (solid blue lines) and a reference explicit Euler integrator (dashed red lines), averaged over 8 tests. Both integrators use $N_T = 8 \times 10^4$ time steps.

Figure 14: Linear SDE case, showing the difference of the first 8 moments from the exact solution at $t = 1$ for the POS integrator using the individual method (solid blue lines) and a reference explicit Euler integrator (dashed red lines), averaged over 8 tests. Both integrators use $N_T = 8 \times 10^4$ time steps.
Figure 15: Linear SDE case, showing the difference of the first 8 cumulants from the exact solution at $t = 1$ for the POS integrator using the individual method (solid blue lines) and a reference explicit Euler integrator (dashed red lines), averaged over 8 tests. Both integrators use $N_T = 8 \times 10^4$ time steps.

Decreases proportional to $1/\sqrt{N_S}$. One can see that the POS error is significantly decreased even for the moments that were not directly optimized. The error improvement is largest for the individual method, but is still several orders of magnitude even for the simpler combined method. For the remaining graphs we will mostly focus on the individual method which give better results, to minimize the number of graphs, as both have similar general behavior.

The general behavior in the linear case can be understood better if we plot the errors in the cumulants $E[\kappa_p]$ instead, as in Fig. 15. It shows that the error in the first 6 cumulants is greatly reduced, while the non-optimized cumulants stay roughly the same. In Fig. 14, 7-th and 8-th moments, while not being optimized directly, still depend on the lower cumulants, and thus benefit from their reduced error. This behavior is caused by the fact that in a linear SDE, cumulants of different orders are not coupled with each other; however moments are coupled, leading to the observed error performance.

5.3.2. Nonlinear drift, additive noise

As an example of a real-world, nonlinear SDE we can apply the POS method to, we take an equation with a nonlinear drift term and additive noise:

$$dx = (x - x^3) \, dt + dw,$$

where $\langle dw \rangle = 0$ and $\langle dw^2 \rangle = 1$. This is similar to a known case where moment hierarchy methods fail to give correct results [18]. Since the exact solution for this equation is not known, we use an accurate solution obtained from a regular SDE integrator (with $N_T = 8 \times 10^4$ time steps and $N_S = 10^9$ trajectories, central difference integration method [23]).

As in the previous section, we plot the first 8 moments and first 8 cumulants of $x$ at $t = 4$, but we optimize only the first 6 moments when the POS is used. The reference SDE integrator uses the same integration method (explicit Euler) and the same number of trajectories as POS.

In Figs. 16 and 17 we are plotting differences from the accurate solution for the first 8 moments:

$$E[\langle x^m \rangle] = \left| \langle x^m \rangle - \langle x^m \rangle^{(acc)} \right|$$

averaged over 8 independent test runs. We compare these errors with the full error $\epsilon_T + \epsilon_S$ of the accurate solution. Both of the POS integrators consistently outperform the reference integrator by several orders of magnitude, typically a factor of 100. A sampling error reduction of this order of magnitude requires $10^4$
Figure 16: Nonlinear SDE case, showing the difference of the first 8 moments from the accurate solution at $t = 4$ for the POS integrator using the combined method (solid blue lines) and a reference explicit Euler integrator (dashed red lines), averaged over 8 tests. Both integrators use $N_T = 8 \times 10^4$ time steps. The combined time step and sampling error of the accurate solution is shown as dotted grey lines. Dash-dotted grey lines show the slopes for $\propto N_S^{-1/2}$ and $\propto N_S^{-1}$ dependencies.

Figure 17: Nonlinear SDE case, showing the difference of the first 8 moments from the accurate solution at $t = 4$ for the POS integrator using the individual method (solid blue lines) and a reference explicit Euler integrator (dashed red lines), averaged over 8 tests. Both integrators use $N_T = 8 \times 10^4$ time steps. The combined time step and sampling error of the accurate solution is shown as dotted grey lines. Dash-dotted grey lines show the slopes for $\propto N_S^{-1/2}$ and $\propto N_S^{-1}$ dependencies.
times more samples using standard integrators. This is consistent with a sampling error reduction from $N_S^{-1/2}$ using standard methods, down to nearly $N_S^{-1}$ for the low order moments with POS.

Finally, in Fig. 18 we show results for the cumulants in the nonlinear case, for completeness. For brevity, we only give the individual results, which are slightly better, but otherwise very similar to the combined algorithm. Unlike the linear case, where the improvement was confined to only the explicitly optimized cumulants, we see that in the nonlinear case even the non-optimized cumulants are optimized as well. This is because the nonlinear equations couple all cumulants with each other.

6. Conclusions

In summary, we have proposed and implemented a novel parallel optimized sampling technique for solving stochastic differential equations. The essential feature is that it unifies moment hierarchy and independent stochastic methods. A finite moment hierarchy condition is imposed as a nonlinear constraint on the random noises generated at each step in the integration. This gives a dramatic reduction in sampling error for all moments calculated. In the case of linear equations, we find that the low order moments have sampling errors reduced to machine accuracy. While higher order moments also have their errors reduced, the higher order cumulants are not affected. For nonlinear equations, the error reduction is not as large, but it occurs over all moments and cumulants studied, even including the non-optimized cumulants of higher order than the optimization limit. We emphasize that the proposed algorithms have a general applicability to all types of stochastic equations, and can be extended in principle to higher order methods as well as the simple Euler integration treated here.

Acknowledgements

We wish to acknowledge financial support from the Australian Research Council through the Discovery Grant program, as well as useful discussions with Rodney Polkinghorne, Piotr Deuar and Brian Dalton.
References

[1] P. Langevin, Sur la théorie du mouvement brownien, C. R. Acad. Sci. (Paris) 146 (1908) 530–533.
[2] D. S. Lemons, Paul Langevin’s 1908 paper “On the Theory of Brownian Motion” (“Sur la théorie du mouvement brownien,” C. R. Acad. Sci. (Paris) 146, 530–533 (1908)), Am. J. Phys. 65 (11) (1997) 1079. doi:10.1119/1.18726
[3] G. Uhlenbeck, L. Ornstein, On the Theory of the Brownian Motion, Phys. Rev. 36 (5) (1930) 823–841. doi:10.1103/PhysRev.36.823
[4] L. Arnold, Stochastic differential equations: theory and applications, reprint Edition, Folens Publishers, 1992.
[5] H. Risken, The Fokker-Planck Equation, 2nd Edition, Springer-Verlag, Berlin, 1996.
[6] C. W. Gardiner, Handbook of Stochastic Methods, 2nd Edition, Springer-Verlag, Berlin, 1985.
[7] I. Karatzas, S. Shreve, Brownian motion and stochastic calculus, 2nd Edition, Springer, 1991.
[8] N. G. Van Kampen, Stochastic Processes in Physics and Chemistry, 3rd Edition, North Holland, 2007.
[9] P. Glasserman, Monte Carlo Methods in Financial Engineering, reprint Edition, Springer, 2010.
[10] G. N. Mil’shtein, Approximate Integration of Stochastic Differential Equations, Theor. Probab. Appl. 19 (3) (1975) 557–562. doi:10.1137/1119062
[11] D. L. Ermak, H. Buckholz, Numerical integration of the Langevin equation: Monte Carlo simulation, J. Comput. Phys. 35 (2) (1980) 169–182. doi:10.1016/0021-9991(80)90084-4
[12] D. Talay, L. Tubaro, Expansion of the global error for numerical schemes solving stochastic differential equations, Stoch. Anal. Appl. 8 (4) (2007) 483–509. doi:10.1080/0736299008809220
[13] P. E. Kloeden, E. Platen, Numerical Solution of Stochastic Differential Equations, Springer-Verlag, Berlin, 1992.
[14] K. Burrage, T. Tian, A note on the stability properties of the euler methods for solving stochastic differential equations, New Zeal. J. Math. 29 (2) (2000) 115–127.
[15] D. J. Higham, An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations, SIAM Rev. 43 (3) (2001) 525–546. doi:10.1137/S0036144500378302
[16] W. J. Morokoff, R. E. Caflisch, Quasi-Monte Carlo Integration, J. Comput. Phys. 122 (2) (1995) 218–230. doi:10.1006/jcph.1995.1209
[17] M. B. Giles, Multilevel Monte Carlo Path Simulation, Oper. Res. 56 (3) (2008) 607–617. doi:10.1287/opre.1070.0496
[18] M. Morillo, J. Gómez-Ordóñez, J. M. Casado, Checking the Validity of Truncating the Cumulant Hierarchy Description of a Small System, in: R. Carretero-González, J. Cuevas-Maraver, D. Frantzeskakis, N. Karachalios, P. Kevrekidis, F. Palmero-Acebedo (Eds.), Localized Excitations in Nonlinear Complex Systems, Vol. 7 of Nonlinear Systems and Complexity, Springer International Publishing, 2014, pp. 377–387. doi:10.1007/978-3-319-02057-0
[19] P. Hänggi, P. Talkner, A remark on truncation schemes of cumulant hierarchies, J. Stat. Phys. 22 (1) (1980) 65–67. doi:10.1007/BF01007988
[20] A. Gilchrist, C. W. Gardiner, P. D. Drummond, Positive P representation: Application and validity, Phys. Rev. A 55 (4) (1997) 3014–3032. doi:10.1103/PhysRevA.55.3014
[21] P. P. Deuar, P. D. Drummond, Gauge P representations for quantum-dynamical problems: Removal of boundary terms, Phys. Rev. A 66 (3) (2002) 033812. doi:10.1103/PhysRevA.66.033812
[22] T. Aimi, M. Imada, Gaussian-Basis Monte Carlo Method for Numerical Study on Ground States of Itinerant and Strongly Correlated Electron Systems, J. Phys. Soc. Jpn. 76 (8) (2007) 084709. doi:10.1143/JPSJ.76.084709
[23] M. J. Werner, P. D. Drummond, Robust Algorithms for Solving Stochastic Partial Differential Equations, J. Comput. Phys. 132 (2) (1997) 312–326. doi:10.1006/jcph.1996.5638