

A modified diffusion Monte Carlo and other ensemble sampling methods

Jonathan Weare (U of Chicago)
with
Martin Hairer (U of Warwick)

February 14, 2012

Diffusion Monte Carlo

The original motivation for DMC was to compute averages with respect to the ground state eigenfunction of

$$\mathcal{H}\psi = -\frac{1}{2}\Delta\psi + u\psi.$$

Notice that if ψ solves the PDE

$$\partial_t\psi = -\mathcal{H}\psi$$

then, for example,

$$\lambda_0 \approx \frac{\int \mathcal{H}\psi(t, x) dx}{\int \psi(t, x) dx} \approx \frac{\int \psi(t, x) \mathcal{H}1 dx}{\int \psi(t, x) dx} \approx \frac{\int u(x)\psi(t, x) dx}{\int \psi(t, x) dx}$$

for large t , and

$$\int f(x)\psi(t, x) dx = \mathbf{E} \left[f(W(t)) e^{-\int_0^t u(W(s)) ds} \right].$$

where W is a Brownian motion.

Diffusion Monte Carlo generates an ensemble of points $X_i(t)$ so that

$$\mathbf{E} \left[\sum_{i=1}^{N(t_k)} f(X_i(t_k)) \right] = \mathbf{E} \left[f(X(t_k)) e^{-\sum_{j=1}^k v(X(t_{j-1}), X(t_j))} \right]$$

for any reasonable observable f where $X(t_0), X(t_1), X(t_2), \dots$ is some underlying process.

The ensemble of points evolves in two steps:

- 1 Evolve each point according to the underlying dynamics for one increment.
- 2 To incorporate the additional “weight” factor $e^{-v(X(s-1), X(s))}$ copy particles with large weight and kill those with low weight.

DMC proceeds from an ensemble $\{X_i(t_{k-1})\}$ of size $N(t_{k-1})$ at “time” t_{k-1} to an ensemble of size $N(t_k)$ at time t_k as follows:

- 1: for $i = 1 : N(t_{k-1})$
- 2: evolve the sample $X_i(t_{k-1})$
 to time t_k : $X_i(t_{k-1}) \rightarrow \tilde{X}_i(t_k)$
- 3: generate a random integer $N_i \geq 0$
 with $\mathbf{E}[N_i] = e^{-\nu(X_i(t_{k-1}), \tilde{X}_i(t_k))}$
- 4: add N_i copies of $\tilde{X}_i(t_k)$ to the
 time t_k ensemble
- 5: set $N(t_k) = \sum_{i=1}^{N(t_{k-1})} N_i$

Back to Quantum Monte Carlo:

$$\mathbf{E} \left[f(W(t)) e^{-\int_0^t u(W(s)) ds} \right] \\ \approx \mathbf{E} \left[f(W(t)) e^{-\sum_{j=1}^{\lfloor t/h \rfloor} \frac{1}{2}(u(W(jh)) + u(W((j-1)h))) ds} \right]$$

This is of the form

$$\mathbf{E} \left[f(X(t_k)) e^{-\sum_{j=1}^k v(X(t_{j-1}), X(t_j))} \right]$$

so we can use DMC:

$$\lambda_0 \approx \frac{1}{N(kh)} \sum_{i=1}^{N(kh)} u(X_i(kh))$$

where $X_i(kh)$ for $i = 1, \dots, N(kh)$ is the DMC ensemble at time kh .

Over the last 40 years or so the basic DMC idea has spread.

For example in Sequential Monte Carlo (e.g. particle filters) one transforms N samples $X_i(t_{k-1})$ approximately drawn from some density $p_{k-1}(x)$ to N samples $X_i(t_k)$ approximately drawn from

$$p_k(y) \propto \int \mu(y)p(y|x)p_{k-1}(x)dx$$

where μ is some “weight” function (e.g. from data) and $p(y|x)$ is a transition density.

This is done by

- 1 Sampling $X_i(t_k) \sim p(x | X(t_{k-1}))$
- 2 Weighting each sample by $\mu(X_i(t_k))$
- 3 Resampling from the weighted distribution.

Could argue that popular methods for carrying out 2-3 are based on DMC.

Over the last 40 years or so the basic DMC idea has spread.

For example in Sequential Monte Carlo (e.g. particle filters) one transforms N samples $X_i(t_{k-1})$ approximately drawn from some density $p_{k-1}(x)$ to N samples $X_i(t_k)$ approximately drawn from

$$p_k(y) \propto \int \mu(y)p(y|x)p_{k-1}(x)dx$$

where μ is some “weight” function (e.g. from data) and $p(y|x)$ is a transition density.

This is done by

- 1 Sampling $X_i(t_k) \sim p(x | X(t_{k-1}))$
- 2 Weighting each sample by $\mu(X_i(t_k))$
- 3 Resampling from the weighted distribution.

Could argue that popular methods for carrying out 2-3 are based on DMC.

In the Quantum Monte Carlo application v was specified by the problem. In other applications we may want to choose v to achieve some goal.

What if we choose

$$v(x, y) = G(y) - G(x)?$$

Then DMC would compute

$$\mathbf{E} \left[\sum_{i=1}^{N(t_k)} f(X_i(t_k)) \right] = e^{G(X(0))} \mathbf{E} \left[f(X(t_k)) e^{-G(X(t_k))} \right]$$

or (by redefining f)

$$e^{-G(X(0))} \mathbf{E} \left[\sum_{i=1}^{N(t_k)} f(X_i(t_k)) e^{G(X_i(t_k))} \right] = \mathbf{E} [f(X(t_k))].$$

Recall the branching rule:

3 : generate a random integer $N_i \geq 0$

with $\mathbf{E} [N_i] = e^{-(G(\tilde{X}_i(t_k)) - G(X_i(t_{k-1})))}$

4 : add N_i copies of $\tilde{X}_i(t_k)$ to the
time t_k ensemble

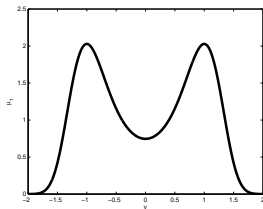
So if G decreases in a step more copies will be created.

By choosing G to be relatively small in a region we can sample that region more thoroughly

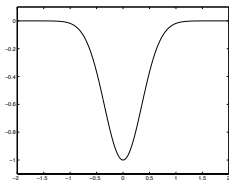
So, for example suppose the underlying dynamics is a discrete sampling (or discrete approximation) $X(kh)$ of

$$dX(t) = -\nabla V(X(t))dt + \sqrt{2kT}dW(t)$$

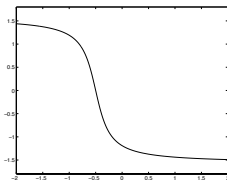
where the invariant measure $e^{-V/kT}$ looks like



Then we might choose



or



if (**left**) we want to compute an average near the low probability saddle

or (**right**) we want to force the system from the left well to the right well.

Unfortunately this won't work.

Note that

$$G(X(kh)) - G(X((k-1)h)) = \mathcal{O}(\sqrt{h})$$

so every h units of time we create or destroy \sqrt{h} samples so if creation and destruction aren't carefully balanced expect bad behavior for small h .

As the time step h is taken smaller and smaller the ensemble size will either blow up or hit zero in a very short time.

Yes... we could only do the killing/cloning step once for every $\mathcal{O}(1/\sqrt{h})$ evolution steps but if the weights degenerate very quickly expect bad results.

A small modification borrowed from another rare event scheme (RESTART/DPR) developed in the computer science community seems to solve the problem

Unfortunately this won't work.

Note that

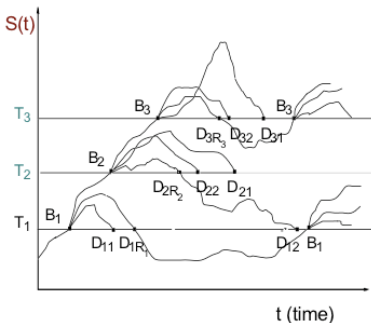
$$G(X(kh)) - G(X((k-1)h)) = \mathcal{O}(\sqrt{h})$$

so every h units of time we create or destroy \sqrt{h} samples so if creation and destruction aren't carefully balanced expect bad behavior for small h .

As the time step h is taken smaller and smaller the ensemble size will either blow up or hit zero in a very short time.

Yes... we could only do the killing/cloning step once for every $\mathcal{O}(1/\sqrt{h})$ evolution steps but if the weights degenerate very quickly expect bad results.

A small modification borrowed from another rare event scheme (RESTART/DPR) developed in the computer science community seems to solve the problem



In RESTART/DPR:

- 1 The state space is first partitioned into cells.
- 2 When an ensemble member crosses from one region to another a certain number of new ensemble members are born.
- 3 These new ensemble members are restricted in what cells they are allowed to visit... they are killed if they try to visit a cell that is off limits to them.

We'll give each of our DMC particles $X_i(t_k)$ a "ticket" $\vartheta_i(t_k)$ that tells the trajectory where (in the range of values of G) it's allowed to visit. When $G(x)$ falls below the ticket the particle is killed.

The branching step is now

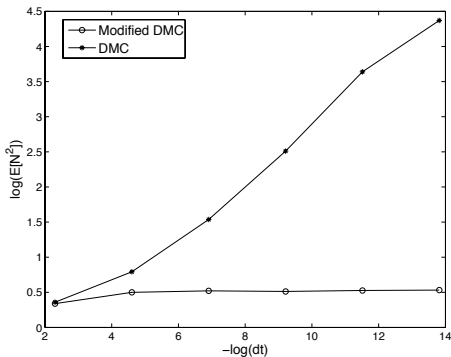
$$3: \text{ if } e^{-G(\tilde{X}_i(t_k))} \geq \vartheta_i \text{ generate } N_i \geq 1 \text{ with}$$
$$\mathbf{E}[N_i] = \min \left\{ 1, e^{-(G(\tilde{X}_i(t_k)) - G(X_i(t_{k-1})))} \right\}$$

otherwise $N_i = 0$

4: add N_i copies of $\tilde{X}_i(t_k)$ to the
time t_k ensemble

5: for each **new** copy generate a ticket
 $\vartheta \sim \mathcal{U}(e^{-G(X_i(t_{k-1}))}, e^{-G(\tilde{X}_i(t_k))})$

The initial ticket $\vartheta_1(t_0)$ is drawn from $\mathcal{U}(0, e^{-G(X(t_0))})$



In contrast to the situation for DMC we're working on exponential bounds ($\mathbf{E}[\exp(\lambda_t N(t))] < \infty$).

To compute averages of the form

$$\mathbf{E} \left[f(X(t_k)) e^{-\sum_{j=1}^k v(X(t_{j-1}), X(t_j))} \right]$$

use the branching step (setting $P_i = e^{-v(X_i(t_{k-1}), \tilde{X}_i(t_k))}$)

3: if $P_i \geq \vartheta_i$ generate $N_i \geq 1$ with

$$\mathbf{E}[N_i] = \min\{1, P_i\}$$

otherwise $N_i = 0$

4: add N_i copies of $\tilde{X}_i(t_k)$ to the
time t_k ensemble

5: for each **new** copy generate a ticket

$$\vartheta \sim \mathcal{U}(P_i^{-1}, 1)$$

discount ticket of surviving particles by P_i^{-1}

The initial ticket $\vartheta_1(t_0)$ is drawn from $\mathcal{U}(0, 1)$

As with DMC the new algorithm produces an unbiased estimate

$$\sum_{i=1}^{N(t_k)} f(X_i(t_k)) \quad \text{of} \quad \mathbf{E} \left[f(X(t_k)) e^{-\sum_{j=1}^k v(X(t_{j-1}), X(t_j))} \right]$$

You get back DMC if you re-draw all tickets from $\mathcal{U}(0, 1)$ at each step.

This allows us to prove that (basically in any setting):

Theorem

The new estimator has lower variance than the old estimator.

It's expected cost is easily seen to be the same so the new algorithm is unambiguously superior.

As we have seen, in some cases the improvement is dramatic.

For example, with $v(x, y) = G(y) - G(x)$, the algorithm seems to have a continuous time limit (as $h \rightarrow 0$).

Characterising this limit and proving convergence as $h \rightarrow 0$ is the subject current work.

As are Quantum Monte Carlo applications.

When using the algorithm with $v(x, y) = G(y) - G(x)$ to compute

$$\mathbf{E}[f(X(t_k))]$$

it's natural to ask what is the optimal choice of G .

One has to consider the effect of the choice of G on both the workload and variance of the resulting estimator.

We have not pursued this but we expect that in the small temperature limit an optimal strategy could be identified by following arguments of Dean and Dupuis for DPR

A simple rare event example:

$$dX(t) = -\nabla V(X(t))dt + \sqrt{2kT}dW(t)$$

Starting from the lower well and running for 1 unit of time.

We use our modified DMC scheme with

$$v(x, y) = G(y) - G(x), \quad G(x) = -\lambda \|x - x_A\|$$

where x_A is the minimum in the lower basin. We want to compute

$$P_{x_A}(X(1) \in B), \quad B = \{x : \|x - x_B\| < 0.25\}.$$

λ is chosen so that the expected number of particles ending in B is close to 1.

kT	λ	estimate	variance \times workload	brute force variance
16	5	0.5133	0.3357	0.2499
8	15	0.2839×10^{-1}	0.5519×10^{-2}	0.2758×10^{-1}
4	25.5	0.4813×10^{-5}	0.1521×10^{-8}	0.4813×10^{-5}
2	33	0.1262×10^{-13}	0.2133×10^{-23}	0.1262×10^{-13}

A more challenging example.

The 7 discs interact with each other through

$$V(x) = \sum_{i < j} 4 \left(\|x_i - x_j\|^{-12} - \|x_i - x_j\|^{-6} \right)$$

at $kT = 0.1$.

Using

$$v(x, y) = G(y) - G(x), \quad G(x) = \lambda \min_{i \geq 2} \{\|x_i\|\}$$

where x_1 is the point at the center.

We compute for example

$$P(X(2) \in B) \approx 7.79 \times 10^{-5}$$

where B is the event that $\min_{i \geq 2} \{\|x_i\|\} < 0.1$.

The workload is

$$\frac{1}{2} \sum_{j=1}^k N(jh)h = 6.9 \quad (h = 0.001)$$

λ is adjusted so that the expected number of particles ending in B is close to 1.