Improved diffusion Monte Carlo

Jonathan Weare

University of Chicago

with

Martin Hairer (U of Warwick)

October 4, 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

$$\psi(t,x) = \sum e^{-\lambda_i t}\phi_i(x)$$

So if you can compute averages with respect to $\frac{\psi}{\int \psi dx}$ for large t then you can approximate averages with respect to $\frac{\phi_0}{\int \phi_0 dx}$.

We have the Feynman–Kac representation

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

where W is a Brownian motion.

Using the trapezoidal approximation of the integral:

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

This is of the general form

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

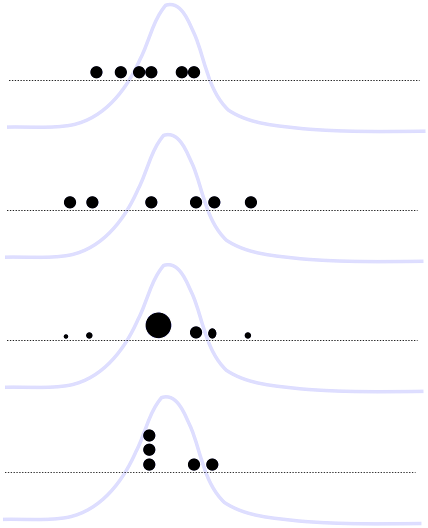
Diffusion Monte Carlo generates an ensemble of points $X_i(t)$ so that

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

for any reasonable observable f where $X(t_0), X(t_1), X(t_2), \ldots$ 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 and 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 $t_k$ :   $X_i(t_{k-1}) \to \tilde{X}_i(t_k)$
3 :     generate a random integer  $N_i \geq 0$
           with $\mathbf{E}[N_i] = e^{-v(X_i(t_{k-1}), \tilde{X}_i(t_k))}$
4 :     add $N_i$ copies of $\tilde{X}_i(t_k)$ to the
         time $t_k$ ensemble
5 : set $N(t_k) = \sum_{i=1}^{N(t_{k-1})} N_i$
```

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

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

where $e^{-v(y)}$ 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 $e^{-v(X_i(t_k))}$
3. Resampling from the weighted distribution.

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

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

where $e^{-v(y)}$ 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 $e^{-v(X_i(t_k))}$
3. Resampling from the weighted distribution.

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)

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

Recall the branching rule:

3 : generate a random integer $N_i \geq 0$
with $\mathbf{E}[N_i] = e^{-\left(G(\tilde{X}_i(t_k)) - G(X_i(t_{k-1}))\right)}$

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

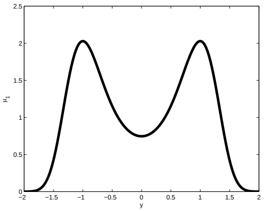
So if G decreases in a step more copies will be created.

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

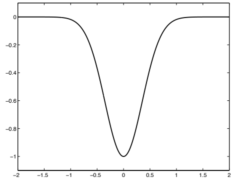
So, for example suppose the underlying dynamics is a discrete sampling (or discrete approximation) $X(kdt)$ of

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

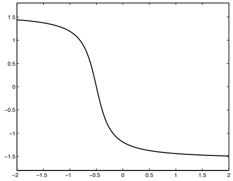
where the invariant measure $e^{-V/\mu}$ looks like



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

$$G(X(kdt)) - G(X((k-1)dt)) = \mathcal{O}(\sqrt{dt})$$

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

As the time step *dt* 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 $\mathcal{O}(1/\sqrt{dt})$ 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

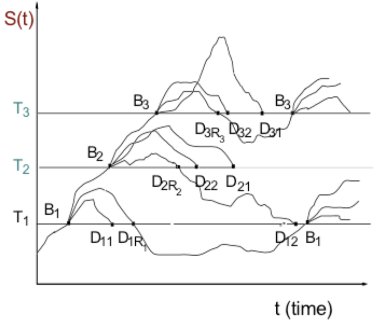
$$G(X(kdt)) - G(X((k-1)dt)) = \mathcal{O}(\sqrt{dt})$$

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

As the time step *dt* 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 $\mathcal{O}(1/\sqrt{dt})$ 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 it's allowed to visit. When e^{-v} is below the ticket the particle is killed.
The branching step is now

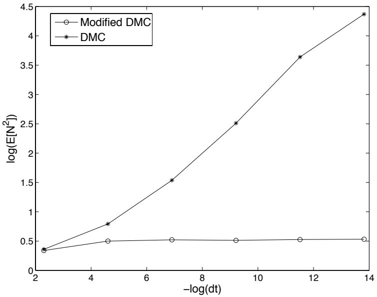
```
3 : if  $e^{-v(X_i(t_{k-1}),\tilde{X}_i(t_k))} \geq \vartheta_i$  generate  $N_i \geq 1$  with
        $\mathbf{E}[N_i] = \min\left\{1, e^{-v(X_i(t_{k-1}),\tilde{X}_i(t_k))}\right\}$
    otherwise  $N_i = 0$
4 : add  $N_i$  copies of  $\tilde{X}_i(t_k)$  to the
        time  $t_k$  ensemble
5 : for each **new** copy generate a ticket
        $\vartheta \sim \mathcal{U}(e^{v(X_i(t_{k-1}),\tilde{X}_i(t_k))}, 1)$
    tickets of **surviving** particles
        are discounted by  $e^{v(X_i(t_{k-1}),\tilde{X}_i(t_k))}$
```

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

Consider the simple dynamics

$$X(kdt) = X((k-1)dt) + \sqrt{dt}\xi_k \qquad \xi_k \sim N(0,1)$$

with $\quad G(x) = x.$



For the new scheme:

$$\sup_{\substack{dt>0 \\ kdt=t}} \mathbf{E}\left[(N(t))^p\right] < \infty$$

for all powers p.

As with DMC the new algorithm produces an unbiased estimate

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

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):

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.

In settings in which DMC fails dramatically **the improved algorithm has a nice continuous time limit.**

We call the limiting object the **Brownian fan.**

The Brownian fan is constructed recursively with each generation being a realization of a poisson point process on an excursion space with intensity depending on the previous generation.

So the new algorithm is better in all settings and in some cases the improvement is really dramatic

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

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

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.

But these strategies will involve finding, e.g. approximate transition paths.

Since you can't expect to do that very accurately you'd like to see the algorithm behave well with more naive design choices.

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

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

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

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

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 rare event example:

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

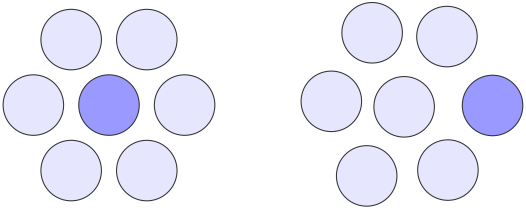


Figure : (**Left**) Initial configuration x^A. (**Right**) A typical snapshot after the transition.

The 7 discs interact with each other through

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

Using

$$v(x,y) = G(y) - G(x), \qquad G(x) = \frac{\lambda}{\mu} \min_{i \geq 2} \left\{ \left\| x_i - \frac{1}{7} \sum_j x_j \right\| \right\}.$$

where x_1 is the point at the center.

We approximate $P(X(2) \in B)$ where B is the event that $\min_{i\geq 2} \left\{ \left\| x_i - \frac{1}{7} \sum_j x_j \right\| \right\} < 0.1$.

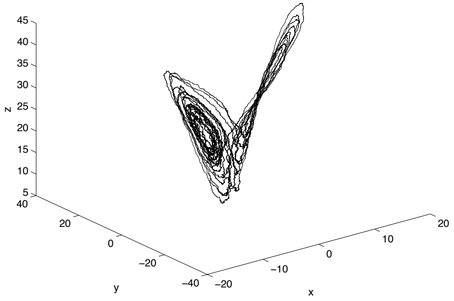
μ	λ	estimate	workload	variance $\times$workload	brute force variance
0.4	1.9	1.125×10^{-2}	6.451	4.732×10^{-3}	1.112×10^{-2}
0.2	1.3	2.340×10^{-3}	5.818	2.344×10^{-4}	2.335×10^{-3}
0.1	1	7.723×10^{-5}	6.936	7.473×10^{-7}	7.722×10^{-5}
0.05	0.85	9.290×10^{-8}	15.42	1.002×10^{-11}	9.290×10^{-8}
0.025	0.8	1.129×10^{-13}	102.4	1.311×10^{-21}	1.129×10^{-13}

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

A filtering example:
The solution to

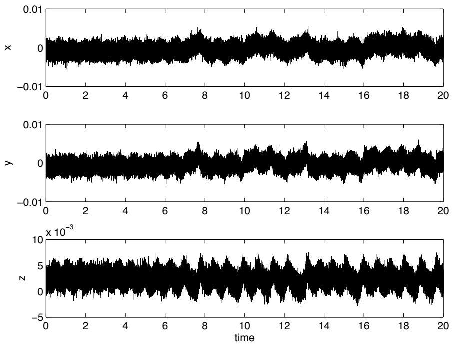
$$dX_1(t) = 10(X_1(t) - X_2(t))dt + \sqrt{2}dW_1(t)$$
$$dX_2(t) = (X_1(t)(28 - X_3(t)) - X_2(t))dt + \sqrt{2}dW_2(t)$$
$$dX_3(t) = (X_1(t)X_2(t) - \frac{8}{3}X_3(t))dt + \sqrt{2}dW_3(t)$$

is hidden from us.



We see only

$$dH(t) = \begin{pmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \end{pmatrix} dt + 0.1\, dB(t)$$



Our task is to estimate the hidden signal by computing

$$\mathbf{E}\left[(X_1(t), X_2(t), X_3(t)) \mid \mathcal{F}_t^H\right]$$

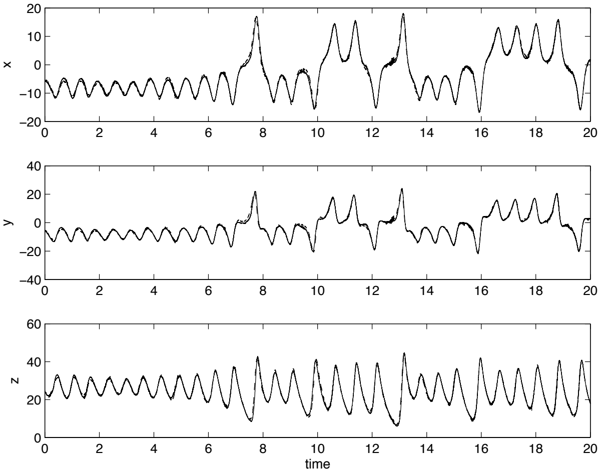
After discretizing a DMC method for sampling from the conditional distribution of the hidden process given the observations becomes

```
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 $P_i^{-1}$
```

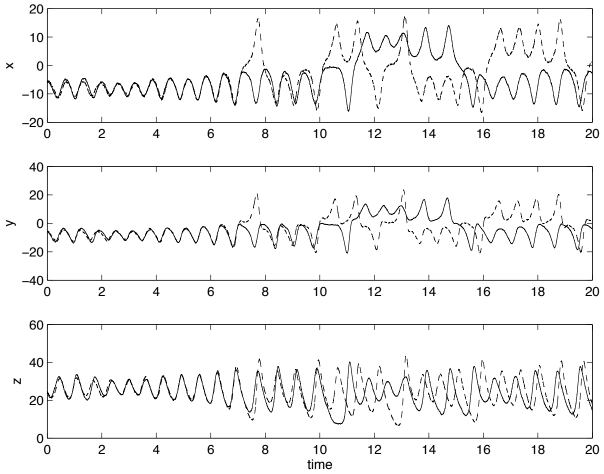
The initial ticket $\vartheta_1(t_0)$ is drawn from $\mathcal{U}(0,1)$ where now

$$P_i = \frac{exp\left(-\frac{\|X_i(t)dt - dH(t)\|^2}{0.02dt}\right)}{\sum_j exp\left(-\frac{\|X_j(t)dt - dH(t)\|^2}{0.02dt}\right)}\overline{N}$$

With 10 particles:



What if we'd done the same thing without the tickets (i.e. standard DMC)


x
y
z
time