Entropy **2014**, *16*, 221-232; doi:10.3390/e16010221

OPEN ACCESS
entropy
ISSN 1099-4300
www.mdpi.com/journal/entropy

Article

Malliavin Weight Sampling: A Practical Guide

Patrick B. Warren [1,*] and Rosalind J. Allen [2,*]

[1] Unilever R&D Port Sunlight, Quarry Road East, Bebington, Wirral, CH63 3JW, UK
[2] Scottish Universities Physics Alliance (SUPA), School of Physics and Astronomy, the University of Edinburgh, the Kings Buildings, Mayfield Road, Edinburgh, EH9 3JZ, UK

* Authors to whom correspondence should be addressed;
E-Mails: patrick.warren@unilever.com (P.B.W.); rallen2@ph.ed.ac.uk (R.J.A.);
Tel.: +44-151-641-3352 (P.B.W.); +44-131-651-7197 (R.J.A.).

Received: 25 September 2013; in revised form: 9 October 2013 / Accepted: 18 October 2013 / Published: 27 December 2013

Abstract: Malliavin weight sampling (MWS) is a stochastic calculus technique for computing the derivatives of averaged system properties with respect to parameters in stochastic simulations, without perturbing the system's dynamics. It applies to systems in or out of equilibrium, in steady state or time-dependent situations, and has applications in the calculation of response coefficients, parameter sensitivities and Jacobian matrices for gradient-based parameter optimisation algorithms. The implementation of MWS has been described in the specific contexts of kinetic Monte Carlo and Brownian dynamics simulation algorithms. Here, we present a general theoretical framework for deriving the appropriate MWS update rule for any stochastic simulation algorithm. We also provide pedagogical information on its practical implementation.

Keywords: stochastic calculus; Brownian dynamics

---

1. Introduction

Malliavin weight sampling (MWS) is a method for computing derivatives of averaged system properties with respect to parameters in stochastic simulations [1,2]. The method has been used in quantitative financial modelling to obtain the "Greeks" (price sensitivities) [3]; and as the Girsanov transform, in kinetic Monte Carlo simulations for systems biology [4]. Similar ideas have been used to study fluctuation-dissipation relations in supercooled liquids [5]. However, MWS appears to be relatively

unknown in the fields of soft matter, chemical and biological physics, perhaps because the theory is relatively impenetrable for non-specialists, being couched in the language of abstract mathematics (e.g., martingales, Girsanov transform, Malliavin calculus, *etc.*); an exception in financial modelling is [6].

MWS works by introducing an auxiliary stochastic quantity, the Malliavin weight, for each parameter of interest. The Malliavin weights are updated alongside the system's usual (unperturbed) dynamics, according to a set of rules. The derivative of any system function, A, with respect to a parameter of interest is then given by the average of the product of A with the relevant Malliavin weight; or in other words, by a weighted average of A, in which the weight function is given by the Malliavin weight. Importantly, MWS works for non-equilibrium situations, such as time-dependent processes or driven steady states. It thus complements existing methods based on equilibrium statistical mechanics, which are widely used in soft matter and chemical physics.

MWS has so far been discussed only in the context of specific simulation algorithms. In this paper, we present a pedagogical and generic approach to the construction of Malliavin weights, which can be applied to any stochastic simulation scheme. We further describe its practical implementation in some detail using as our example one dimensional Brownian motion in a force field.

2. The Construction of Malliavin Weights

The rules for the propagation of Malliavin weights have been derived for the kinetic Monte-Carlo algorithm [4,7], for the Metropolis Monte-Carlo scheme [5] and for both underdamped and overdamped Brownian dynamics [8]. Here, we present a generic theoretical framework, which encompasses these algorithms and also allows extension to other stochastic simulation schemes.

We suppose that our system evolves in some state space, and a point in this state space is denoted as S. Here, we assume that the state space is continuous, but our approach can easily be translated to discrete or mixed discrete-continuous state spaces. Since the system is stochastic, its state at time t is described by a probability distribution, $P(S)$. In each simulation step, the state of the system changes according to a propagator, $W(S \to S')$, which gives the probability that the system moves from point S to point S' during an application of the update algorithm. The propagator has the property that:

$$P'(S') = \int_S dS\; W(S \to S')\, P(S) \tag{1}$$

where $P'(S)$ is the probability distribution after the update step has been applied and the integral is over the whole state space. We shall write this in a shorthand notation as:

$$P' = \int WP \tag{2}$$

Integrating Equation (1) over S', we see that the propagator must obey $\int_{S'} W(S \to S') = 1$. It is important to note, however, that we do *not* assume the detailed balance condition $P_{\text{eq}}(S)\, W(S \to S') = P_{\text{eq}}(S')\, W(S' \to S)$ (for some equilibrium $P_{\text{eq}}(S)$). Thus, our results apply to systems whose dynamical rules do not obey detailed balance (such as chemical models of gene regulatory networks [9]), as well as to systems out of steady state. We observe that the (finite) product:

$$\mathbb{W}(S_1, \ldots, S_n) = W(S_1 \to S_2) \times \cdots \times W(S_{n-1} \to S_n) \tag{3}$$

is proportional to the probability of occurrence of a trajectory of states, $\{S_1, \ldots, S_n\}$, and can be interpreted as a *trajectory weight*.

Let us now consider the average of some quantity, $A(S)$, over the state space, in shorthand:

$$\langle A \rangle = \int A\, P \tag{4}$$

The quantity, A, might well be a complicated function of the state of the system: for example the extent of crystalline order in a particle-based simulation, or a combination of the concentrations of various chemical species in a simulation of a biochemical network. We suppose that we are interested in the sensitivity of $\langle A \rangle$ to variations in some parameter of the simulation, which we denote as λ. This might be one of the force field parameters (or the temperature) in a particle-based simulation or a rate constant in a kinetic Monte Carlo simulation. We are interested in computing $\partial\langle A \rangle/\partial\lambda$. This quantity can be written as:

$$\frac{\partial\langle A \rangle}{\partial\lambda} = \int A\, P\, Q_\lambda \tag{5}$$

where:

$$Q_\lambda = \frac{\partial \ln P}{\partial\lambda} \tag{6}$$

(using the fact that $\partial \ln P/\partial\lambda = (1/P)\partial P/\partial\lambda$).

Let us now suppose that we track in our simulation not only the physical state of the system, but also an auxiliary stochastic variable, which we term q_λ. At each simulation step, q_λ is updated according to a rule that depends on the system state; this does not perturb the system's dynamics, but merely acts as a "readout". By tracking q_λ, we *extend* the state space, so that S becomes $\{S, q_\lambda\}$. We can then define the average $\langle q_\lambda \rangle_S$, which is an average of the value of q_λ in the extended state space, with the constraint that the original (physical) state space point is fixed at S (see further below).

Our aim is to define a set of rules for updating q_λ, such that $\langle q_\lambda \rangle_S = Q_\lambda$, *i.e.*, such that the average of the auxiliary variable, for a particular state space point, measures the *derivative* of the probability distribution with respect to the parameter of interest, λ. If this is the case then, from Equation (5):

$$\frac{\partial\langle A \rangle}{\partial\lambda} = \langle A\, q_\lambda \rangle \tag{7}$$

The auxiliary variable, q_λ, is the Malliavin weight corresponding to the parameter, λ.

How do we go about finding the correct updating rule? If the Malliavin weight exists, we should be able to derive its updating rule from the system's underlying stochastic equations of motion. We obtain an important clue from differentiating Equation (1) with respect to λ. Extending the shorthand notation, one finds:

$$P' Q'_\lambda = \int W P \left(Q_\lambda + \frac{\partial \ln W}{\partial\lambda} \right) \tag{8}$$

This strongly suggests that the rule for updating the Malliavin weight should be:

$$q'_\lambda = q_\lambda + \frac{\partial \ln W}{\partial\lambda} \tag{9}$$

In fact, this is correct. The proof is not difficult and, for the case of Brownian dynamics, can be found in the supplementary material for [8]. It involves averaging Equation (9) in the extended state space, $\{S, q_\lambda\}$.

From a practical point of view, for each time step, we implement the following procedure:

- propagate the system from its current state, S, to a new state, S', using the algorithm that implements the stochastic equations of motion (Brownian, kinetic Monte-Carlo, *etc.*);
- with knowledge of S and S', and the propagator, $W(S \to S')$, calculate the change in the Malliavin weight $\Delta q_\lambda = \partial \ln W(S \to S')/\partial\lambda$;
- update the Malliavin weight according to $q_\lambda \to q'_\lambda = q_\lambda + \Delta q_\lambda$.

At the start of the simulation, the Malliavin weight is usually initialised to $q_\lambda = 0$.

Let us first suppose that our system is not in steady state. However, rather, the quantity, $\langle A \rangle$, in which we are interested is changing in time, and likewise, $\partial\langle A(t)\rangle/\partial\lambda$ is a time-dependent quantity. To compute $\partial\langle A(t)\rangle/\partial\lambda$, we run N independent simulations, in each one tracking, as a function of time $A(t)$, $q_\lambda(t)$ and the product, $A(t)\, q_\lambda(t)$. The quantities, $\langle A(t)\rangle$ and $\partial\langle A(t)\rangle/\partial\lambda$, are then given by:

$$\langle A(t)\rangle \approx \frac{1}{N}\sum_{i=1}^{N} A_i(t), \quad \frac{\partial\langle A(t)\rangle}{\partial\lambda} \approx \frac{1}{N}\sum_{i=1}^{N} A_i(t)\, q_{\lambda,i}(t) \tag{10}$$

where $A_i(t)$ is the value of $A(t)$ recorded in the ith simulation run (and likewise for $q_{\lambda,i}(t)$). Error estimates can be obtained from the variance among the replicate simulations.

If, instead, our system is in steady state, the procedure needs to be modified slightly. This is because the variance in the values of $q_\lambda(t)$ across replicate simulations increases linearly in time (this point is discussed further below). For long times, computation of $\partial\langle A\rangle/\partial\lambda$ using Equation (10) therefore incurs a large statistical error. Fortunately, this problem can easily be solved, by computing the correlation function:

$$C(t,t') = \langle A(t)\,[q_\lambda(t) - q_\lambda(t')]\rangle \tag{11}$$

In steady state, $C(t,t') = C(t-t')$, with the property that $C(\Delta t) \to \partial A/\partial\lambda$ as $\Delta t \to \infty$. In a single simulation run, we simply measure $q_\lambda(t)$ and $A(t)$ at time intervals separated by Δt (which is typically multiple simulation steps). At each measurement, we compute $A(t)\,[q_\lambda(t) - q_\lambda(t-\Delta t)]$. We then average this latter quantity over the whole simulation run to obtain an estimate of $\partial\langle A\rangle/\partial\lambda$. For this estimate to be accurate, we require that Δt is long enough that $C(\Delta t)$ has reached its plateau value; this typically means that Δt should be longer than the typical relaxation time of the system's dynamics. The correlation function approach is discussed in more detail in [7,8].

Returning to a more theoretical perspective, it is interesting to note that the rule for updating the Malliavin weight, Equation (9), depends deterministically on S and S'. This implies that the value of the Malliavin weight at time t is completely determined by the trajectory of system states during the time interval, $0 \to t$. In fact, it is easy to show that:

$$q_\lambda = \frac{\partial \ln \mathbb{W}}{\partial\lambda} \tag{12}$$

where $\mathbb{W}$ is the trajectory weight defined in Equation (3). Similar expressions are given in [5,7]. Thus, the Malliavin weight, q_λ, is not fixed by the state point, S, but by the entire trajectory of states that have led to state point S. Since many different trajectories can lead to S, many values of q_λ are possible for the same state point, S. The average $\langle q_\lambda(t)\rangle_S$ is actually the expectation value of the Malliavin weight, averaged over all trajectories that reach state point S at time t. This can be used to obtain an alternative proof that $\langle q_\lambda\rangle_S = \partial \ln P/\partial\lambda$. Suppose we sample N trajectories, of which N_S end up at state point S

(or a suitably defined vicinity thereof, in a continuous state space). We have $P(S) = \langle N_S \rangle / N$. Then, the Malliavin property implies $\partial P / \partial \lambda = \langle N_S \, q_\lambda \rangle / N$, and hence, $\partial \ln P / \partial \lambda = \langle N_S \, q_\lambda \rangle / \langle N_S \rangle = \langle q_\lambda \rangle_S$.

3. Multiple Variables, Second Derivatives and the Algebra of Malliavin Weights

Up to now, we have assumed that the quantity, A, does not depend explicitly on the parameter, λ. There may be cases, however, when A does have an explicit λ-dependence. In these cases, Equation (7) should be replaced by:

$$\frac{\partial \langle A \rangle}{\partial \lambda} = \left\langle \frac{\partial A}{\partial \lambda} \right\rangle + \langle A \, q_\lambda \rangle \tag{13}$$

If we set A to be a constant in this, we immediately obtain the general result that $\langle q_\lambda \rangle = 0$. Equation (13) reveals a kind of 'algebra' for Malliavin weights: we see that the operations of taking an expectation value and taking a derivative can be commuted, provided the Malliavin weight is introduced as the commutator.

We can also extend our analysis further to allow us to compute higher derivatives with respect to the parameters. These may be useful, for example, for increasing the efficiency of gradient-based parameter optimisation algorithms. Taking the derivative of Equation (13) with respect to a second parameter, μ, gives:

$$\begin{aligned} \frac{\partial^2 \langle A \rangle}{\partial \lambda \partial \mu} &= \frac{\partial}{\partial \mu} \left\langle \frac{\partial A}{\partial \lambda} \right\rangle + \frac{\partial \langle A \, q_\lambda \rangle}{\partial \mu} \\ &= \left\langle \frac{\partial^2 A}{\partial \lambda \partial \mu} \right\rangle + \left\langle \frac{\partial A}{\partial \lambda} \, q_\mu \right\rangle + \left\langle A \, \frac{\partial q_\lambda}{\partial \mu} \right\rangle + \left\langle \frac{\partial A}{\partial \mu} \, q_\lambda \right\rangle + \langle A \, q_\lambda \, q_\mu \rangle \\ &= \langle A \, (q_{\lambda\mu} + q_\lambda q_\mu) \rangle + \left\langle \frac{\partial A}{\partial \lambda} \, q_\mu \right\rangle + \left\langle \frac{\partial A}{\partial \mu} \, q_\lambda \right\rangle + \left\langle \frac{\partial^2 A}{\partial \lambda \partial \mu} \right\rangle \end{aligned} \tag{14}$$

where, in the second line, we iterate the commutation relation and, in the third line, we collect like terms and introduce:

$$q_{\lambda\mu} = \frac{\partial q_\lambda}{\partial \mu} \tag{15}$$

In the case where A is independent of the parameters, this result simplifies to:

$$\frac{\partial^2 \langle A \rangle}{\partial \lambda \partial \mu} = \langle A \, (q_{\lambda\mu} + q_\lambda q_\mu) \rangle \tag{16}$$

The quantity, $q_{\lambda\mu}$, here is a new, second order Malliavin weight, which, from Equations (12) and (15), satisfies:

$$q_{\lambda\mu} = \frac{\partial^2 \ln \mathbb{W}}{\partial \lambda \partial \mu} \tag{17}$$

To compute second derivatives with respect to the parameters, we should therefore track these second order Malliavin weights in our simulation, updating them alongside the existing Malliavin weights by the rule:

$$q'_{\lambda\mu} = q_{\lambda\mu} + \frac{\partial^2 \ln W(S \to S')}{\partial \lambda \partial \mu} \tag{18}$$

Setting A as a constant in Equation (16), we also obtain the interesting result that $\langle q_{\lambda\mu} \rangle = -\langle q_\lambda q_\mu \rangle$.

Steady state problems can be approached by extending the correlation function method to second order weights. Define, *cf.* Equation (11):

$$C(t,t') = \langle A(t)\,\{[q_{\lambda\mu}(t) + q_\lambda(t)q_\mu(t)] - [q_{\lambda\mu}(t') + q_\lambda(t')q_\mu(t')]\}\rangle \tag{19}$$

As in the first order case, in steady state, we expect $C(t,t') = C(t-t')$, with the property that $C(\Delta t) \to \partial^2\langle A\rangle/\partial\lambda\partial\mu$ as $\Delta t \to \infty$.

4. One-Dimensional Brownian Motion in a Force Field

We now demonstrate this machinery by way of a practical, but very simple, example, namely, one-dimensional (overdamped) Brownian motion in a force field. In this case, the state space is specified simply by the particle position, x, which evolves according to the Langevin equation:

$$\frac{dx}{dt} = f(x) + \eta \tag{20}$$

where $f(x)$ is the force field and η is Gaussian white noise of amplitude $2T$, where T is temperature. Without loss of generality, we have chosen units, so that there is no prefactor multiplying the force field. We discretise the Langevin equation to the following updating rule:

$$x' = x + f(x)\,\delta t + \xi \tag{21}$$

where δt is the time step and ξ is a Gaussian random variate with zero mean and variance $2T\,\delta t$. Corresponding to this updating rule is an explicit expression for the propagator:

$$W(x \to x') = \frac{1}{\sqrt{4\pi T\,\delta t}}\,\exp\Big(-\frac{(x'-x-f(x)\,\delta t)^2}{4T\,\delta t}\Big) \tag{22}$$

This follows from the statistical distribution of ξ. Let us suppose that the parameter of interest, λ, enters into the force field (the temperature, T, could also be chosen as a parameter). Making this assumption:

$$\frac{\partial \ln W(x \to x')}{\partial\lambda} = \frac{(x'-x-f\,\delta t)}{2T}\,\frac{\partial f}{\partial\lambda} \tag{23}$$

We can simplify this result by noting that from Equation (21), $x'-x-f\,\delta t = \xi$. Making use of this, the final updating rule for the Malliavin weight is:

$$q'_\lambda = q_\lambda + \frac{\xi}{2T}\,\frac{\partial f}{\partial\lambda} \tag{24}$$

where ξ is the *exact same* value that was used for updating the position in Equation (21). Because the value of ξ is the same for the updates of position and of q_λ, the change in q_λ is completely determined by the end points, x and x'. The derivative, $\partial f/\partial\lambda$, should be evaluated at x, since that is the position at which the force is computed in Equation (21). Since ξ in Equation (21) is a random variate uncorrelated with x, averaging Equation (24) shows that $\langle q'_\lambda\rangle = \langle q_\lambda\rangle$. As the initial condition is $q_\lambda = 0$, this means that $\langle q_\lambda\rangle = 0$, as predicted in the previous section. Equation (24) is essentially the same as that derived in [8].

If we differentiate Equation (23) with respect to a second parameter, μ, we get:

$$\frac{\partial^2 \ln W(x \to x')}{\partial\lambda\partial\mu} = \frac{(x' - x - f\,\delta t)}{2T}\frac{\partial^2 f}{\partial\lambda\partial\mu} - \frac{\delta t}{2T}\frac{\partial f}{\partial\lambda}\frac{\partial f}{\partial\mu} \tag{25}$$

Hence, the updating rule for the second order Malliavin weight can be written as:

$$q'_{\lambda\mu} = q_{\lambda\mu} + \frac{\xi}{2T}\frac{\partial^2 f}{\partial\lambda\partial\mu} - \frac{\delta t}{2T}\frac{\partial f}{\partial\lambda}\frac{\partial f}{\partial\mu} \tag{26}$$

where, again, ξ is the exact same value as that used for updating the position in Equation (21). If we average Equation (26) over replicate simulation runs, we find $\langle q'_{\lambda\mu}\rangle = \langle q_{\lambda\mu}\rangle - (\delta t/2T)(\partial f/\partial\lambda)(\partial f/\partial\mu)$. Hence, the mean value, $\langle q_{\lambda\mu}\rangle$, drifts in time, unlike $\langle q_\lambda\rangle$ or $\langle q_\mu\rangle$. However, one can show that the mean value of the sum, $\langle(q_{\lambda\mu} + q_\lambda q_\mu)\rangle$, is constant in time and equal to zero, as long as, initially, $q_\lambda = q_\mu = 0$.

Now, let us consider the simplest case of a particle in a linear force field, $f = -\kappa x + h$ (also discussed in [8]). This corresponds to a harmonic trap with the potential $U = \frac{1}{2}\kappa x^2 - hx$. We let the particle start from x_0 at $t = 0$ and track its time-dependent relaxation to the steady state. We shall set $T = 1$ for simplicity. The Langevin equation can be solved exactly for this case, and the mean position evolves according to:

$$\langle x(t)\rangle = x_0 e^{-\kappa t} + \frac{h}{\kappa}(1 - e^{-\kappa t}) \tag{27}$$

We suppose that we are interested in derivatives with respect to both h and κ, for a "baseline" parameter set in which κ is finite, but $h = 0$. Taking derivatives of Equation (27) and setting $h = 0$, we find:

$$\frac{\partial\langle x(t)\rangle}{\partial h} = \frac{1 - e^{-\kappa t}}{\kappa}, \quad \frac{\partial\langle x\rangle(t)}{\partial\kappa} = -x_0 t e^{-\kappa t}, \quad \frac{\partial^2\langle x(t)\rangle}{\partial h\partial\kappa} = \frac{te^{-\kappa t}}{\kappa} - \frac{1 - e^{-\kappa t}}{\kappa^2} \tag{28}$$

We now show how to compute these derivatives using Malliavin weight sampling. Applying the definitions in Equations (24) and (26), the Malliavin weight increments are:

$$q'_h = q_h + \frac{\xi}{2}, \quad q'_\kappa = q_\kappa - \frac{x\,\xi}{2}, \quad q'_{h\kappa} = q_{h\kappa} + \frac{x\,\delta t}{2} \tag{29}$$

and the position update itself is:

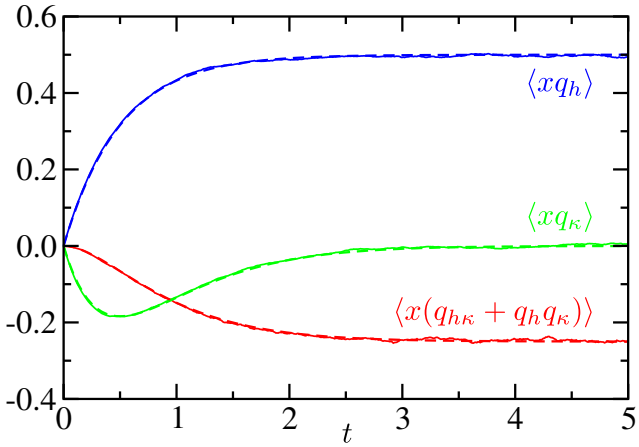
$$x' = x - \kappa x\,\delta t + \xi \tag{30}$$

We track these Malliavin weights in our simulation and use them to calculate derivatives according to:

$$\frac{\partial\langle x(t)\rangle}{\partial h} = \langle x(t)q_h(t)\rangle, \quad \frac{\partial\langle x(t)\rangle}{\partial\kappa} = \langle x(t)q_\kappa(t)\rangle, \quad \frac{\partial^2\langle x(t)\rangle}{\partial h\partial\kappa} = \langle x(t)(q_{h\kappa}(t) + q_h(t)q_\kappa(t))\rangle \tag{31}$$

Equations (29)–(31) have been coded up as a MATLAB script, described in Section 5. A typical result generated by running this script is shown in Figure 1. Equations (29) and (30) are iterated with $\delta t = 0.01$ up to $t = 5$, for a trap strength $\kappa = 2$ and initial position $x_0 = 1$. The weighted averages in Equation (31) are evaluated as a function of time, for $N = 10^5$ samples, as in Equation (10). These results are shown as the solid lines in Figure 1. The dashed lines are theoretical predictions for the time dependent derivatives from Equation (28). As can be seen, the agreement between the time-dependent derivatives and the Malliavin weight averages is very good.

Figure 1. Time-dependent derivatives, $\partial\langle x\rangle/\partial h$ (top curve, blue), $\partial\langle x\rangle/\partial\kappa$ (middle curve, green) and $\partial^2\langle x\rangle/\partial h\partial\kappa$ (bottom curve, red). Solid lines (slightly noisy) are the Malliavin weight averages as indicated in the Figure, generated by running the MATLAB script described in Section 5. Dashed lines are theoretical predictions from Equation (28).

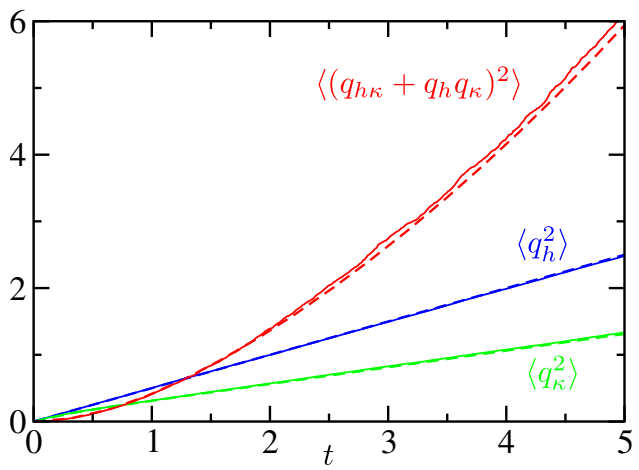


As discussed briefly above, in this procedure, the sampling error in the computation of $\partial\langle A(t)\rangle/\partial\lambda$ is expected to grow with time. Figure 2 shows the mean square Malliavin weight as a function of time for the same problem. For the first order weights, q_h and q_κ, the growth rate is typically linear in time. Indeed, from Equation (29), one can prove that in the limit $\delta t \to 0$ (see Section 5):

$$\frac{d\langle q_h^2\rangle}{dt} = \frac{1}{2}, \quad \frac{d\langle q_\kappa^2\rangle}{dt} = \frac{\langle x^2\rangle}{2} \tag{32}$$

Thus, q_h behaves exactly as a random walk, as should be obvious from the updating rule. The other weight, q_κ, also ultimately behaves as a random walk, since $\langle x^2\rangle = 1/\kappa$ in steady state (from equipartition). Figure 2 also shows that the second order weight, $q_{h\kappa}$, grows superdiffusively; one can show that, eventually, $\langle(q_{h\kappa}+q_hq_\kappa)^2\rangle \sim t^2$, although the transient behaviour is complicated. Full expressions are given in Section 5. This suggests that computation of second order derivatives is likely to suffer more severely from statistical sampling problems than the computation of first order derivatives.

Figure 2. Growth of mean square Malliavin weights with time. The solid lines are from simulations and the dashed lines are from Equation (35) in the Appendix. Parameters are as for Figure 1.



5. Conclusions

In this paper, we have provided an outline of the generic use of Malliavin weights for sampling derivatives in stochastic simulations, with an emphasis on practical aspects. The usefulness of MWS for a particular simulation scheme hinges on the simplicity, or otherwise, of constructing the propagator, $W(S \to S')$, which fixes the updating rule for the Malliavin weights according to Equation (9). The propagator is determined by the algorithm used to implement the stochastic equations of motion; MWS may be easier to implement for some algorithms than for others. We note, however, that there is often some freedom of choice about the algorithm, such as the choice of a stochastic thermostat in molecular dynamics, or the order in which update steps are implemented. In these cases, a suitable choice may simplify the construction of the propagator and facilitate the use of Malliavin weights.

Acknowledgments

Rosalind J. Allen is supported by a Royal Society University Research Fellowship.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Bell, D.R. *The Malliavin Calculus*; Dover: Mineola, NY, USA, 2006.
2. Nualart, D. *The Malliavin Calculus and Related Topics*; Springer: New York, NY, USA, 2006.
3. Fournié, E.; Lasry, J.M.; Lebuchoux, J.; Lions, P.L.; Touzi, N. Applications of Malliavin calculus to Monte Carlo methods in finance. *Financ. Stoch.* **1999**, *3*, 391–412.
4. Plyasunov, A.; Arkin, A.P. Efficient stochastic sensitivity analysis of discrete event systems. *J. Comput. Phys.* **2007**, *221*, 724–738.
5. Berthier, L. Efficient measurement of linear susceptibilities in molecular simulations: Application to aging supercooled liquids. *Phys. Rev. Lett.* **2007**, *98*, 220601.
6. Chen, N.; Glasserman, P. Malliavin Greeks without Malliavin calculus. *Stoch. Proc. Appl.* **2007**, *117*, 1689–1723.
7. Warren, P.B.; Allen, R.J. Steady-state parameter sensitivity in stochastic modeling via trajectory reweighting. *J. Chem. Phys.* **2012**, *136*, 104106.
8. Warren, P.B.; Allen, R.J. Malliavin weight sampling for sensitivity coefficients in Brownian dynamics simulations. *Phys. Rev. Lett.* **2012**, *109*, 250601.
9. Warren, P.B.; ten Wolde, P.R. Chemical models of genetic toggle switches. *J. Phys. Chem. B* **2005**, *109*, 6812–6823.

Appendix

MATLAB Script

The MATLAB script in Listing 1 was used to generate the results shown in Figure 1. It implements Equations (29)–(31) above, making extensive use of the compact MATLAB syntax for array operations, for instance, invoking '.*' for element-by-element multiplication of arrays.

Listing 1. MATLAB script used to generate Figure 1.

```
clear all
randn('seed', 12345);
kappa = 2; x0 = 1; tend = 5; dt = 0.01; nsamp = 10^5;
npt = round(tend/dt) + 1;
t = (0:npt-1)' * dt;
x = zeros(npt, 1); xi = zeros(npt, 1);
qh = zeros(npt, 1); qk = zeros(npt, 1); qhk = zeros(npt, 1);
x_av = zeros(npt, 1); xqh_av = zeros(npt, 1);
xqk_av = zeros(npt, 1); xqhk_av = zeros(npt, 1);
for samp = 1:nsamp
 x(1) = x0; qh(1) = 0; qk(1) = 0; qhk(1) = 0;
 xi = randn(npt, 1) * sqrt(2*dt);
 for i = 1:npt-1
 x(i+1) = x(i) - kappa*x(i)*dt + xi(i);
 qh(i+1) = qh(i) + 0.5*xi(i);
 qk(i+1) = qk(i) - 0.5*x(i)*xi(i);
 qhk(i+1) = qhk(i) + 0.5*x(i)*dt;
 end
 x_av = x_av + x;
 xqh_av = xqh_av + x.*qh;
 xqk_av = xqk_av + x.*qk;
 xqhk_av = xqhk_av + x.*(qhk + qh.*qk);
end
x_av = x_av / nsamp; xqh_av = xqh_av / nsamp;
xqk_av = xqk_av / nsamp; xqhk_av = xqhk_av / nsamp;
hold on
plot(t, x_av, 'k'); plot(t, xqh_av, 'b');
plot(t, xqk_av, 'g'); plot(t, xqhk_av, 'r');
plot(t, x0*exp(-kappa*t), 'k--')
plot(t, (1-exp(-kappa*t))/kappa, 'b--')
plot(t, -x0*t.*exp(-kappa*t), 'g--')
plot(t, t.*exp(-kappa*t)/kappa-(1-exp(-kappa*t))/(kappa^2), 'r--')
result = [t x_av xqh_av xqk_av xqhk_av];
save('bd1d.dat', '-ascii', 'result');
```

Here is a brief explanation of the script. *Lines 1–3* initialise the problem and the parameter values. *Lines 4* and *5* calculate the number of points in a trajectory and initialise a vector containing the time coordinate of each point. *Lines 6–9* set aside storage for the actual trajectory, Malliavin weights and cumulative statistics. *Lines 10–23* implement a pair of nested loops, which are the kernel of the simulation. Within the outer (trajectory sampling) loop, *Line 11* initialises the particle position and Malliavin weights, *Line 12* precomputes a vector of random displacements (Gaussian random variates) and *Lines 13–18* generate the actual trajectory. Within the inner (trajectory generating loop), *Lines 14–17* are a direct implementation of Equations (29) and (30). After each individual trajectory has been generated, the cumulative sampling step implied by Equation (31) is done in *Lines 19–22*; after all the trajectories have been generated, these quantities are normalised in *Lines 24* and *25*. Finally, *Lines 26–32* generate a plot similar to Figure 1 (albeit with the addition of $\langle x \rangle$), and *Lines 33* and *34* show how the data can be exported in tabular format for replotting using an external package.

Listing 1 is complete and self-contained. It will run in either MATLAB or Octave. One minor comment is perhaps in order. The choice was made to precompute a vector of Gaussian random variates, which are used as random displacements to generate the trajectory and update the Malliavin weights. One could equally well generate random displacements on-the-fly, in the inner loop. For this one-dimensional problem, storage is not an issue, and it seems more elegant and efficient to exploit the vectorisation capabilities of MATLAB. For a more realistic three-dimensional problem, with many particles (and a different programming language), it is obviously preferable to use an on-the-fly approach.

Selected Analytic Results

Here, we present analytic results for the growth in time of the mean square Malliavin weights. We can express the rate of growth of the mean of a generic function, $f(x, q_h, q_\kappa, q_{h\kappa})$, as:

$$\frac{d\langle f \rangle}{dt} = \lim_{\delta t \to 0} \frac{\langle f(x', q'_h, q'_\kappa, q'_{h\kappa}) - f(x, q_h, q_\kappa, q_{h\kappa}) \rangle}{\delta t} \tag{33}$$

where, on the right-hand side (RHS), the values of x', q'_h, q'_κ and $q_{h\kappa}$ are substituted from the updating rules in Equations (29) and (30). In calculating the RHS average, we note that the distribution of ξ is a Gaussian independent of the position and Malliavin weights, and thus, one can substitute $\langle \xi \rangle = 0$, $\langle \xi^2 \rangle = 2\,\delta t$, $\langle \xi^3 \rangle = 0$, $\langle \xi^4 \rangle = 12\,\delta t^2$, *etc.*. Proceeding in this way, with judicious choices for f, one can obtain the following set of coupled ordinary differential equations (ODEs):

$$\begin{gathered}
\frac{d\langle q_h^2 \rangle}{dt} = \frac{1}{2}\,, \quad \frac{d\langle q_\kappa^2 \rangle}{dt} = \frac{\langle x^2 \rangle}{2}\,, \quad \frac{d\langle x^2 \rangle}{dt} + 2\kappa\langle x^2 \rangle = 2\,, \quad \frac{d\langle x q_h \rangle}{dt} + \kappa\langle x q_h \rangle = 1 \\
\frac{d\langle x^2 q_h^2 \rangle}{dt} + 2\kappa\langle x^2 q_h^2 \rangle = 2\langle q_h^2 \rangle + 4\langle x q_h \rangle + \frac{\langle x^2 \rangle}{2}\,, \quad \frac{d\langle x q_h q_\kappa \rangle}{dt} + \kappa\langle x q_h q_\kappa \rangle = -\langle x q_h \rangle - \frac{\langle x^2 \rangle}{2} \\
\frac{d\langle (q_{h\kappa} + q_h q_\kappa)^2 \rangle}{dt} = \frac{\langle q_\kappa^2 \rangle}{2} - \langle x q_h q_\kappa \rangle + \frac{\langle x^2 q_h^2 \rangle}{2} \quad \left(= \frac{\langle (q_\kappa - x q_h)^2 \rangle}{2} \right)
\end{gathered} \tag{34}$$

Some of these have already been encountered in the main text. The last one is for the desired mean square second order weight. The ODEs can be solved with the initial conditions that at $t = 0$, all averages involving Malliavin weights vanish, but $\langle x^2 \rangle = x_0^2$. The results include *inter alia*

$$\langle q_h^2 \rangle = \frac{t}{2}\,, \quad \langle q_\kappa^2 \rangle = \frac{t}{2\kappa} + \frac{(\kappa x_0^2 - 1)(1 - e^{-2\kappa t})}{4\kappa^2}$$

$$\langle (q_{h\kappa} + q_h q_\kappa)^2 \rangle = \frac{2\kappa^2 t^2 + (19 + \kappa x_0^2)\kappa t + 2\kappa x_0^2 - 34}{8\kappa^3} + \frac{2\kappa t + 10 - \kappa x_0^2}{2\kappa^3}\, e^{-\kappa t} + \frac{(1 - \kappa x_0^2)\kappa t + 2\kappa x_0^2 - 6}{8\kappa^3}\, e^{-2\kappa t} \tag{35}$$

These are shown as the dashed lines in Figure 2. The leading behaviour of the last as $t \to \infty$ is:

$$\langle (q_{h\kappa} + q_h q_\kappa)^2 \rangle = \frac{t^2}{4\kappa} + \text{subdominant terms} \tag{36}$$

however, the approach to the pure asymptotic limit is slow.