Fast Randomized Iteration: Diffusion Monte Carlo through the Lens of Numerical Linear Algebra

Raul Enrique Platero
University of Illinois at Urbana-Champaign

Recent research in numerical linear algebra has been exploring randomization techniques to increase efficiency. One attempt is to look at Monte Carlo methods. A recent paper published in 2017 [1] looks at diffusion Monte Carlo methods for applications to common numerical linear algebra problems such as matrix exponentiation, linear systems and eigenvalue problems. Diffusion Monte Carlo methods are a set of methods that study zero-temperature quantum systems. One common application of diffusion Monte Carlo is for computing ground state's energy of electrons which is equivalent to computing the smallest eigenpair. The advantage that Monte Carlo methods have over typical numerical linear algebra techniques is that they can work for problems of far larger dimensions. This paper has been motivated by the recent application of diffusion Monte Carlo schemes to matrices as large as $10^{108} \times 10^{108}$ [2].

In order to understand how to apply diffusion Monte Carlo to general linear problems, we will consider an outline of the derivation for these methods. Diffusion Monte Carlo starts with the imaginary-time Schödinger equation:

$$\partial_t v = -\mathcal{H} v,$$

where $\mathcal{H}$ is the Hamiltonian operator and v is a vector. This equation can be solved using the following iterative method:

$$\lambda_t = -\frac{1}{\epsilon} \log \int e^{-\epsilon\mathcal{H}} v_{t-1}(x)dx \quad \text{and} \quad v_t = \frac{e^{-\epsilon\mathcal{H}} v_{t-1}}{\int e^{-\epsilon\mathcal{H}} v_{t-1}(x)dx}.$$

The end goal of these methods is to introduce randomizations and include a sum of weights at a set of given points. We will denote the random approximation of v_t as V_t^m. The first step is to approximate the integral using the following randomization:

$$\int f(x)[e^{\frac{\epsilon}{2}\Delta}\delta_y](x)dx = \mathbf{E}_y[f(B_\epsilon)],$$

a special case of the Feynman-Kac formula, where f is a test function, B_s is a standard Brownian motion evaluated at time $s \geq 0$. So, now if we let

$$\tilde{V}_t^m = K_\epsilon V_{t-1}^m,$$

where K_ϵ is the discretization of $e^{-\epsilon\mathcal{H}}$. We can then rewrite the iterative method as

$$V_{t+1}^m = \frac{\tilde{V}_{t+1}^m}{\int \tilde{V}_{t+1}^m(x)dx} = \sum_{j=1}^{m} W_{t+1}^{(j)}\, \delta_{\xi_{t+1}^{(j)}}^{(j)},$$

where weights are recursively defined

$$W_{t+1}^{(j)} = \frac{e^{\frac{\epsilon}{2}(U(\epsilon_{t+1}^{(j)})+U(X_t^{(j)}))}W_t^{(j)}}{\sum_{\ell=1}^{m} e^{\frac{\epsilon}{2}(U(\epsilon_{t+1}^{(\ell)})+U(X_t^{(\ell)}))}W_t^{(\ell)}}.$$

After the first step, we get the desired form for V_t^m. However, in order to decrease the error introduced in the first step of randomization, we need to control the variance.

The second step is to control the variance. Error is introduced because the points $\xi_j^{(j)}$ do not reference the potential U (sampled from m independent Brownian motions). In diffusion Monte Carlo, this can be accomplished by removing the points with very small weights and duplicate points with large weights such that

$$\mathbf{E}[Y_t^m \mid V_t^m] = V_t^m,$$

where Y_t^m is an approximation of V_t^m. Overall, the cost of each iteration is dominated by the first step to approximating the iterative method($O(dm)$, where d is dimension of problem and m are the number of nonzero entries in solution).

The method developed in this paper, fast randomized iteration, follows closely to diffusion Monte Carlo except the focus is on the second step discussed above. Fast randomized iteration:

$$V_{t+1}^m = \mathcal{M}(\Phi_t^m(V_t^m)),$$

where $\Phi_t^m : \mathbb{C}^n \to \mathbb{C}^n$ satisfying $\mathbf{E}[\Phi_t^m(v)] = v$ and $\mathcal{M}$ is an operator. The second step for approximating the solution in diffusion Monte Carlo is applied through the function Φ called the compression rule or scheme. Again, the purpose of the compression rule is to control the variance along with being a sparsifier of the solution vector. One simple example of a compression rule is:

$$(\Phi_t^m(v))_j = \begin{cases} N_j \frac{||v||_1}{m} \frac{v_j}{|v|_j} & \text{if } |v_j| > 0 \\ 0 & \text{if } |v_j| = 0 \end{cases},$$

where each N_j is a random, nonnegative, integer. Note that this compression rule does not increase sparsity as error increases which drives the efficiency of fast randomized iteration. The method used by the authors of this paper is demonstrated in Algorithm 1.

The main competitor involves a simple scheme that is called truncation-by-size (TbS). Tbs is a thresholding method that sets every element of the solution vector to zero if its index is greater than the index of the largest element of the solution vector. Some of the numerical results found by comparing fast randomized iteration to TbS are presented in Figure 5 and Figure 6. Results are based on a matrix arising in the spectral gap of a diffusion process governing the evolution of a system of up to five two-dimensional particles. The resulting matrices are of size up to $10^{20} \times 10^{20}$. Figure 5 has matrix size $10^{16} \times 10^{16}$ and varies the number of nonzero entries in the solution vector. Note that for this specific problem, fast randomized iteration converges to the same solution independent of the number of nonzeros used. In Figure 6, the results show only fast randomized iteration with a matrix size of $10^{20} \times 10^{20}$.

Algorithm 1 A simple compression rule.

```
**Data:** $v \in \mathbb{C}^n$ with all nonzero entries, $m \in \mathbb{N}$.
**Result:** $V = \Phi^m(v) \in \mathbb{C}^n$ with at most $m$ nonzero entries.
$\tau_v^m = 0$;
$V = 0$;
$r = \|v\|_1/m$;
$\sigma_1 = \arg\max_i\{|v_i|\}$;
**while** $|v_{\sigma_{\tau_v^m+1}}| \geq r$ **do**
    $\tau_v^m = \tau_v^m + 1$;
    $V_{\sigma_{\tau_v^m}} = v_{\sigma_{\tau_v^m}}$;
    $v_{\sigma_{\tau_v^m}} = 0$;
    $r = \|v\|_1/(m - \tau_v^m)$;
    $\sigma_{\tau_v^m+1} = \arg\max_i\{|v_i|\}$;
**end**
```

For each j let N_j be a nonnegative random integer with $\mathbf{E}[N_j \mid v] = (m - \tau_v^m)|v_j|/\|v\|_1$. Finally, for $j \in \{1, 2, \ldots, n\} \setminus \{\sigma_1, \sigma_2, \ldots, \sigma_{\tau_v^m}\}$, set

$$V_j = N_j \frac{v_j\|v\|_1}{|v_j|(m - \tau_v^m)}.$$

(Note that v here may fewer nonzero entries than it did upon input.)

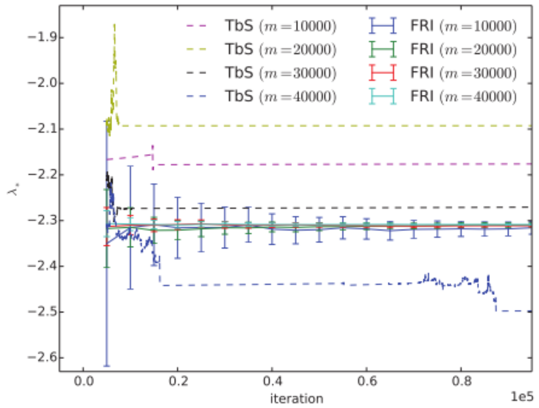

Fig. 5 *Trajectory averages of the approximation, Λ_t^m, of the largest negative eigenvalue of a backwards Kolmogorov operator for a four two-dimensional particle (eight-dimensional) system, with 95% confidence intervals for the FRI method with $m = 1,\ 2,\ 3,$ and 4×10^4. The operator is discretized using a Fourier basis with 101 modes per dimension for a total of more than 10^{16} basis elements (half that after taking advantage of the fact that the desired eigenvector is real). The step-size parameter ε is set to 10^{-3}. Also on this graph are shown trajectories of Λ_t^m for the TbS method for the same values of m.*

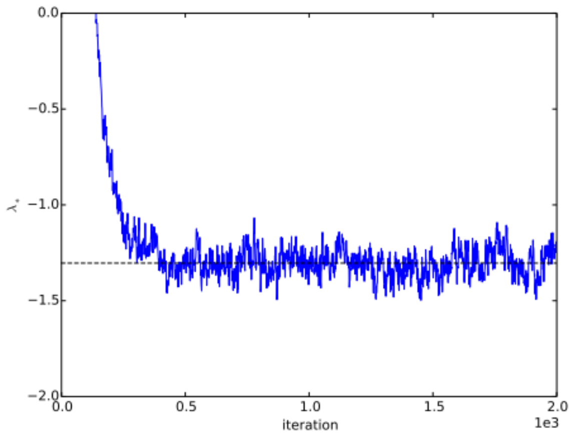

Fig. 6 *Trajectory of the approximation, Λ_t^m (solid line), of the largest negative eigenvalue of a backwards Kolmogorov operator for the five-particle system as computed by the FRI method with $m = 10^6$ over 2×10^3 iterations. The total dimension of the discretized system is more than 10^{20}. The average value of Λ_t^m (ignoring the first 500 iterations) is -1.3 and is shown by a horizontal dotted line.*

References

[1] Lek-Heng Lim and Jonathan Weare. Fast randomized iteration: Diffusion monte carlo through the lens of numerical linear algebra. *SIAM Review*, 59(3):547–587, 2017.

[2] James J. Shepherd, George Booth, Andreas Grüneis, and Ali Alavi. Full configuration interaction perspective on the homogeneous electron gas. *Phys. Rev. B*, 85:081103, Feb 2012.