Semistochastic importance sampling of second-quantized operators in determinant space

Adam Holmes

Valeev Group Meeting
June 7, 2021

Overview

1. Motivation
2. Algorithm
3. Applications

Motivation

Types of quantum chemistry algorithms

1. Operator-based algorithms:

Types of quantum chemistry algorithms

1. Operator-based algorithms:
    1. Coupled cluster

Types of quantum chemistry algorithms

1. Operator-based algorithms:
    1. Coupled cluster
    2. Moller-Plesset perturbation theory

Types of quantum chemistry algorithms

1. Operator-based algorithms:
    1. Coupled cluster
    2. Moller-Plesset perturbation theory

2. Wavefunction-based/determinant-space algorithms:

Types of quantum chemistry algorithms

1. Operator-based algorithms:
   1. Coupled cluster
   2. Moller-Plesset perturbation theory
2. Wavefunction-based/determinant-space algorithms:
   1. Full CI

Types of quantum chemistry algorithms

1. Operator-based algorithms:
    1. Coupled cluster
    2. Moller-Plesset perturbation theory
2. Wavefunction-based/determinant-space algorithms:
    1. Full CI
    2. Selected CI

Types of quantum chemistry algorithms

1. Operator-based algorithms:
    1. Coupled cluster
    2. Moller-Plesset perturbation theory
2. Wavefunction-based/determinant-space algorithms:
    1. Full CI
    2. Selected CI
    3. Epstein-Nesbet perturbation theory

Types of quantum chemistry algorithms

1. Operator-based algorithms:
    1. Coupled cluster
    2. Moller-Plesset perturbation theory

2. Wavefunction-based/determinant-space algorithms:
    1. Full CI
    2. Selected CI
    3. Epstein-Nesbet perturbation theory
    4. Full CI Quantum Monte Carlo (FCIQMC)

In determinant-space algorithms, one of the key steps is applying a second-quantized operator (usually H) to a Slater determinant!

In determinant-space algorithms, one of the key steps is applying a second-quantized operator (usually H) to a Slater determinant!

1. Full CI: Iterative diagonalization of the Hamiltonian (e.g. using Davidson) diagonalizes in the basis of Krylov vectors

$$\{v_i = f(Hv_{i-1})\} \tag{1}$$

In determinant-space algorithms, one of the key steps is applying a second-quantized operator (usually H) to a Slater determinant!

1. Full CI: Iterative diagonalization of the Hamiltonian (e.g. using Davidson) diagonalizes in the basis of Krylov vectors

$$\{v_i = f(Hv_{i-1})\} \tag{1}$$

2. Selected CI: New determinants are 'selected' using a criterion that is a function of $H\psi_0$

In determinant-space algorithms, one of the key steps is applying a second-quantized operator (usually H) to a Slater determinant!

1. Full CI: Iterative diagonalization of the Hamiltonian (e.g. using Davidson) diagonalizes in the basis of Krylov vectors

$$\{v_i = f(Hv_{i-1})\} \tag{1}$$

2. Selected CI: New determinants are 'selected' using a criterion that is a function of $H\psi_0$
3. Epstein-Nesbet perturbation theory:

$$\Delta E = \left\langle \psi_0 \middle| V \frac{1}{E_0 - H_0} V \middle| \psi_0 \right\rangle \tag{2}$$

In determinant-space algorithms, one of the key steps is applying a second-quantized operator (usually H) to a Slater determinant!

1. Full CI: Iterative diagonalization of the Hamiltonian (e.g. using Davidson) diagonalizes in the basis of Krylov vectors

$$\{v_i = f(Hv_{i-1})\} \tag{1}$$

2. Selected CI: New determinants are 'selected' using a criterion that is a function of $H\psi_0$
3. Epstein-Nesbet perturbation theory:

$$\Delta E = \left\langle \psi_0 \middle| V \frac{1}{E_0 - H_0} V \middle| \psi_0 \right\rangle \tag{2}$$

4. FCIQMC: Stochastically simulate the power method,

$$\psi_0 \propto \lim_{n \to \infty} (1 - \tau H)^n \psi_T \tag{3}$$

Approaches for computing $H\left|D_i\right\rangle$

Broadly, there are three main approaches:

1. **Deterministic:** Simply evaluating all single and double excitations from the initial Slater determinant. Complexity: $\mathcal{O}(N^2(M-N)^2)$

Approaches for computing $H\,|D_i\rangle$

Broadly, there are three main approaches:

1. **Deterministic:** Simply evaluating all single and double excitations from the initial Slater determinant. Complexity: $\mathcal{O}(N^2(M-N)^2)$
2. **Stochastic:** Sampling excitations according to some distribution. Reduces the complexity, but introduces a stochastic uncertainty (and in projector methods, the fermion sign problem!)

Approaches for computing $H |D_i\rangle$

Broadly, there are three main approaches:

1. **Deterministic:** Simply evaluating all single and double excitations from the initial Slater determinant. Complexity: $\mathcal{O}(N^2(M-N)^2)$
2. **Stochastic:** Sampling excitations according to some distribution. Reduces the complexity, but introduces a stochastic uncertainty (and in projector methods, the fermion sign problem!)
3. **Semistochastic:** Evaluating the largest-magnitude components, and sampling the remaining, smaller components. Reduces complexity relative to the deterministic approach, but with a **greatly reduced stochastic uncertainty** relative to fully stochastic methods. In projector methods, it also mitigates the bias incurred in taming the fermion sign problem, such as the initiator bias in FCIQMC.

Semistochastic approach

Semistochastic approach

1. Separate out the sum into a **few large terms** and **many small terms** using some threshold ϵ

Semistochastic approach

1. Separate out the sum into a **few large terms** and **many small terms** using some threshold ϵ
2. Compute the sum of the few large terms deterministically (and quickly)

Semistochastic approach

1. Separate out the sum into a **few large terms** and **many small terms** using some threshold ϵ
2. Compute the sum of the few large terms deterministically (and quickly)
3. Importance sample the many small terms

Semistochastic approach

1. Separate out the sum into a **few large terms** and **many small terms** using some threshold ϵ
2. Compute the sum of the few large terms deterministically (and quickly)
3. Importance sample the many small terms

$$H\left|D_i\right\rangle = \sum_j H_{ji} \tag{4}$$

$$= \underbrace{\sum_j H_{ji}\Big|_{|H_{ji}|\geq\epsilon}}_{\substack{\text{few large terms:}\\ \text{sum deterministically}}} + \underbrace{\sum_j H_{ji}\Big|_{|H_{ji}|<\epsilon}}_{\substack{\text{many small terms:}\\ \text{importance sample}}} \tag{5}$$

Algorithm

Deterministic component

- **Setup:**

Deterministic component

- **Setup:**
  - For each pair of orbitals $\{p, q\}$, store a list of $\{r, s, |H(rs \leftarrow pq)|\}$, sorted by $|H(rs \leftarrow pq)|$ in decreasing order

Deterministic component

- **Setup:**
  - For each pair of orbitals $\{p, q\}$, store a list of $\{r, s, |H(rs \leftarrow pq)|\}$, sorted by $|H(rs \leftarrow pq)|$ in decreasing order
  - Also compute (for sampling algorithm):

$$\sum_{rs \in \text{remaining}} |H(rs \leftarrow pq)| \tag{6}$$

Deterministic component

- **Setup:**
  - For each pair of orbitals $\{p, q\}$, store a list of $\{r, s, |H\left(rs \leftarrow pq\right)|\}$, sorted by $|H\left(rs \leftarrow pq\right)|$ in decreasing order
  - Also compute (for sampling algorithm):

$$\sum_{rs \in \text{remaining}} |H(rs \leftarrow pq)| \tag{6}$$

- **Finding important connected determinants** D_j (for which $|H_{ji}| \geq \epsilon$):

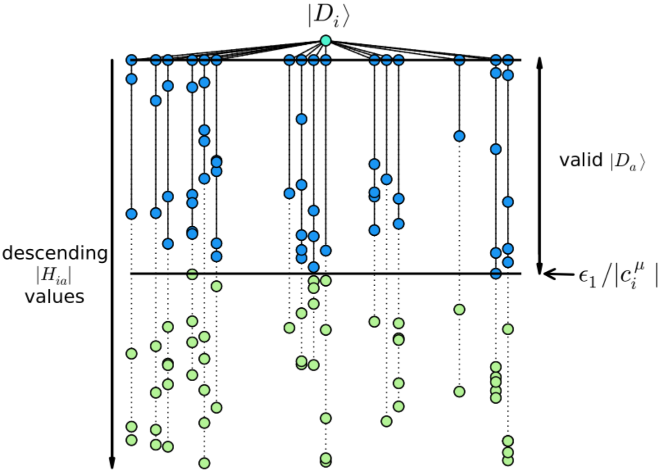
Deterministic component

- **Setup:**
  - For each pair of orbitals $\{p, q\}$, store a list of $\{r, s, |H(rs \leftarrow pq)|\}$, sorted by $|H(rs \leftarrow pq)|$ in decreasing order
  - Also compute (for sampling algorithm):

$$\sum_{rs \in \text{remaining}} |H(rs \leftarrow pq)| \tag{6}$$

- **Finding important connected determinants** D_j (for which $|H_{ji}| \geq \epsilon$):
  - For each pair of occupied orbitals $\{p, q\}$, look up the stored list of $\{r, s, |H(rs \leftarrow pq)|\}$ and iterate until $|H(rs \leftarrow pq)| < \epsilon$

Deterministic component



Discrete sampling algorithms

- **Alias sampling:**

Discrete sampling algorithms

- **Alias sampling:**
  1. Sample an element with uniform probability

Discrete sampling algorithms

- **Alias sampling:**
  1. Sample an element with uniform probability
  2. If a low-probability element was selected, it has some probability of 'aliasing' to another, high-probability element

Discrete sampling algorithms

- **Alias sampling:**
  1. Sample an element with uniform probability
  2. If a low-probability element was selected, it has some probability of 'aliasing' to another, high-probability element

  - Setup time: $\mathcal{O}(N)$, Time per sample: $\mathcal{O}(1)$

Discrete sampling algorithms

- **Alias sampling:**
  1. Sample an element with uniform probability
  2. If a low-probability element was selected, it has some probability of 'aliasing' to another, high-probability element

  - Setup time: $\mathcal{O}(N)$, Time per sample: $\mathcal{O}(1)$
- **Cumulative distribution function (CDF) searching:**

Discrete sampling algorithms

- **Alias sampling:**
  1. Sample an element with uniform probability
  2. If a low-probability element was selected, it has some probability of 'aliasing' to another, high-probability element
  - Setup time: $\mathcal{O}(N)$, Time per sample: $\mathcal{O}(1)$
- **Cumulative distribution function (CDF) searching:**
  1. Sample a real number r uniformly between 0 and 1

Discrete sampling algorithms

- **Alias sampling:**
  1. Sample an element with uniform probability
  2. If a low-probability element was selected, it has some probability of 'aliasing' to another, high-probability element

  - Setup time: $\mathcal{O}(N)$, Time per sample: $\mathcal{O}(1)$
- **Cumulative distribution function (CDF) searching:**
  1. Sample a real number r uniformly between 0 and 1
  2. Binary-search the CDF for r (i.e., find the smallest index i for which $r < \mathrm{CDF}_i$)

Discrete sampling algorithms

- **Alias sampling:**
  1. Sample an element with uniform probability
  2. If a low-probability element was selected, it has some probability of 'aliasing' to another, high-probability element

  - Setup time: $\mathcal{O}(N)$, Time per sample: $\mathcal{O}(1)$
- **Cumulative distribution function (CDF) searching:**
  1. Sample a real number r uniformly between 0 and 1
  2. Binary-search the CDF for r (i.e., find the smallest index i for which $r < \mathrm{CDF}_i$)

  - Setup time: $\mathcal{O}(N)$, Time per sample: $\mathcal{O}(\log N)$

Discrete sampling algorithms

- **Alias sampling:**
  1. Sample an element with uniform probability
  2. If a low-probability element was selected, it has some probability of 'aliasing' to another, high-probability element

  - Setup time: $\mathcal{O}(N)$, Time per sample: $\mathcal{O}(1)$
- **Cumulative distribution function (CDF) searching:**
  1. Sample a real number r uniformly between 0 and 1
  2. Binary-search the CDF for r (i.e., find the smallest index i for which $r < \mathrm{CDF}_i$)

  - Setup time: $\mathcal{O}(N)$, Time per sample: $\mathcal{O}(\log N)$
- Usually, Alias sampling is preferred because it has lower sampling complexity, but CDF searching has a unique use case that we will need...

CDF searching's unique use case

Suppose we have a stored CDF, and we want to be able to efficiently sample one of the first n elements, where n is not known ahead of time. CDF searching can do this efficiently with a small modification:

1. Sample a real number r uniformly between 0 and CDF_n (Note that $\mathrm{CDF}_N = 1$ reduces to the original algorithm)

CDF searching's unique use case

Suppose we have a stored CDF, and we want to be able to efficiently sample one of the first n elements, where n is not known ahead of time. CDF searching can do this efficiently with a small modification:

1. Sample a real number r uniformly between 0 and CDF_n (Note that $\mathrm{CDF}_N = 1$ reduces to the original algorithm)
2. Binary-search the CDF for r (i.e., find the smallest index i for which $r < \mathrm{CDF}_i$)

Importance-sampling excitations

- Recall that we have already described how to treat the large-magnitude ($|H_{ji}| \geq \epsilon$) excitations deterministically

Importance-sampling excitations

- Recall that we have already described how to treat the large-magnitude ($|H_{ji}| \geq \epsilon$) excitations deterministically
- Now, we want to "importance-sample" the remaining small-magnitude ($|H_{ji}| < \epsilon$) excitations, i.e.:

$$P(H_{ji}) \begin{cases} \propto f(|H_{ji}|), & \text{if } |H_{ji}| < \epsilon; \\ = 0, & \text{if } |H_{ji}| \geq \epsilon. \end{cases} \tag{7}$$

Importance-sampling excitations

- Recall that we have already described how to treat the large-magnitude ($|H_{ji}| \geq \epsilon$) excitations deterministically
- Now, we want to "importance-sample" the remaining small-magnitude ($|H_{ji}| < \epsilon$) excitations, i.e.:

$$P(H_{ji}) \begin{cases} \propto f(|H_{ji}|), & \text{if } |H_{ji}| < \epsilon; \\ = 0, & \text{if } |H_{ji}| \geq \epsilon. \end{cases} \tag{7}$$

- Examples of importance-sampling distributions include:

$$f(|H_{ji}|) = |H_{ji}|, \qquad f(|H_{ji}|) = |H_{ji}|^2 \tag{8}$$

Importance-sampling excitations

We divide the sampling of an excitation into two stages:

1. Sample an exciting electron pair

Importance-sampling excitations

We divide the sampling of an excitation into two stages:

1. Sample an exciting electron pair
    - with probability proportional to the sum of remaining (small-magnitude) targets it could excite to

Importance-sampling excitations

We divide the sampling of an excitation into two stages:

1. Sample an exciting electron pair
   - with probability proportional to the sum of remaining (small-magnitude) targets it could excite to
2. Sample a target orbital pair

Importance-sampling excitations

We divide the sampling of an excitation into two stages:

1. Sample an exciting electron pair
   - with probability proportional to the sum of remaining (small-magnitude) targets it could excite to
2. Sample a target orbital pair
   - with probability proportional to its excitation magnitude

Sampling an exciting electron pair

- During the deterministic step, for each exciting electron pair, we iterate over the sorted excitation magnitudes

Sampling an exciting electron pair

- During the deterministic step, for each exciting electron pair, we iterate over the sorted excitation magnitudes
- As soon as the threshold is reached, we record the sum of remaining excitation magnitudes as that electron pair's relative probability

Sampling an exciting electron pair

- During the deterministic step, for each exciting electron pair, we iterate over the sorted excitation magnitudes
- As soon as the threshold is reached, we record the sum of remaining excitation magnitudes as that electron pair's relative probability
- After doing this for all electron pairs, we set up the Alias sampling data structure in $\mathcal{O}(N^2)$ time

Sampling an exciting electron pair

- During the deterministic step, for each exciting electron pair, we iterate over the sorted excitation magnitudes
- As soon as the threshold is reached, we record the sum of remaining excitation magnitudes as that electron pair's relative probability
- After doing this for all electron pairs, we set up the Alias sampling data structure in $\mathcal{O}(N^2)$ time
- This enables **importance-sampling an exciting electron pair in $\mathcal{O}(1)$ time**

Sampling a target orbital pair

- Once an exciting electron pair is selected, we now have to sample a target orbital pair to excite to

Sampling a target orbital pair

- Once an exciting electron pair is selected, we now have to sample a target orbital pair to excite to
- Since the deterministic stage has already been completed, we know which target orbital pair (r, s) is the first one that hasn't been treated deterministically

Sampling a target orbital pair

- Once an exciting electron pair is selected, we now have to sample a target orbital pair to excite to
- Since the deterministic stage has already been completed, we know which target orbital pair (r, s) is the first one that hasn't been treated deterministically
- We have a CDF (in reverse order) corresponding to all target orbital pairs, and we need to sample one of the orbital pairs whose index is at least as high as this one

Sampling a target orbital pair

- Once an exciting electron pair is selected, we now have to sample a target orbital pair to excite to
- Since the deterministic stage has already been completed, we know which target orbital pair (r, s) is the first one that hasn't been treated deterministically
- We have a CDF (in reverse order) corresponding to all target orbital pairs, and we need to sample one of the orbital pairs whose index is at least as high as this one
- So, we just use CDF searching to **sample a target orbital pair in $\mathcal{O}(\log M)$ time**

What about single excitations?

- Unlike double excitations, single excitation matrix element magnitudes depend on the orbital occupancies of the exciting determinant

What about single excitations?

- Unlike double excitations, single excitation matrix element magnitudes depend on the orbital occupancies of the exciting determinant
- But we can compute upper bounds to the single excitation magnitudes:

$$|H(p \to r)| \leq |H(p \to r)|_{\max}, \tag{9}$$

where $|H(p \to r)|_{\max}$ is the largest magnitude the $p \to r$ excitation can have from *any* initial determinant

What about single excitations?

- Unlike double excitations, single excitation matrix element magnitudes depend on the orbital occupancies of the exciting determinant
- But we can compute upper bounds to the single excitation magnitudes:

$$|H(p \to r)| \leq |H(p \to r)|_{\max}\,, \tag{9}$$

  where $|H(p \to r)|_{\max}$ is the largest magnitude the $p \to r$ excitation can have from *any* initial determinant
- The simplest way to incorporate singles is to sample them with probability $P(|H(p \to r)|) \propto f(|H(p \to r)|_{\max})$ and skip them in the deterministic step

Sampling 'invalid' excitations

- Sometimes this algorithm will sample 'invalid' excitations to already-occupied orbitals

Sampling ‘invalid’ excitations

- Sometimes this algorithm will sample ‘invalid’ excitations to already-occupied orbitals
- While unbiased, this increases the variance in the samples

Sampling ‘invalid’ excitations

- Sometimes this algorithm will sample ‘invalid’ excitations to already-occupied orbitals
- While unbiased, this increases the variance in the samples
- We can fix this as follows:

Sampling 'invalid' excitations

- Sometimes this algorithm will sample 'invalid' excitations to already-occupied orbitals
- While unbiased, this increases the variance in the samples
- We can fix this as follows:
  1. If an 'invalid' excitation is sampled, discard it and resample until a valid excitation is sampled

Sampling ‘invalid’ excitations

- Sometimes this algorithm will sample ‘invalid’ excitations to already-occupied orbitals
- While unbiased, this increases the variance in the samples
- We can fix this as follows:
  1. If an ‘invalid’ excitation is sampled, discard it and resample until a valid excitation is sampled
  2. Scale the sample probabilities by a constant factor so they remain normalized (can be performed in $\mathcal{O}(N^4 + N_{\det} N^3)$ time during deterministic step)

Applications

Epstein-Nesbet perturbation theory

Given a variational wavefunction $\psi_0 = \sum_i c_i \left|D_i\right\rangle$ with energy E_0, the Epstein-Nesbet perturbative correction to the energy is give by

$$\Delta E\left[V\left|\psi_0\right\rangle\right] = \left\langle \psi_0 \left| V \frac{1}{E_0 - H_0} V \right| \psi_0 \right\rangle \tag{10}$$

$$= \sum_{\substack{a \in \mathcal{C}(\mathcal{V}) \\ \notin \mathcal{V}}} \frac{1}{E_0 - E_a} \left(\sum_{ij \in \mathcal{V}} c_j H_{ja} H_{ai} c_i \right), \tag{11}$$

so we can now evaluate the H_{ai} components semistochastically using importance sampling.

Semistochastic Epstein-Nesbet Perturbation Theory

1. Divide $V\left|\psi_0\right\rangle$ into two pieces:

$$V\left|\psi_0\right\rangle = \left(V\left|\psi_0\right\rangle\right)_{\geq\epsilon} + \left(V\left|\psi_0\right\rangle\right)_{<\epsilon}. \tag{12}$$

Semistochastic Epstein-Nesbet Perturbation Theory

1. Divide $V\left|\psi_0\right\rangle$ into two pieces:

$$V\left|\psi_0\right\rangle = \left(V\left|\psi_0\right\rangle\right)_{\geq\epsilon} + \left(V\left|\psi_0\right\rangle\right)_{<\epsilon}. \tag{12}$$

2. Estimate ΔE deterministically:

$$\Delta E\left[V\left|\psi_0\right\rangle\right] \approx \Delta E\left[\left(V\left|\psi_0\right\rangle\right)_{\geq\epsilon}\right] \tag{13}$$

Semistochastic Epstein-Nesbet Perturbation Theory

1. Divide $V|\psi_0\rangle$ into two pieces:

$$V|\psi_0\rangle = \left(V|\psi_0\rangle\right)_{\geq\epsilon} + \left(V|\psi_0\rangle\right)_{<\epsilon}. \tag{12}$$

2. Estimate ΔE deterministically:

$$\Delta E\left[V|\psi_0\rangle\right] \approx \Delta E\left[\left(V|\psi_0\rangle\right)_{\geq\epsilon}\right] \tag{13}$$

3. Stochastically evaluate the difference between the exact and approximate ΔE

Sampling algorithm in Epstein-Nesbet PT

How to evaluate the stochastic component?

- Old approach (SHCI):

Sampling algorithm in Epstein-Nesbet PT

How to evaluate the stochastic component?

- Old approach (SHCI):
  - Sample a batch of determinants $\{D_i\}$ from ψ_0, compute $\sum_a H_{ai} c_i$ deterministically for each sampled D_i

Sampling algorithm in Epstein-Nesbet PT

How to evaluate the stochastic component?

- Old approach (SHCI):
  - Sample a batch of determinants $\{D_i\}$ from ψ_0, compute $\sum_a H_{ai} c_i$ deterministically for each sampled D_i
  - Drawbacks:

Sampling algorithm in Epstein-Nesbet PT

How to evaluate the stochastic component?

- Old approach (SHCI):
  - Sample a batch of determinants $\{D_i\}$ from ψ_0, compute $\sum_a H_{ai} c_i$ deterministically for each sampled D_i
  - Drawbacks:
    - No importance sampling of H matrix elements

Sampling algorithm in Epstein-Nesbet PT

How to evaluate the stochastic component?

- Old approach (SHCI):
  - Sample a batch of determinants $\{D_i\}$ from ψ_0, compute $\sum_a H_{ai}c_i$ deterministically for each sampled D_i
  - Drawbacks:
    - No importance sampling of H matrix elements
    - Memory-limited because requires computing all determinants $\{D_a\}$ for which $H_{ai} \neq 0$

Sampling algorithm in Epstein-Nesbet PT

How to evaluate the stochastic component?

- Old approach (SHCI):
  - Sample a batch of determinants $\{D_i\}$ from ψ_0, compute $\sum_a H_{ai}c_i$ deterministically for each sampled D_i
  - Drawbacks:
    - No importance sampling of H matrix elements
    - Memory-limited because requires computing all determinants $\{D_a\}$ for which $H_{ai} \neq 0$
    - Wastes time on small-magnitude H_{ai} elements

Sampling algorithm in Epstein-Nesbet PT

- New approach (Importance sampling):

Sampling algorithm in Epstein-Nesbet PT

- New approach (Importance sampling):
  - Sample pairs of determinants $\{D_i, D_a\}$ using importance sampling:

$$P(D_i, D_a) \propto (H_{ai}c_i)^2 \tag{14}$$

Sampling algorithm in Epstein-Nesbet PT

- New approach (Importance sampling):
  - Sample pairs of determinants $\{D_i, D_a\}$ using importance sampling:

$$P(D_i, D_a) \propto (H_{ai}c_i)^2 \tag{14}$$

  - Use those samples to evaluate the stochastic component of ΔE, whose largest-magnitude component is

$$\sum_{ai} \frac{(H_{ai}c_i)^2}{E_0 - E_a} \tag{15}$$

Sampling algorithm in Epstein-Nesbet PT

- New approach (Importance sampling):
  - Sample pairs of determinants $\{D_i, D_a\}$ using importance sampling:

$$P(D_i, D_a) \propto (H_{ai} c_i)^2 \tag{14}$$

  - Use those samples to evaluate the stochastic component of ΔE, whose largest-magnitude component is

$$\sum_{ai} \frac{(H_{ai} c_i)^2}{E_0 - E_a} \tag{15}$$

  - Advantages:

Sampling algorithm in Epstein-Nesbet PT

- New approach (Importance sampling):
  - Sample pairs of determinants $\{D_i, D_a\}$ using importance sampling:
$$P(D_i, D_a) \propto (H_{ai}c_i)^2 \tag{14}$$
  - Use those samples to evaluate the stochastic component of ΔE, whose largest-magnitude component is
$$\sum_{ai} \frac{(H_{ai}c_i)^2}{E_0 - E_a} \tag{15}$$
  - Advantages:
    - Uses importance sampling to greatly reduce the variance!

Sampling algorithm in Epstein-Nesbet PT

- New approach (Importance sampling):
  - Sample pairs of determinants $\{D_i, D_a\}$ using importance sampling:

$$P(D_i, D_a) \propto (H_{ai}c_i)^2 \tag{14}$$

  - Use those samples to evaluate the stochastic component of ΔE, whose largest-magnitude component is

$$\sum_{ai} \frac{(H_{ai}c_i)^2}{E_0 - E_a} \tag{15}$$

  - Advantages:
    - Uses importance sampling to greatly reduce the variance!
    - No time or memory is wasted on deterministically evaluating contributions from small-magnitude H_{ai} elements!

FCIQMC Overview

- Power method to project out the Full CI ground state:

$$\psi_0 \propto \lim_{n\to\infty} (1 - \tau H)^n \psi_T \tag{16}$$

FCIQMC Overview

- Power method to project out the Full CI ground state:

$$\psi_0 \propto \lim_{n\to\infty} (1 - \tau H)^n \psi_T \tag{16}$$

- When the Hilbert space is too large, we can simulate each iteration of the power method

$$\psi^{(t)} = (1 - \tau H)\psi^{(t-1)} \tag{17}$$

by applying H stochastically (or semistochastically)

FCIQMC Overview

- Power method to project out the Full CI ground state:

$$\psi_0 \propto \lim_{n\to\infty} (1 - \tau H)^n \psi_T \tag{16}$$

- When the Hilbert space is too large, we can simulate each iteration of the power method

$$\psi^{(t)} = (1 - \tau H)\psi^{(t-1)} \tag{17}$$

by applying H stochastically (or semistochastically)

- The energy can be estimated using a mixed estimator:

$$E_0 = \frac{\langle \psi_T \left| H \right| \psi_0 \rangle}{\langle \psi_T | \psi_0 \rangle}, \tag{18}$$

using a precomputed ψ_T (e.g. from Selected CI)

Semistochastic algorithm in FCIQMC

How to perform the semistochastic projection $\psi^{(t)} = (1 - \tau H)\psi^{(t-1)}$?

- Old approach (Semistochastic FCIQMC):

Semistochastic algorithm in FCIQMC

How to perform the semistochastic projection $\psi^{(t)} = (1 - \tau H)\psi^{(t-1)}$?

- Old approach (Semistochastic FCIQMC):
  1. Choose a fixed set of 'important' determinants $\mathcal{D}$ (e.g., using Selected CI) before the run

Semistochastic algorithm in FCIQMC

How to perform the semistochastic projection $\psi^{(t)} = (1 - \tau H)\psi^{(t-1)}$?

- Old approach (Semistochastic FCIQMC):
  1. Choose a fixed set of 'important' determinants $\mathcal{D}$ (e.g., using Selected CI) before the run
  2. Deterministically apply the diagonal Hamiltonian matrix elements, and the off-diagonal elements between pairs of important determinants

Semistochastic algorithm in FCIQMC

How to perform the semistochastic projection $\psi^{(t)} = (1 - \tau H)\psi^{(t-1)}$?

- Old approach (Semistochastic FCIQMC):
    1. Choose a fixed set of ‘important’ determinants $\mathcal{D}$ (e.g., using Selected CI) before the run
    2. Deterministically apply the diagonal Hamiltonian matrix elements, and the off-diagonal elements between pairs of important determinants
    3. Importance-sample off-diagonal elements, discard the ones that were already treated deterministically

Semistochastic algorithm in FCIQMC

- New approach (Dynamic, importance-sampled semistochastic FCIQMC):

Semistochastic algorithm in FCIQMC

- New approach (Dynamic, importance-sampled semistochastic FCIQMC):

  1. Apply the Hamiltonian using semistochastic importance sampling to each determinant in $\psi^{(t)}$

Semistochastic algorithm in FCIQMC

- New approach (Dynamic, importance-sampled semistochastic FCIQMC):

    1. Apply the Hamiltonian using semistochastic importance sampling to each determinant in $\psi^{(t)}$

- Advantages:

Semistochastic algorithm in FCIQMC

- New approach (Dynamic, importance-sampled semistochastic FCIQMC):

  1. Apply the Hamiltonian using semistochastic importance sampling to each determinant in $\psi^{(t)}$

- Advantages:

  - Deterministic/stochastic division is dynamic: it depends on the current $\psi^{(t)}$, which should greatly reduce the stochastic fluctuations and mitigate the fermion sign problem

Semistochastic algorithm in FCIQMC

- New approach (Dynamic, importance-sampled semistochastic FCIQMC):

  1. Apply the Hamiltonian using semistochastic importance sampling to each determinant in $\psi^{(t)}$

- Advantages:

  - Deterministic/stochastic division is dynamic: it depends on the current $\psi^{(t)}$, which should greatly reduce the stochastic fluctuations and mitigate the fermion sign problem

  - Directly samples the off-diagonal H elements that were not treated deterministically, which should greatly reduce the computational time