10

Power methods

Quantum Monte Carlo versions of the power method for finding ground states come with different names, including the projector method, diffusion Monte Carlo, and Green's function Monte Carlo. These quantum Monte Carlo methods are primarily adoptions of methods developed for classical problems. We summarize the basics of deterministic power methods, detail key features of Monte Carlo power methods, and put these concepts into the context of quantum ground state calculations. We conclude the chapter by outlining power methods that allow the computation of a few excited states. In the computation of excited states, we encounter sign problems, which are discussed in more detail in Chapter 11. The methods and techniques of the present chapter are very general and most usefully applied to systems of Bosons and to systems of Fermions and quantum spins that do not suffer from a sign problem.

10.1 Deterministic direct and inverse power methods

The power method is over a century old. As a deterministic method, it originally was combined with the deflation technique (Meyer, 2000; Stewart, 2001b) as a method to compute all the eigenvalues and eigenvectors of small matrices. Today, its deterministic use is limited primarily to special applications involving very large, sparse matrices. Many of these applications are in the study of quantum lattice models as the corresponding Hamiltonian matrices are typically very sparse. In these applications, we can perhaps initially store in computer memory the relatively small number of nonzero matrix elements and all the vectors. Soon, because of what is undoubtedly the now familiar exponentially increasing number of basis states that determines the order of the Hamiltonian matrix and hence the size of our vectors, we reach the point where computer memory restrictions allow us to store only a

few vectors and we need to compute the matrix elements on the fly. When we have problems for which we cannot even store all the components of one vector, the power method's expression as a Monte Carlo procedure is our only option. With it, we can compute a few of the extremal eigenvalues, that is, a few of the largest or smallest eigenvalues.

The mathematical basis for the method is simple. We already briefly discussed it in Section 2.4.2. If $\{(\lambda_i, |i\rangle), i = 1, 2, \ldots, N\}$ is the set of eigenpairs of A, the eigenvalues are ordered as $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_N|$, and the state $|\phi\rangle$ is of the form $|\phi\rangle = \sum_i w_i^0 |i\rangle$, then for an operator or matrix A

$$A^n|\phi\rangle = \lambda_1^n \left[w_1^0|1\rangle + \sum_{i=2}^{N} \left(\frac{\lambda_i}{\lambda_1}\right)^n w_i^0|i\rangle \right]. \tag{10.1}$$

By making n large enough, we project the dominant state $|1\rangle$ out of $|\phi\rangle$ because $(\lambda_{i>1}/\lambda_1)^n \to 0$ as $n \to \infty$. Given an initial state $|\phi\rangle$ and an operator A, the power method (Meyer, 2000; Stewart, 2001b) reaches the limit of large n by iterating the following steps:

$$\begin{aligned} |\psi\rangle &= A|\phi\rangle, \\ |\phi\rangle &= |\psi\rangle / \|\psi\|. \end{aligned} \tag{10.2}$$

On convergence, the dominant eigenpair $(\lambda_1, |1\rangle)$ of A is $(\|\psi\|, |\phi\rangle)$. Any choice of norm works. For example, we commonly use $\|\phi\|_\infty = \sum_i |w_i^0|$. The method assumes that the initial state overlaps with the dominant state, that is, $w_1^0 \neq 0$. The rate of convergence to the dominant state is controlled by $|\lambda_2|/|\lambda_1|$.

For eventual points of comparison with our Monte Carlo power methods, we state a more detailed deterministic method in Algorithm 33. It is common to shift

Algorithm 33 Shifted power method.

Input: matrix A, shift σ, convergence criterion ϵ, starting state $|\phi\rangle$.

```
norm = ||A|| ;
|φ⟩ ← |φ⟩/ ||φ|| ;
repeat
   |ψ⟩ = A|φ⟩ ;
   λ = ⟨φ|ψ⟩ ;
   |r⟩ = |ψ⟩ − λ|φ⟩ ;
   |φ⟩ ← |ψ⟩ − σ|φ⟩ ;
   |φ⟩ ← |φ⟩/ ||φ|| ;
until ||r|| ≤ (norm)ε.
return the eigenpair (λ, |φ⟩).
```

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

the origin of the eigenvalue spectrum and iterate with $A - \sigma I$ instead of A. Convergence is now to the eigenvalue farthest from σ, which is either the largest or smallest one. Choosing σ becomes equivalent to making a good guess for which of these two eigenvalues is sought. With the shift, convergence scales as $|\lambda_2 - \sigma|/|\lambda_1 - \sigma|$ or $|\lambda_{N-1} - \sigma|/|\lambda_N - \sigma|$. The closer the guess is to the exact answer, the more rapid is the convergence.

As stated, Algorithm 33 partially disguises the Rayleigh quotient (9.4) by adjusting the estimate of the dominant eigenvalue at each step and at each step reducing the norm of the residual state $|r\rangle = |\psi\rangle - \langle\psi|\phi\rangle|\phi\rangle$. The Rayleigh quotient gives a more precise eigenvalue estimate than the one based on the norm. In the variational Monte Carlo method (Chapter 9.1), we used this same quotient and residual to provide an upper bound of the eigenvalue. Here, we are driving the iteration not to a bound but to the exact answer.

A second power method is the *inverse power method* (Meyer, 2000; Stewart, 2001b). In this method, with some initial state $|\phi\rangle$ and a shifted matrix $A - \sigma I$, we iterate

$$\begin{aligned} |\psi\rangle &= (A - \sigma I)^{-1}|\phi\rangle, \\ |\phi\rangle &= |\psi\rangle/\|\psi\|. \end{aligned} \tag{10.3}$$

As for the direct power method, the iteration converges to a dominant eigenpair, but this time it is to the dominant pair of $(A - \sigma I)^{-1}$, that is, the eigenpair of $A - \sigma I$ whose eigenvalue is closest to zero.

Since knowing the inverse of $A - \sigma I$ is akin to knowing the solution to the problem, we instead iterate the following steps:

$$\begin{aligned} &\text{Solve } (A - \sigma I)\,|\psi\rangle = |\phi\rangle, \\ &|\phi\rangle = |\psi\rangle/\|\psi\|. \end{aligned} \tag{10.4}$$

This seems just as problematic because the first step of each iteration requires solving a linear system of equations. However, solving a linear system is more efficient and stable than inverting a matrix.

Algorithm 34 gives a detailed version of the inverse power method. It uses the same initial good estimate of the ground state energy in each iterative step. We typically solve the linear systems of equations with the use of Gaussian elimination with partial pivoting (Stewart, 2001a). The first step of this method, which also consumes the most computer time, is an LU factorization (Stewart, 2001a) of the matrix $A - \sigma I$. By fixing our guess σ, we need only factor the matrix once, and we can do this outside of the iterative loop. Then, in the loop, we solve the linear system by the less computationally intensive forward and back substitution methods (Stewart, 2001a).

Algorithm 34 Shifted inverse power method.

Input: matrix A, shift σ, convergence criterion ϵ, nonzero starting state $|\phi\rangle$.
  norm $= \|A\|$;
  Perform an LU factorization of $A - \sigma I$;
  repeat
    Solve $(A - \sigma I)|\psi\rangle = |\phi\rangle$;
    $|\varphi\rangle = |\psi\rangle / \|\psi\|$;
    $|\chi\rangle = |\phi\rangle / \|\psi\|$;
    $\mu = \langle\varphi|\chi\rangle$;
    $\lambda = \sigma + \mu$;
    $|r\rangle = |\chi\rangle - \mu|\varphi\rangle$;
    $|\phi\rangle = |\varphi\rangle$;
  until $\|r\| \leq (\text{norm})\epsilon$.
  return the eigenpair $(\lambda, |\phi\rangle)$.

10.2 Monte Carlo power methods

The just concluded discussion of the power method described the standard deterministic implementation. What we represented as $|i\rangle$ could have been some classical unit vector or some quantum basis state. Similarly, A could have been a matrix or an operator.

The core computation in a deterministic power method is a conventional matrix-vector multiplication for dense or sparse matrices and dense vectors. A Monte Carlo power method is most useful for cases where we have a means to generate the hopefully sparse matrix elements on the fly. In these implementations, what we move from one iteration step to another is not all the vector components but a subset of them that we call walkers. A *walker* is a tuple of attributes that is at least (*weight*, *state*) where "state" is a label specifying one of the basis vectors. It is useful to view the subset of components as a stack of walkers. In other words, we let the stack represent a vector in such a way that the representation is asymptotically exact in the limit of many walkers. During the simulation, we pop walkers off one stack and push them onto a new one or back onto the old one. It becomes convenient to interpret the symbol w_i as more than just being the i-th component of a vector but also as the weight of walker i, which just happens to be in a state labeled by i. Using a stack data structure is convenient for the exposition but not necessary for the implementation.

The computational and memory bottleneck in a deterministic power method is the multiplication of a vector by a matrix. In a Monte Carlo power method, we replace the exact (deterministic) multiplications with a Monte Carlo estimate. Each

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

Monte Carlo step (one matrix-vector multiplication) generates a sample of the components of the resultant vector. From this sample the iteration then produces another sample in such a way that averaging the iteration over many steps and a sufficient number of walkers estimates the complete multiplication.

Replacing these multiplications by a Monte Carlo procedure requires replacing the estimation of the eigenvalue by estimators that differ from the ones we used deterministically and adding extra procedures to make the sampling efficient. In this section, we describe only ways to do the multiplications and to solve the linear system by Monte Carlo sampling. In the following section (Section 10.3), we address efficiency issues. When this discussion is complete, we jump to the basic types of quantum Monte Carlo power methods that employ the matrix-vector multiplication sampling (Section 10.4). We then complete the discussion of quantum power methods in the sections after this one by discussing estimators of the ground state energy, correlation functions, and exited states.

10.2.1 Monte Carlo direct power method

At this point, we recall our discussion of Markov chain Monte Carlo in Section 2.4.2. Sampling via a Markov chain is a Monte Carlo power method. In Markov chain Monte Carlo, we project to a stationary distribution that is the right eigenvector of the stochastic matrix defining the chain. Because the matrix is stochastic, we know that its dominant eigenvalue is 1. In a detailed balanced algorithm, where we specify a priori the stationary distribution, we are designing a stochastic matrix such that this distribution is its right-hand eigenvector. We are now concerned with problems where we cannot use detailed balance because we want to estimate an a priori unknown eigenvalue and concomitantly averages of other quantities with respect to an unknown stationary distribution, that is, from an unknown right eigenvector of our matrix. We still have to construct a transition probability matrix.

Constructing a transition probability matrix is straightforward if A has no negative matrix elements. For notational simplicity, we now absorb the shift σI into the definition of A. Matrix representations of A, even if all the elements A_{ij} of A are nonnegative, in general, do not lead to a stochastic matrix. We can, however, always define the elements P_{ij} of a stochastic matrix P related to the A_{ij} elements of A via[1]

$$A_{ij} = w_{ij} P_{ij},$$

where $\sum_i P_{ij} = 1$, $P_{ij} = 0$ only if $A_{ij} \geq 0$, and $w_{ij} > 0$.

[1] More generally, let us suppose we have a matrix A that is nonnegative and irreducible and the relation $Aa = \alpha a$, where the components of the vector αa are positive. Then, if D is a diagonal matrix such that $De = a$, where e is a vector with unit elements, $D^{-1}AD/\alpha$ is a stochastic matrix (Householder, 2006).

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

How do we choose P_{ij} and how do we sample from it? Common and effective choices[2] for P_{ij} and w_{ij} are[3]

$$P_{ij} = \frac{A_{ij}}{\sum_i A_{ij}}, \quad w_{ij} = \sum_{i'} A_{i'j} \text{ (independent of } i\text{)}.$$

Often, w_{ij} is called the weight transfer multiplier. An obvious assumption is that from a given state $|j\rangle$, we can determine (or look up) all the possible values of P_{ij}. To sample P_{ij}, which in a direct power method we regard as the transition probability from state $|j\rangle$ to state $|i\rangle$, we use the sampling concepts from Section 2.3. For example, if from a given state $|j\rangle$, the number of possible states $|i\rangle$ is relatively small, we calculate the cumulative probabilities and use Algorithm 1 or 2 to select the new state $|i\rangle$. If the number of available states from any given state is large, other techniques such as the sewing algorithm (Booth and Gubernatis, 2009a) are potentially useful. Still another approach is expressing A as a product of sparse matrices, say, $A = B_L B_{L-1} \cdots B_1$, then sampling the transition matrix in L stages.

To estimate the matrix-vector multiplications

$$|\psi\rangle = A^n |\phi\rangle$$

by Monte Carlo sampling, we suppose the initial state is $|\phi\rangle = \sum_i w_i^0 |i\rangle$, where $w_i^0 > 0$ and $\sum_i w_i^0 = 1$. Then, after n multiplications,

$$|\psi\rangle = \sum_{i_1,\ldots,i_n} A|i_n\rangle\langle i_n|A|i_{n-1}\rangle \cdots \langle i_2|A|i_1\rangle w_{i_1}^0 \tag{10.5}$$

so the i-th component of $|\Psi\rangle$, $\langle i|\psi\rangle = \sum_i \langle i|\psi\rangle |i\rangle = \sum_i w_i |i\rangle$, is simply

$$w_i = \sum_{i_1,i_1,\ldots,i_n} A_{ii_n} A_{i_n i_{n-1}} \cdots A_{i_2 i_1} w_{i_1}^0 > 0. \tag{10.6}$$

This last equation we can rewrite as

$$w_i = \sum_{i_1,i_2,\ldots,i_n} w_{ii_n} w_{i_n i_{n-1}} \cdots w_{i_2 i_1} w_{i_1}^0 P_{ii_n} P_{i_n i_{n-1}} \cdots P_{i_2 i_1}, \tag{10.7}$$

allowing us to interpret w_i as being the expectation value of

$$w_{ii_n} w_{i_n i_{n-1}} \cdots w_{i_2 i_1} w_{i_1}^0 \tag{10.8}$$

with respect to the joint probability

$$P(i, i_n, i_{n-1}, \ldots, i_1) = P_{ii_n} P_{i_n i_{n-1}} \cdots P_{i_2 i_1}. \tag{10.9}$$

[2] We are not considering a case where P_{ij} depends on the iteration step or the current weight of the walker.

[3] With these choices w_{ij} is independent of i. For notational convenience it is customary and useful to retain this subscript

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

This joint distribution naturally enjoys a Markovian factorization. The positivity of the w_i leads one to refer to them not just as components of $|\psi\rangle$ in some basis $\{|i\rangle\}$ but also as weights.

If we were to consider the i_1-th component of the initial state as a random walker, then we could interpret the indices $\{i, i_n, i_{n-1}, \ldots, i_1\}$ of each term in the sum (10.7) as defining a weighted path γ of length n taken by this walker through probability space and interpret the summation as a summing over all possible paths, which we represent as

$$w_i = \langle w_i(\gamma) \rangle_\gamma,$$

where $w_i(\gamma)$ is (10.8), the product accumulation of weight transfer along a particular path γ.[4] The Monte Carlo task is to sample the important paths for the important walkers.

We sample the paths in steps with an ensemble of walkers. At the first step, we sample M random walkers from the distribution defined by w^0, possibly leading to a population of walkers where a state is represented more than once. Next, we process each walker in this sample, one at a time. If the walker is in state $|j\rangle$, whose current weight is w_j, then we sample P_{ij} for a new state $|i\rangle$ and transition the walker to state $|i\rangle$ with the weight $w_i = w_{ij} w_j$ (pushing the walker onto a new stack). After we process the entire population of walkers (all that were in our original stack), we have completed one step, and then we repeat this procedure on the new walker population (the new stack). By design, the number of walkers is $M \ll N$ where N is the dimension of the basis.

The two panels on the left side of Fig. 10.1 schematically represent the difference between a matrix-vector multiplication step in a deterministic power method and one that is performed by sampling. In the figure, a column of circles represents all possible states in the system. When the matrix-vector multiplication is performed exactly, the old states may contribute to multiple new ones, and a new one may receive contributions from multiple old ones. The details of the matrix determine the connections between old and new. When the matrix-vector multiplication step is performed by Monte Carlo, a sample of old states contributes to a new sample of states usually on a one-to-one basis. From the way they were defined, the nonzero elements of P and A have the same ij index pairs, but in a given step, we do not use all of them. The population sizes are orders of magnitude smaller than the number of available states. Because these sizes are small, two or more old states contributing to the same new one state is a rare event.

Algorithm 35 details one way to perform a sweep (one matrix-vector multiplication sampling step) in a Monte Carlo implementation of the matrix-vector

[4] The subscript on the expectation value is a reminder that the average is over the distribution of paths (10.9).

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

Algorithm 35 Monte Carlo matrix-vector multiplication.

Input: Stack W of walkers and procedures to calculate (or look up) the transition probabilities and weight transfer matrices P and w.
Initialize a new stack W' ;
repeat
Pop a walker (with weight w_k in state $|k\rangle$) off W ;
Sample i from P_{ik} ;
Update the weight $w_i = w_{ik} w_k$;
Push a walker in state $|i\rangle$ with weight w_i onto W' ;
until W is empty.
return W'.

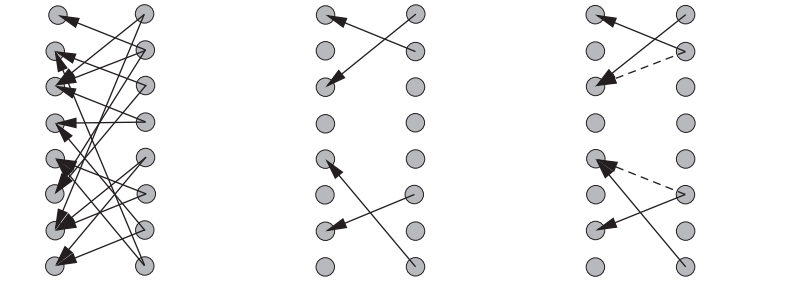

Figure 10.1 Matrix-vector multiplication. A column of open circles represents the possible states in the systems. When performed exactly, each old state contributes to multiple new ones, and a new state receives contributions from multiple old ones. When performed by Monte Carlo, a sample of old states contributes to a sample of new states, usually on a one-to-one basis. The right panel is referenced in Section 10.6.2.

multiplication in the direct power method. The other parts of a Monte Carlo power method are practically the same as in any other Monte Carlo method: We initialize and perform a number of steps until we equilibriate. In the equilibration stage, while stepping, we periodically make measurements. After we collect a sufficient number of measurements we stop.

We monitor convergence to stationarity in the Monte Carlo power method in the same way we would for any other Markov chain Monte Carlo simulation. In contrast to a deterministic power method, we do not explicitly monitor the decrease in the norm of the residual as a measure of convergence as we would in a deterministic method, because having only a sample of the vector components, we have insufficient information to compute the inner product between two different vectors

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

and the subtraction of two vectors. We monitor the convergence of an estimator of the eigenvalue. We discuss estimators of the eigenvalue in Section 10.5. We note that the Monte Carlo algorithm we need to insert a new type of step in both the equilibration and measurements stages: *Occasionally stochastically reconfigure*. We will say more about this in Section 10.3.

10.2.2 Monte Carlo inverse power method

We now shift our attention to solving the linear system of equations in a Monte Carlo implementation of the inverse shifted power method. For notational simplicity, we again start by absorbing the σI term into the definition of A, but now we rewrite the first equation in (10.4) as

$$|\psi\rangle = (I - A)\,|\psi\rangle + |\phi\rangle.$$

With the extra definition of $B = I - A$ and the expression of the states and operator B in some basis, where, for example, $|\psi\rangle = \sum_i w_i |i\rangle$, the equation of interest is

$$w_i = w_i^0 + \sum_j B_{ij} w_j.$$

On iteration this equation becomes the Neumann series

$$w_i = w_i^0 + \sum_j B_{ij} w_j^0 + \sum_{jk} B_{ij} B_{jk} w_k^0 + \sum_{jkl} B_{ij} B_{jk} B_{kl} w_l^0 + \cdots, \tag{10.10}$$

which we rewrite more compactly as

$$w_i = \sum_{k=1}^{\infty} \sum_{i_1} \cdots \sum_{i_k} \delta_{i i_k} B_{i_k i_{k-1}} \cdots B_{i_2 i_1} w_{i_1}^0. \tag{10.11}$$

The task in Monte Carlo sampling (10.11) is more ambitious than that for (10.7). We can do the multiplications in a manner similar to the way we did them for (10.7), but we need to add a way to "sample" the number of terms in the power series. A stopping probability is the means to this end.

To proceed we note that the Neumann series converges if the matrix norm of B satisfies $\|B\| < 1$. Examples of *matrix norms* are (Meyer, 2000)

$$\|B\|_1 = \max_i \left(\sum_j |B_{ij}| \right), \quad \|B\|_\infty = \max_i \left(\sum_j |B_{ij}| \right). \tag{10.12}$$

A matrix norm less than unity implies that the spectral radius of the matrix is less than unity. The *spectral radius* of a matrix defines a circle centered at the origin in the complex plane such that all eigenvalues of the matrix lie on or within this circle. If the spectral radius of B exceeds unity, then rescaling B is necessary.

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

Assuming $0 \leq B_{ij}$ and $0 < \sum_j B_{ij} < 1$, we next define

$$B_{ij} = w_{ij} P_{ij}, \quad \sum_i P_{ij} = 1 - p_j,$$

and rewrite (10.11) as

$$w_i = \sum_{k=1}^{\infty} \sum_{i_1} \cdots \sum_{i_k} \delta_{ii_k} W_k p_{i_k} P_{i_k i_{k-1}} \cdots P_{i_2 i_1}, \tag{10.13}$$

where

$$W_k(\gamma) = w_{i_k i_{k-1}} \cdots w_{i_2 i_1} w^0_{i_1} / p_{i_k}. \tag{10.14}$$

Here again, P_{ij} is the probability of a walker transitioning from state $|j\rangle$ to $|i\rangle$; however, its i-th row sum is less than one by an amount p_j. This amount is the probability of stopping at state $|j\rangle$.

Several things need noting. First, requiring all the elements of the B matrix to be nonnegative is an assumption that the problem under consideration has no sign problem. Assuming the spectral radius of this matrix is less than one so the series converges then means $\sum_i B_{ij} < 1$. Our choice of P_{ij} could then just as well be B_{ij} itself, in which case the definition of the weight transfer multiplier w_{ij} is unity. Instead of explicitly setting it to unity, we carry it along for hopeful insight in the structure of the procedures that evolve. Finally we now interpret the solution as being the following expectation value over all possible paths of the walker:

$$w_i = \langle W_i(\gamma) \rangle_\gamma. \tag{10.15}$$

With the above definitions we sample (10.13) in the following manner: We again sample the paths in steps. At the first step, we generate M random walkers from the distribution defined by w^0. Next, we process each walker in this sample, one at a time. If the walker is in state $|j\rangle$, whose current weight is w_j and $p_j < \xi$, then we sample P_{ij} for a new state $|i\rangle$, transition the walker to state $|i\rangle$, update its weight via $w_i = w_{ij} w_j$, and then repeat this step. If $p_j > \xi$, we update the walker's weight $w_j = w_j / p_j$, push the walker onto the new stack, and terminate its walk. Now, we pop a new walker off the original stack and restart the entire process. After we process the entire population of walkers (all that were in our original stack), we have completed one step, and then we repeat this procedure on the new walker population (the new stack).

The procedure just described performs the sampling in a Monte Carlo method that solves the linear system of equations in the inverse shifted power method. As for our discussion of the Monte Carlo direct shifted power method, we need to add steps that stochastically reconfigure the walkers and estimate the eigenvalue. We discuss these two topics in the following subsections.

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

Another way to sample (10.11) starts by rewriting it as

$$w_i = \sum_{k=1}^{\infty} \sum_{i_1} \cdots \sum_{i_k} \mathcal{W}_k p_{i_k} P_{i_k i_{k-1}} \cdots P_{i_2 i_1}, \tag{10.16}$$

where

$$\mathcal{W}_k(\gamma) = \sum_{m=1}^{k} W_m \delta_{i i_m}. \tag{10.17}$$

Now our solution becomes

$$w_i = \langle \mathcal{W}_i(\gamma) \rangle_\gamma.$$

Following Spanier and Gelbard (1969), we can show that this expectation value is equivalent to the previous one (10.15):

$$\begin{aligned}
\langle \mathcal{W}_i(\gamma) \rangle_\gamma &= \sum_{k=1}^{\infty} \sum_{i_1} \cdots \sum_{i_k} \left(\sum_{m=1}^{k} W_m \delta_{i i_m} \right) p_{i_k} P_{i_k i_{k-1}} \cdots P_{i_2 i_1} \\
&= \sum_{m=1}^{\infty} \sum_{k_m}^{\infty} \sum_{i_1} \cdots \sum_{i_k} W_m \delta_{i i_m} p_{i_k} P_{i_k i_{k-1}} \cdots P_{i_2 i_1} \\
&= \sum_{m=1}^{\infty} \sum_{i_1} \cdots \sum_{i_k} W_m \delta_{i i_m} P_{i_m i_{m-1}} \cdots P_{i_2 i_1} \\
&\quad \times \left\{ p_{i_m} + \sum_{i_{m+1}} p_{i_{m+1}} P_{i_{m+1} i_m} + \sum_{i_{m+2}} \sum_{i_{m+1}} p_{i_{m+2}} P_{i_{m+2} i_{m+1}} P_{i_{m+1} i_m} + \cdots \right\}.
\end{aligned}$$

Now we use the relation

$$p_{i_{m+1}} = 1 - \sum_{i_{m+2}} P_{i_{m+2} i_{m+1}}$$

to collapse the terms in the braces to $1 - \lim_{t \to \infty} P^t$ where P^t is the t-th power of the transition matrix. Because the spectral radius of P is less than one, $\lim_{t \to \infty} P^t = 0$ and we can then replace the term in the braces by one. The remainder of the expression is the same as (10.13).

What (10.16) and (10.17) say is that at each step we clone the walker, update its weight by dividing by the stopping probability for the current state, and push it onto the new stack. Then we let the walker proceed with the next step until an actual termination is sampled. In short, we accumulate contributions to the solution to the system of equations as the walker steps along. Sometimes, this way of sampling is more efficient than the previous one. Algorithm 36 details one way to execute this procedure.

Algorithm 36 Monte Carlo linear system solver.

Input: Stack W of walkers and the stopping and transition probabilities p and P.
  Initialize stack W' ;
  repeat
    Pop a walker (with weight w_k in state $|k\rangle$) off W ;
    Draw a ζ ;
    while $\zeta < 1 - p_k$ **do**
      Set $w \leftarrow w_k/p_k$;
      Push a walker in state $|k\rangle$ with w onto W' ;
      Sample m from P_{mk} ;
      Set $w_m = w_{mk} w_k$;
      Set $k = m$;
      Draw a ζ ;
    end while
  until W is empty.
  return W'.

Algorithm 35 performs the sampling in a second Monte Carlo method that solves the linear system of equations in the inverse shifted power method. As for our discussion of the direct shifted Monte Carlo power method and the first Monte Carlo method to solve a linear system of equations, we need to add steps that stochastically reconfigure the walkers and estimate the eigenvalue.

This second way of sampling is an illustration of the method of *expected values* (Spanier and Gelbard, 1969; Kalos and Whitlock, 1986). The method of expected values is a common procedure to reduce the variance of a Monte Carlo estimate of an average. It says that doing parts of the estimate exactly, instead of letting the Monte Carlo estimate it, under certain conditions reduces the variance of the entire result. In the present case, instead of recording the walker's contribution to the solution when it stops, we observe that at each step along the path it could have stopped. We record at each step along the path what we expect to happen on the average.

We remark that when using the power method deterministically, the requirement that A (or B for the linear system problem) has no negative matrix elements is unnecessary. In a Monte Carlo power method, direct or inverse, when an A_{ij} is negative, the absolute value of A_{ij} is used whenever we need to use it to define a transition probability, and its sign multiplies the weight transfer factor w_{ij}. If A has no negative matrix elements, then the *Perron-Frobenius theorem* (Section 2.4.2; Meyer, 2000) applies, saying that the dominant eigenvalue must be real and positive and the components of its eigenvector in this basis must be real and positive.

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

In our algorithms, we push walkers onto a stack without attempting to combine those that are in the same state. In general, it is a rare event if they are. If the walkers all have positive weight, the walkers would combine constructively. In some sense, there is not much to gain by combining them. If they have mixed signs, they would combine destructively. In this case, by not combining them, we can lose important cancellation effects. Generally, we must add an extra procedure to accommodate walkers of mixed sign. In Section 10.6, we discuss one such method, originally developed for the Green's function Monte Carlo method (Section 10.4), that has been successfully used in excited state calculations. *The sign problem is not just a Fermion problem or even just a quantum problem*.

10.3 Stochastic reconfiguration

The above Monte Carlo algorithms are incomplete until we define what we mean by *stochastic reconfiguration*.[5] There are several intertwined concepts in this phrase. One is ensuring ergodicity, another is promoting efficiency, and a third is computing conveniently.

The problem is that the successive updating of the walker weights generates a wide range in the values of these weights: Some walkers develop very large weights, other walkers very small ones. Having a few large-weighted ones leads to just a few states underrepresenting the many possible states. For the sampling to be ergodic, we need to ensure all important states are properly accessed. One way of doing this is limiting the weight accumulation of any one walker by occasionally splitting a large-weighted one into a number of smaller-weighted ones that share the original weight. Each of the smaller-weighted walkers subsequently takes a path different from the original one. The splitting and termination changes the size of the walker population. Thus, we also have to ensure that the population does not become too large or too small, which requires a procedure to control the population size. The control and efficiency tasks are often combined into one procedure. We call the combination of Monte Carlo techniques that help accomplish these goals *stochastic reconfiguration*. They change the mix of the walkers in the current configuration. Doing this reconfiguration stochastically, and occasionally, can reduce the potential for an unwanted bias in the resulting estimates.

There are two classes of stochastic reconfiguration procedures. One class operates on the walkers one at a time, the other operates on the entire population at once. For the direct power algorithm, it is natural to reconfigure the entire population

[5] Sorella (1998, 2000) uses the term "stochastic reconfiguration" to describe a method to stabilize the sign problem in the linear Green's function Monte Carlo method (Section 10.4.1) using the lattice fixed node approximation (Section 11.2).

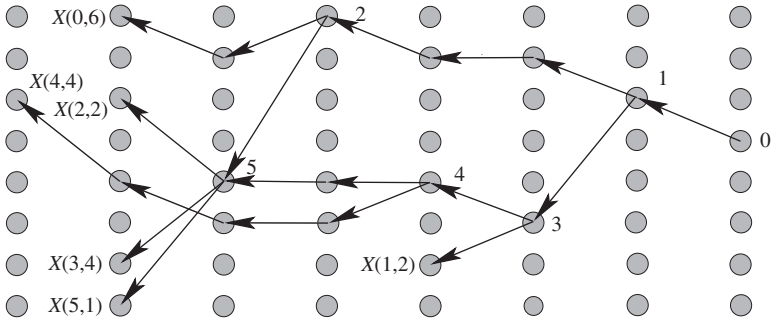

Figure 10.2 On-the-fly reconfiguration. A column of circles represents the possible states in the system, and each column represents the states after a power method step, moving right to left. Only one walker is being considered. This walker starts at the event labeled zero. Branching occurs at the nonzero event numbers, generating new walkers labeled by that event number. The bookkeeping symbol $X(m, n)$ marks the end of a walk: m is the event number of origin, and n is the number of steps from the origin to termination.

occasionally at the end of an iteration step. In the inverse power method, it is natural to follow a single walker until it terminates. Accordingly, walker control here is usually done individually and on the fly.

Figures 10.2 and 10.3 illustrate these two procedures. In Fig. 10.2, a single walker starts at the event labeled zero. Branching occurs at each other numbered event, generating new walkers labeled by that event number. The symbol $X(m, n)$ marks the end of a walk. Here, m is the event number of origin, and n is the number of steps from the event of origin to termination. Because branches are followed one at a time, walkers from different branches may occupy the same state, as they do in the state labeled event 5, but the occupancy is not simultaneous. When the initial walker and all the walkers it spawned terminate, the procedure moves to the next walker in the initial sample.

In Fig. 10.3 a sample of walkers starts simultaneously. For illustration purposes, we pretend we are reconfiguring at each step by splitting arbitrarily. At each step, the weights of the walkers change (this change is not illustrated in the figure), and as a result, some walkers terminate and others branch. During population reconfiguration, different walkers may occupy the same state simultaneously. If their weights are positive, the reconfiguration generally does not take this into consideration. If the weights are of mixed sign, it is advantageous to combine these walkers into one whose weight is the sum of the weights of those in the same state.

The iconic procedure for doing walker control on the fly is called *roulette*. It has a prespecified survival weight w_s and a cut-off weight $w_l < w_s$. If a walker's weight

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

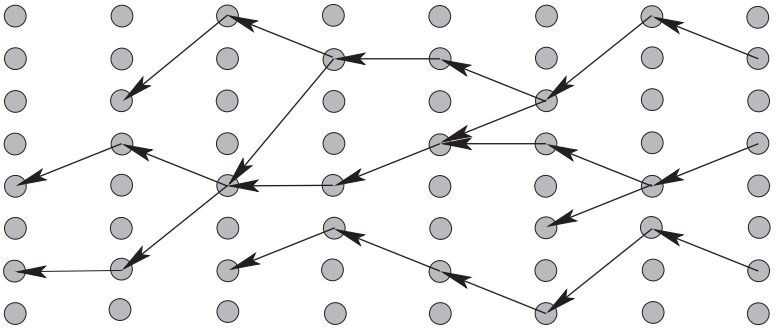

Figure 10.3 Population reconfiguration. A column of circles represents the possible states in the system, and each column represents the states after a power method step, moving right to left. For the purpose of illustration, reconfiguration occurs at each step. This procedure changes the weights of each walker in the population, leading to some branching and others terminating. Multiple walkers may occupy the same state.

w falls below w_l, then if $\zeta < w/w_s$, the walker is kept but its weight is increased to w_s. Otherwise, the walker is terminated.

The second commonly used procedure seems to lack a name, but we will call it *branching*. It addresses problems with weights being too large and too small. For this procedure, we calculate $m = \text{integer}(w + \zeta)$, where ζ is a random number in $[0, 1]$. The addition of ζ to weight reduces any bias the truncation to an integer might cause. If $m = 0$, we terminate the walker. If not, then we split it m times, with each replica assigned a weight w/m.

A third procedure, called *weight window*, offers better control at the expense of having several parameters to adjust (Booth, 2009). This method combines roulette and branching. The weight window has a lower, survival, and upper weight arranged so that $w_l < w_s < w_u$ with $w_u \geq 2w_l$. Typically, $w_u = 5w_l$ and $w_s = 3w_l$. As long as the walker's weight is in the weight window $[w_l, w_u]$, we don't do anything. If the walker's weight falls below w_l, we roulette it. If it rises above w_u, we branch it by finding the smallest integer m that places w/m within the weight window and split the walker m times. This method should generally be preferred over branching because it preserves the total weight, while branching introduces weight fluctuations and preserves only the total weight on the average. Algorithm 37 details the weight window algorithm.

In any of the on-the-fly procedures that split the walker into m walkers, $m - 1$ walkers with their current state and modified weights are pushed back onto the list of current walkers. The remaining walker, with its new weight, continues the walk. When its walk terminates, then a new walk starts with one of the waiting walkers. One iteration is completed when the stack of original and spawned

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

Algorithm 37 The weight window.

Input: A walker with weight w in state $|i\rangle$, a stack W of walkers plus the upper and lower cut-off w_u and w_l, survival weight w_s.
  if $w > w_u$ **then**
    Set $m = \text{floor}(w/w_u)$;
    $w = w_s$;
    Push $m - 1$ walkers with weight w in state $|i\rangle$ onto W ;
  else if $w < w_l$ **then**
    Set $p = w/w_s$;
    if $\zeta < p$ **then**
      $w = w_s$;
      Push a walker with weight w in state $|i\rangle$ onto W ;
    else
      $w = 0$;
    end if
  end if
  return the walker and W.

walkers depletes. An important point is that since the population of walkers sums the Neumann series, there is no need to terminate a walker arbitrarily on the basis of its weight falling below some prescribed value or exceeding a number of steps. Doing either of these terminations introduces a bias. The procedures just described stochastically terminate a walker in a way that at least preserves its weight on the average.

We could apply the on-the-fly techniques to the entire population of walkers in one swoop. Doing so however may generate a large sudden variation in population size and total weight. Instead, we use one of several convenient techniques for stochastically reconfiguring the entire population of walkers and resetting the population to a fixed size. The simplest procedure is *sampling with replacement*. Here, occasionally, but after an iteration step, we create the cumulative distribution function (CDF) of the walker weights, $C_i = \sum_{j=1}^{i} w_j$, and then from it sample with replacement M walkers with their weights and states tagging along. Those with small weights have a low probability of being selected, while those with large weights may get selected multiple times.

A second procedure is *stratified sampling* (Spanier and Gelbard, 1969; Kalos and Whitlock, 1986). Again we form the cumulative distribution of walker weights. After doing so, we divide its 0 to 1 range into a number of strata, for example, based on weight percentiles or fractions of walkers. Then we sample with replacement from each strata a prefixed number of walkers such that the total from all strata

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

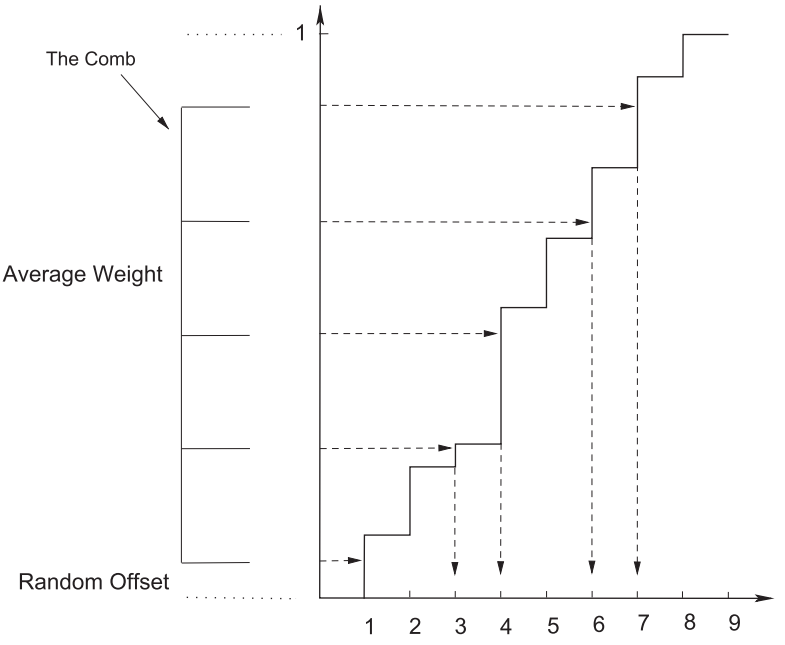

Figure 10.4 The comb. This figure illustrates the comb with equal spacing (average weight) between its teeth. The main figure is the cumulative distribution function (CDF), C_i, for nine walkers originally. The comb is randomly offset from the CDF by a random shift. Sampling with the comb assigns each walker the average weight per walker, the distance between the teeth. Here there are five walkers. The number of times a walker is selected is given by the number of teeth that lie in an interval of the CDF. In the present illustration, the original walkers 1, 3, 4, 6, and 7 are selected once. The others are not selected.

is the desired number. For example, if we want M samples, we could divide the [0, 1] range of the cumulative distribution into M equal intervals and then draw one sample from each interval.

Still another procedure is the *comb* (e.g., Booth, 2009; Booth and Gubernatis, 2009b). It preserves the before and after total weight, eliminating a fluctuation, and uses only one random number (Fig. 10.4). Each selected walker has the same weight. If we have M walkers and want N, where M could be greater than or less than N, the value of the common weight is $w_{\text{average}} = \sum_{i=1}^{M} w_i/N$. Walker splitting and termination starts with the formation of the cumulative distribution C_i from the weights of M walkers. We point out, however, that we can compute the cumulative distribution on the fly, eliminating the need to store it. We detail the comb in Algorithm 38.

The three on-the-fly and the three population reconfiguration methods are *variance reduction methods*. Their individual effectiveness for this purpose, however,

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

Algorithm 38 The comb.

Input: Stack W on M walker and N, the size of the new stack.
  Initialize the new stack W' ;
  Set $C = 0$;
  Set $\delta = 1/M$;
  Compute the total weight w_{total} of walkers in W ;
  Set $w_{\text{average}} = w_{\text{total}}/N$;
  Draw a random number $\zeta \in [0, 1]$;
  Set $\Delta = \zeta\delta$;
  repeat
    Pop a walker (with weight w in state $|i\rangle$) off W ;
    $C \leftarrow C + w$;
    while $\Delta > C/w_{\text{total}}$ **do**
      Push walker of weight w_{average} in state $|i\rangle$ onto W' ;
      $\Delta \leftarrow \Delta + \delta$;
    end while
  until W is empty.
  return W'.

can vary widely. In variance reduction, we seek to get the biggest "bang for the buck," that is, the smallest variance for a given amount of computer time. In general, the objective of variance reduction is minimizing the size of the fluctuations of individual measurements about their averages. Its objective therefore differs from that of population control. Often the two intents are confused. Eliminating branching is not always a good idea from the point of view of variance reduction, although it is convenient for population control. As noted above, branching promotes ergodic sampling so its elimination might in fact be ill-advised. In a similar vein, resetting the population to a fixed size, even with branching included, is also convenient for population control, but whether the population is adjusted one walker at a time or all at once, the main objective should be bias and variance reduction obtained by setting the parameters of the simulation so a roughly constant population propagates from iteration to iteration. In each reconfiguration method described, observable averages before and after reconfiguration differ. This fluctuation adds to the variance. Thus, we want to make these changes as small and as infrequently as possible. One way of doing this is reducing fluctuations among the various walker weights; that is, we can try to keep them within a weight window. If we were reconfiguring by means of the weight window algorithm, for example, then we would virtually eliminate splittings and terminations if we were able to keep most of the weights within the window.

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

We comment that reconfiguring the entire population at once has two consequences. One is that the simulation is no longer Markovian. Reconfiguration introduces correlations between the walkers, since each walker's weight now depends on the others in the population. The correlations carry from one step to another. These correlations are a reason why we stated we reconfigure "occasionally." Reconfiguration should not be done after every iteration. Another consequence of reconfiguring the entire population at once is that a bias, which varies as the reciprocal of the number of walkers, generally develops in the estimates. Typically, the number of walkers is large, so this bias is small. If the computation time is affordable, the cure is simply increasing the number of walkers. Other procedures exist that help reduce the need for huge numbers of walkers. The series method of Brissenden and Garlick (1986) and Gelbard and Gu (1994) is easily applied to the block averages with noticeable positive consequences.

Reconfiguring on the fly does not eliminate these consequences. This type of reconfiguration changes both the total weight and the number of walkers. The overarching need is to maintain both at a nearly constant size throughout the simulation. Making the adjustment to do this can induce non-Markovian behavior and biases. In general, for both types of reconfiguration, these adverse effects of reconfiguration are small but should be noted. While the simulations may appear to implement what mathematically are Markovian and unbiased procedures and estimates, what occurs in practice can often be less than ideal.

10.4 Green's function Monte Carlo methods

In quantum ground state calculations, the power method projects to the dominant eigenpair of some operator A. For A, Green's function Monte Carlo methods use a function of the Hamiltonian H whose dominant eigenvalue maps to the subordinate eigenvalue of H. Three functions are used:

1. $A = I - \tau(H - \sigma I)$, which shifts and rescales the eigenspectrum of the Hamiltonian
2. $A = [I + \tau(H - \sigma I)]^{-1}$, which inverts a shifted and rescaled Hamiltonian
3. $A = \exp[-\tau(H - \sigma I)]$, which exponentiates the shifted Hamiltonian.

Because the first case uses a linear mapping of the eigenvalues, we call it the *linear method*. The Green's function method using the exponential is called *diffusion Monte Carlo*. The method using the inverse is called *exact Green's function Monte Carlo*. All three are often simply called Green's function Monte Carlo. Clearly, a lowest order power series expansion of the last two operators reduces both to the first. Indeed, as we discuss next, the three approaches have a common interpretation and share many techniques.

To develop this interpretation, we start with the time-dependent Schrödinger equation (Section 1.2)

$$i\frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle,$$

which after the substitution

$$|\psi(t)\rangle = |\psi\rangle e^{-itE}$$

becomes the eigenvalue equation

$$H|\psi\rangle = E|\psi\rangle, \tag{10.18}$$

called the time-independent Schrödinger equation. Associated with the inverse of the Hamiltonian is an operator G called the *time-independent* Green's function that by definition satisfies

$$HG = GH = I.$$

Multiplying (10.18) on the left with G yields the eigenvalue equation

$$|\psi\rangle = EG|\psi\rangle. \tag{10.19}$$

If we were to express this Green's function operator G in some basis $\{|C\rangle\}$, this equation would become the matrix-vector equation

$$\langle C|\psi\rangle = E\sum_{C'}\langle C|G|C'\rangle\langle C'|\psi\rangle.$$

The transformation $it \to \tau$ converts the time-dependent Schrödinger equation into the imaginary-time-dependent Schrödinger equation

$$\frac{\partial}{\partial \tau}|\psi(\tau)\rangle = -H|\psi(\tau)\rangle, \tag{10.20}$$

which upon formal integration creates an operator that enables the generation of the state at one imaginary time from another state at another time, that is,

$$|\psi(\tau_2)\rangle = e^{-H(\tau_2-\tau_1)}|\psi(\tau_1)\rangle = G(\tau_2-\tau_1)|\psi(\tau_1)\rangle.$$

This result defines the *imaginary-time-dependent* Green's function, which is sometimes called the *imaginary-time propagator*. Because of the implied time translation invariance, we rewrite the above as

$$|\psi(\tau_2)\rangle = \underbrace{G(\tau)\,G(\tau)\cdots G(\tau)}_{M\text{ factors}}|\psi(\tau_1)\rangle,$$

where $\tau_2 - \tau_1 = M\tau > 0$.

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

To obtain the state at one time in terms of a state at another time, we could also use a finite-difference approximation (Euler's method) to the imaginary-time derivative

$$\frac{|\psi(\tau_1+\tau)\rangle - |\psi(\tau_1)\rangle}{\tau} = -H\,|\psi(\tau_1)\rangle$$

or

$$|\psi(\tau_1+\tau)\rangle = [I-\tau H]\,|\psi(\tau_1)\rangle\,.$$

These considerations motivate calling the techniques "Green's function" methods, although this name is a bit deceptive. Our intent is computing the ground state properties and not solving the initial-value problem represented by the equation of motion (10.20). *Accordingly, for what we called the linear method, τ need not be small.* In general, τ is a parameter we use to rescale the eigenvalue spectrum of the shifted Hamiltonian so that the matrix G used in the power method has nonnegative matrix elements. For the exact Green's function method, we also use τ to scale the shifted Hamiltonian so that its spectral density is less than one, a necessary condition for the convergence of the Neumann series representing the inverse. τ is small in the diffusion Monte Carlo method because we use the Trotter approximation to evaluate the exponential. In short, when possible, we use a combination of τ and σ to create a nonnegative matrix representation of the operator. For some basis $\{|C\rangle\}$, necessary conditions for nonnegativity are

1. For $A = I - \tau(H-\sigma I)$: $\langle C|H-\sigma I|C'\rangle \le 0$ for all C and C'
2. For $A = [I+\tau(H-\sigma I)]^{-1}$: $\langle C|H-\sigma I|C'\rangle \le 0$ for all $C \ne C'$
3. For $\exp[-\tau A]$ with $A = (H-\sigma I)$: $\langle C|H-\sigma I|C'\rangle \le 0$ for all $C \ne C'$.

Of the three Green's function methods, the exact Green's function method, which is based on the inverse power method, is the one most infrequently used. We have given its basic definition and next focus on just discussing extra details of the other two methods.

10.4.1 Linear method

The linear method is convenient and generally simple to implement. As just discussed, we want the matrix elements of $A = I - \tau(H-\sigma I)$ in some basis $\{|C\rangle\}$ to be nonnegative and can ensure this if $C \ne C'$ and $\langle C|H-\sigma I|C'\rangle < 0$. As an example of how to construct the power method, we take for H the one-dimensional quantum Ising model in a transverse field (5.6) and choose for our basis the spin configuration basis (5.5).

The Hamiltonian is

$$H = H_0 + H_1 \tag{10.21}$$

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

with

$$H_0 = -J^z \sum_{i=1}^{N} S_i^z S_{i+1}^z \quad \text{and} \quad H_1 = -H^x \sum_{i=1}^{N} S_i^x. \tag{10.22}$$

As noted before (Section 6.5.2), this model is exactly solvable, so we do not need quantum Monte Carlo simulations to understand its properties. What we do know from the exact solution is that the model possesses a *quantum critical point* (Sachdev, 1999); that is, as a function of the transverse field, it has a phase transition at zero temperature.

To motivate this transition, we regard J^z as setting the scale of energy so that the properties of the model become a function of the coupling constant $g = H^x/J^z$. If this constant is very small, we expect the model to behave like the zero-field Ising model whose ground state is the state of long-range magnetic order with all spins up or down. On the other hand, if g is very large, the natural basis functions are not the $|C\rangle = |s_1\rangle|s_2\rangle \cdots |s_N\rangle$ but rather $|C\rangle = |t_1\rangle|t_2\rangle \cdots |t_N\rangle$ where the $|t_i\rangle$ are eigenfunctions of S_i^x. These states are $|1\rangle = \frac{1}{\sqrt{2}}(|+1\rangle + |-1\rangle)$ and $|2\rangle = \frac{1}{\sqrt{2}}(|+1\rangle - |-1\rangle)$ with eigenvalues $\lambda_1 = +1$ and $\lambda_2 = -1$. In the large coupling constant limit, the S_i^z independently fluctuate between these two states, and we expect the ground state to be a linear combination of all spin configurations, a quantum paramagnet (Sachdev, 1999). Thus, as a function of g, the ground state changes. The critical point in fact is at $g_c = \frac{1}{2}$, that is, where $J^z = 2H^x$.

From the above discussion, we discover that our chosen basis is not the most appropriate one for a phase of the model. Accordingly, our basis choice is unlikely to capture the physics of that phase easily. We also expect difficulty near the critical point.

We first note that H_0 is diagonal in our basis ($s_{i,k} = \pm\frac{1}{2}$)

$$\langle C_i|H_0 - \sigma I|C_j\rangle = \left[-\sum_k \left(J^z s_{i,k} s_{i,k+1} + \sigma \right) \right] \delta_{ij}. \tag{10.23}$$

We next note that H_1 is strictly off-diagonal:

$$\langle C_i|H_1|C_j\rangle \equiv -H^x \sum_k \delta_{C_i^{jk} C_j}.$$

H_1 acting on some state $|C_j\rangle$ produces the sum of N states $|C_i^{jk}\rangle$ where each differs from $|C_j\rangle$ only at lattice site k where it has the flipped spin $-s_{j,k}$. Thus, for $H^x > 0$ all off-diagonal matrix elements are negative, and from (10.23), it is clear that we can make all diagonal elements non positive if $\sigma = J^z$. Accordingly,

$$\langle C_i|I - \tau\,(H - \sigma I)\,|C_j\rangle = \sum_k w_{jk} \left[P_{jk} \delta_{C_i, C_j} + \bar{P}_{jk} \delta_{C_i^{jk}, C_j} \right],$$

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

with

$$P_{jk} = \left[1 + \tau J^z \left(s_{j,k} s_{j,k+1} + 1\right)\right] / w_{jk}, \quad \bar{P}_{jk} = \tau H^x / w_{jk},$$

and

$$w_{jk} = 1 + \tau J^z \left(s_{j,k} s_{j,k+1} + 1\right) + \tau H^x.$$

By unveiling possible transition probabilities P and weight transfer multipliers w, the result leads to a simple sampling strategy we can use in Algorithm 35 to generate a sweep: For a walker representing a spin configuration $|C_j\rangle$, we choose a lattice site k at random, update the walker's weight by $w \leftarrow w w_{jk}$, and then if $P_{jk} < \zeta$, where $\zeta \in [0, 1]$ we keep the walker in its current state or else we jump it to a new state formed from the old one by flipping the spin state at the lattice site k. In the present case, $\tau > 0$ is a free parameter for adjusting the efficiency of the sampling, for example, keeping the flipping and nonflipping rates roughly equal as the coupling constant is varied.

Algorithm 35 generates different states, each of which is a spin configuration; that is, it generates different populations of walkers. With them a variety of measurements are possible (Section 10.5). How well does this algorithm work? We do not know. We constructed it mainly for illustration purposes. In general, it is hard to predict the efficiency of an algorithm. During the implementation, usually an understanding emerges about the degree of efficiency and the location of bottlenecks. Since for this problem the exact answer is known, trying to reproduce it becomes a learning opportunity to experiment with the proposed algorithm and to improve it.

10.4.2 Diffusion Monte Carlo

To illustrate the diffusion Monte Carlo method, we again take the transverse-field Ising model as our example and continue using the spin configuration basis. This method differs from the linear method by requiring that we exponentiate the Hamiltonian. To do the exponentiation, we have to use a Trotter approximation. In the present case we need it because the spin operators S_i^x and S_j^z lack on-site ($i = j$) commutivity, so that the spin-spin Hamiltonian H_0 does not commute with the spin-field Hamiltonian H_1. These two parts of H are thus not simultaneously diagonalizable. Accordingly, the states in (5.5) are not eigenstates of the Hamiltonian (10.21). However, in this basis both $\langle C'| \exp(-H_0/kT)|C\rangle$ and $\langle C'| \exp(-H_1/kT)|C\rangle$ are easily found. In contrast to the linear method, τ must be small so the Trotter approximation is accurate.

We also need slightly different techniques for sampling the matrix-vector multiplication, but we still have to ensure that in our chosen basis the matrix elements

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

of the exponential are nonnegative. The condition to ensure nonnegativity is the same as for the linear method. However, this condition ensures the positivity of the exact exponentiation and not of the individual pieces of the Trotter approximation. The positivity of each piece has to be established separately. In any case, in what follows we take $\sigma = J^z$.

There are several different ways to construct the algorithm. Our choice is somewhat arbitrary. We first write

$$H - \sigma I = -\sum_{k=1}^{N} \left[J^z \left(S_k^z S_{k+1}^z + I \right) + H^x S_k^x \right],$$

define the terms $h_{k,k+1}^{(0)} = -J^z \left(S_k^z S_{k+1}^z + I \right)$ and $h_{k,k+1}^{(1)} = -H^x S_k^x$, and then use the following form of the Trotter approximation (5.13),

$$e^{-\tau H} \approx \prod_k e^{-\frac{1}{2}\tau h_{k,k+1}^{(0)}} e^{-\tau h_{k,k+1}^{(1)}} e^{-\frac{1}{2}\tau h_{k,k+1}^{(0)}}.$$

The matrix elements of exponentials of the individual factors are easily calculated:

$$\langle C_i | e^{-\frac{1}{2}\tau h_{k,k+1}^{(0)}} | C_j \rangle = e^{\frac{1}{2}\tau J^z \left(s_{j,k} s_{j,k+1} + 1 \right)} \delta_{C_i C_j},$$
$$\langle C_i | e^{-\tau h_{k,k+1}^{(1)}} | C_j \rangle = \cosh\left(\tfrac{1}{2}\tau H^x \right) \delta_{C_i C_j} + 2 \sinh\left(\tfrac{1}{2}\tau H^x \right) \delta_{C_i^{jk} C_j}.$$

Thus,

$$\langle C_i | e^{-\tau (H - \sigma I)} | C_j \rangle \approx \prod_k w_{jk} \left[P_{jk} \delta_{C_i C_j} + \bar{P}_{jk} \delta_{C_i^{jk} C_j} \right],$$

where

$$P_{jk} = e^{\tau J^z s_{j,k} s_{j,k+1}} \cosh\left(\tfrac{1}{2}\tau H^x \right) \Big/ w_{jk}, \quad \bar{P}_{jk} = 2 \sinh\left(\tfrac{1}{2}\tau H^x \right) \Big/ w_{jk},$$

and

$$w_{jk} = e^{\frac{1}{2}\tau J^z s_{j,k} s_{j,k+1}} \cosh\left(\tfrac{1}{2}\tau H^x \right) + 2 \sinh\left(\tfrac{1}{2}\tau H^x \right).$$

A sampling strategy emerges that we can fit into Algorithm 35. For a walker of weight w representing a spin configuration $|C_j\rangle$, we choose the first lattice site $k = 1$, update the walker's weight by $w \leftarrow w w_{jk}$, and then if $P_{jk} < \zeta$, where $\zeta \in [0, 1]$, we keep the walker in its current state, updating its weight via $w \leftarrow w \exp[-\frac{1}{2}\tau J^z s_{j,k} s_{j,k+1}]$, or else we jump it to a new state formed from the old one by flipping the spin at the lattice site $k = 1$. We then move to the next lattice site ($k = 2$) and repeat the process until a sweep of the lattice is completed. We note that for very small τ, P_{jk}, $\bar{P}_{jk}$, and w_{jk} reduce to those in the linear method. The method

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

here differs by successively updating the walkers' spin configuration through the entire lattice before moving to the next walker.

Using the above in Algorithm 35 is one way to generate configurations in the form of a population of walkers. With these configurations, we can make a variety of measurements. One difference in quantum uses of the power method is that they invariably modify the transition probability (and hence the weight transfer multiplier) by using what is called importance sampling. Proper importance sampling is a very important performance issue.

10.4.3 Importance sampling

Importance sampling is a common Monte Carlo technique to reduce variance. We first define it and then discuss its use in the diffusion Monte Carlo method. Its use in the linear and exact Green's function methods follows by analogy. It is difficult to find any quantum Monte Carlo power method that does not use importance sampling. We follow the standard route to the introduction of the technique.

Suppose we seek a Monte Carlo estimate of

$$\langle X \rangle = \int dx\, X(x) f(x), \tag{10.24}$$

where $f(x)$ is some probability distribution that is not especially convenient to sample. Also suppose $g(x)$ is a more convenient one. Then, instead of estimating (10.24) by sampling $f(x)$ to obtain $\langle X \rangle = \frac{1}{M} \sum_{i=1}^{M} X(x_i)$, we estimate

$$\langle X \rangle = \int dx \frac{X(x) f(x)}{g(x)} g(x) \tag{10.25}$$

by sampling $g(x)$ to obtain

$$\langle X \rangle \approx \frac{1}{M} \sum_{i=1}^{M} \frac{X(x_i) f(x_i)}{g(x_i)}.$$

The variance of X when $g(x)$ is sampled is

$$\sigma_X^2 = \int dx \left[\frac{X(x) f(x)}{g(x)} \right]^2 g(x) - \langle X \rangle^2.$$

If we want the $g(x)$ that minimizes this variance, subject to the normalization constraint $\int dx\, g(x) = 1$, then it is straightforward to show by substitution into the above that the optimal (zero variance) importance function is

$$g(x) = \frac{X(x) f(x)}{\langle X \rangle},$$

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

a comforting result that says if we know the answer, that is, $\langle X \rangle$, then the Monte Carlo measurement will produce it! The more practical inference is that a good importance function reduces the variance. A poor one, however, can increase it. Unfortunately, there are few good rules to guide the choice of this function other than the rule that $g(x)$ should "cover" $f(x)$ closely but have longer tails.

If we have a good starting state $|\phi\rangle$, which is more conventionally written as $|\psi_T\rangle$ for the power method, we can use this same state for the importance function $\langle C|\psi_G\rangle \equiv \langle C|\psi_T\rangle$. The starting states typically come from approximate theories or from the variational Monte Carlo method (Chapter 9). The importance function $|\psi_G\rangle$ is often called a *guiding function* as its use has the interpretation of guiding the walkers toward the important parts of configuration space. We assume that $|\psi_G\rangle = |\psi_T\rangle$. We also use this same $|\psi_T\rangle$ to estimate σ via $\sigma \equiv E_T = \langle \psi_T|H|\psi_T\rangle / \langle \psi_T|\psi_T\rangle$.

We perform importance sampling in a quantum Monte Carlo power method by transforming the basic iteration step (10.2)

$$\langle C|\psi\rangle = \sum_{C'} \langle C|A|C'\rangle \langle C'|\phi\rangle$$

to

$$\langle C|\psi_T\rangle \langle C|\psi\rangle = \sum_{C'} \frac{\langle C|\psi_T\rangle \langle C|A|C'\rangle}{\langle C'|\psi_T\rangle} \langle C'|\psi_T\rangle \langle C'|\phi\rangle.$$

This is a similarity transformation, $A' = SAS^{-1}$, of the original eigenvalue equation with a diagonal transformation matrix S whose nonzero elements are $\langle C|\psi_T\rangle$. The eigenvalues are unchanged but the eigenstates are not. What we now iterate is

$$\langle C|\bar{\psi}\rangle = \sum_{C'} \langle C|A|C'\rangle \frac{\langle C|\psi_T\rangle}{\langle C'|\psi_T\rangle} \langle C'|\bar{\phi}\rangle$$

where the ratio $\langle C|\psi_T\rangle / \langle C'|\psi_T\rangle$ is the analog of $f(x)/g(x)$ in (10.25) and becomes an extra factor contributing to the weight-transfer multiplier $w_{CC'}$. $\langle C|\bar{\psi}\rangle = \langle C|\psi_T\rangle \langle C|\psi\rangle$ and $\langle C|\bar{\phi}\rangle = \langle C|\psi_T\rangle \langle C|\phi\rangle$. The guiding function need not be the same as the starting state, but if the starting state is good, then it should be a good guiding function. One use of variational Monte Carlo is providing this function (Chapter 9).

10.5 Measurements

We now come to the promised discussion of measurements. In diffusion Monte Carlo the standard estimator of the eigenvalue is called a *mixed estimator* (Hammond et al., 1994),

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

$$E_0 \leq E_m \equiv \frac{\langle \psi_T | H | \psi \rangle}{\langle \psi_T | \psi \rangle},$$

as it uses both the starting state $|\psi_T\rangle$ and the projected state $|\psi\rangle$. Clearly, this expression yields an exact estimate as $|\psi\rangle$ approaches the ground state. We use this same estimator for any observable that commutes with H. For other operators, we use a combination of the mixed and variational estimates called the *extrapolated estimate*

$$\langle X \rangle = 2\frac{\langle \psi_T | X | \psi \rangle}{\langle \psi_T | \psi \rangle} - \frac{\langle \psi_T | X | \psi_T \rangle}{\langle \psi_T | \psi_T \rangle} + \mathcal{O}\left(\|\psi_T - \psi\|^2\right).$$

The diffusion Monte Carlo method often uses a special estimator of the ground state energy, the *growth estimator* (Hammond et al., 1994). As we previously remarked, in the power method the norm of the eigenstate estimates this eigenvalue. Writing the norm as $N(\psi) = \sum_C |\langle C|\psi\rangle|$, we note that if we are sampling from the ground state, which we assume to be real, then because this state is nodeless, $|\langle C|\psi\rangle| = \langle C|\psi\rangle$, and the norm of the new state produced by one iteration step in the ground state is

$$N(\phi) = \sum_C \langle C|\phi\rangle = e^{-\tau(E_0 - E_T)} N(\psi).$$

However, the energy over any τ step actually has a value E_τ that fluctuates about E_0, so we have

$$N(\phi) = \sum_C \langle C|\phi\rangle = e^{-\tau(E_\tau - E_T)} N(\psi).$$

Thus,

$$E_\tau = E_T - \frac{1}{\tau} \ln \frac{N(\phi)}{N(\psi)}$$

so that

$$E_0 \equiv \langle E_\tau \rangle.$$

Common and useful refinements of the measurement process are *forward walking* (Hammond et al., 1994) and *back propagation* (Zhang et al., 1997). To discuss back propagation, let us suppose we have propagated $|\psi_T\rangle$ in the stationary state by n steps to obtain $|\psi\rangle = A^n|\psi_T\rangle$ and have saved this state. Now suppose we take $m \ll n$ additional steps and save the m configurations leading to the new state $|\psi'\rangle$. Instead of computing an average via

$$\langle \mathcal{O} \rangle = \frac{\langle \psi_T | \mathcal{O} | \psi' \rangle}{\langle \psi_T | \psi' \rangle},$$

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

where $|\psi'\rangle = A^m|\psi\rangle$, we use

$$\langle \mathcal{O} \rangle = \frac{\langle \psi_T | A^m \mathcal{O} | \psi \rangle}{\langle \psi_T | A^m | \psi \rangle} = \frac{\langle \psi'_T | \mathcal{O} | \psi \rangle}{\langle \psi'_T | \psi \rangle},$$

that is, we do not make measurements with the configuration at the current $(m + n)$-th step, but we make it with the configuration of a past step at n and use the intervening configurations to project $|\psi_T\rangle$ m steps closer to the ground state. We saw something similar in the ground state determinant method (Appendix I). There, we made measurements around the $\beta/2$ point in the projection whose imaginary-time length was β.

Without importance sampling, the Monte Carlo calculation estimates

$$|\psi\rangle = \sum_C w_C |C\rangle,$$

and accordingly for any observable $\mathcal{O}$, its mixed estimate is

$$\langle \mathcal{O} \rangle = \frac{\sum_C w_C \langle \psi_T | \mathcal{O} | C \rangle}{\sum_C w_C \langle \psi_T | C \rangle}.$$

Importance sampling, however, returns a weight $\bar{w}_C = w_C \langle \psi_T | C \rangle$ instead of w_C. In terms of the importance sampled weights, the mixed estimate becomes

$$\langle \mathcal{O} \rangle = \frac{\sum_C w_C \langle \psi_T | C \rangle \frac{\langle \psi_T | \mathcal{O} | C \rangle}{\langle \psi_T | C \rangle}}{\sum_C w_C \langle \psi_T | C \rangle} = \frac{\sum_C \bar{w}_C \frac{\langle \psi_T | \mathcal{O} | C \rangle}{\langle \psi_T | C \rangle}}{\sum_C \bar{w}_C}.$$

10.6 Excited states

Excited state information is important, and with the power method a few such states are often accessible. Computing them accurately is usually difficult. Thus, their computation is infrequent. We now briefly discuss two methods for computing excited states. The first is a classic with a track record, and the other is new and less developed. The classic method (Ceperley and Bernu, 1988), called the *correlation function quantum Monte Carlo method*, starts with the standard variational method for computing excited states and adds features of the variational and diffusion Monte Carlo methods to it. The excited states are determined concurrently, not one at a time.

10.6.1 Correlation function Monte Carlo

In the deterministic variational method, if we want to compute M excited states, we need M trial states, $\{|\psi_{T,i}\rangle, i = 1, 2, \ldots, M\}$. If we parametrize these states in terms

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

of a linear combination of some set of basis states $\{|\phi_i\rangle, i = 1, 2, \ldots, N \geq M\}$, with an N not less than M, such that

$$|\psi_{T,i}\rangle = \sum_j |\phi_j\rangle N_{ji},$$

then minimizing the Rayleigh quotient with respect to the elements of the matrix N yields the generalized eigenvalue problem

$$A|\psi\rangle = \nu S|\psi\rangle,$$

where the Hamiltonian and overlap matrices are $N \times N$ matrices defined as

$$A_{ij} = \langle\phi_i|H|\phi_j\rangle, \quad S_{ij} = \langle\phi_i|\phi_j\rangle.$$

The computed eigenvalues ν_i interlace the eigenvalues λ_i of A (9.5); that is, there is a set of the eigenvalues of A such that $\lambda_1 \leq \nu_1 \leq \lambda_2 \leq \nu_2 \cdots$ (Section 9.1).

In the correlation function quantum Monte Carlo method (Chapter 9), we first generate the best possible trial states from a variational Monte Carlo calculation or a mean-field theory and then use them in the generalized eigenvalue problem

$$\bar{A}|\phi\rangle = \nu\bar{S}|\phi\rangle,$$

where

$$\bar{A}_{ij}(\tau) = \langle\phi_i|Ae^{-\tau A}|\,\phi_j\rangle, \quad \bar{S}_{ij}(\tau) = \left\langle\phi_i|e^{-\tau A}|\phi_j\right\rangle.$$

Next we create a sampling density by adding a guiding function $|\psi_G\rangle$ and express these matrices in a form similar to those of the Hamiltonian and overlap matrices in the variational Monte Carlo method (Section 9.3),

$$\bar{A}_{ij}(\tau) = \sum_C \frac{\langle\phi_i|e^{-\tau A/2}|C\rangle}{\langle\psi_G|C\rangle} \frac{\langle C|Ae^{-\tau A/2}|\phi_j\rangle}{\langle C|\psi_G\rangle} \frac{\langle C|\psi_G\rangle^2}{\sum_C \langle C|\psi_G\rangle^2},$$

$$\bar{S}_{ij}(\tau) = \sum_C \frac{\langle\phi_i|e^{-\tau A/2}|C\rangle}{\langle\psi_G|C\rangle} \frac{\langle C|e^{-\tau A/2}|\phi_j\rangle}{\langle C|\psi_G\rangle} \frac{\langle C|\psi_G\rangle^2}{\sum_C \langle C|\psi_G\rangle^2}.$$

We need to sample a configuration from the distribution defined by the guiding function. Then we need to project all basis states and compute their overlap for the sampled configuration. We do the projection with diffusion Monte Carlo. As $\tau \to \infty$, the eigenvalues from the generalized eigenvalue problem using the Monte Carlo average of $\bar{A}$ and $\bar{S}$ approach those of the excited states overlapped by the trial states.

In choosing a basis for the variational trial states and also in choosing a guiding state, we must make these states general enough to avoid inadvertently missing an

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

excited state with a symmetry of interest. Of course, if we know the symmetries of interest, we tailor the basis and guiding function to incorporate them.

The original papers give satisfying detail about the validity of the method and useful information about its implementation. However, as we discussed in the variational Monte Carlo chapter (Section 9.3), newer methods for optimizing the trial state now exist, and it has been understood that while the Hamiltonian and overlap matrices in the generalized eigenvalue problem should be symmetric, symmetrizing the expression that arises upon sampling destroys the zero-variance property of the variational method.

10.6.2 Modified power method

The second method for computing excited states is quite different. The components of the excited eigenvectors must have mixed signs. The correlation function quantum Monte Carlo method never directly samples the excited states. The new method does and hence has to solve a sign problem. We call this new method the *modified power method* (Gubernatis and Booth, 2008; Booth and Gubernatis, 2009b). As with the correlation function quantum Monte Carlo method, it needs as many starting states as the number of eigenvalues sought and determines several eigenpairs concurrently. To simplify the discussion we assume we are seeking just two eigenpairs. The generalization to more than two is fairly obvious.

An important feature of this new method is that it converges to the extremal states, say, the ground state and first excited state, as $(\lambda_3/\lambda_1)^n$ for the ground state and as $(\lambda_3/\lambda_2)^n$ for the first excited state. (Recall (10.1).) Convergence is faster to the ground state and first excited state because during convergence the excited state is subtracted out of the ground state and vice versa as opposed to simply being reduced ("powered out") by the iteration. The modified power method explicitly projects to both states. The other state are "powered out." Details are given in Booth (2003a,b), Gubernatis and Booth (2008), and Booth and Gubernatis (2009b).

If the starting states are $|\phi_1\rangle$ and $|\phi_2\rangle$, the modified power method iterates

$$
\begin{aligned}
|\psi_1\rangle &= A\,|\phi_1\rangle,\\
|\psi_2\rangle &= A\,|\phi_2\rangle,\\
|\phi_1\rangle &= |\psi_1\rangle + \eta_1\,|\psi_2\rangle,\\
|\phi_2\rangle &= |\psi_1\rangle + \eta_2\,|\psi_2\rangle,\\
|\phi_1\rangle &\leftarrow |\phi_1\rangle \big/ \|\phi_1\|,\\
|\phi_2\rangle &\leftarrow |\phi_2\rangle \big/ \|\phi_2\|.
\end{aligned}
$$

The middle two equations prevent the method from being two independent power methods. If the iteration were two independent power methods, then both starting

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

states would project to the same ground state. The middle two equations make the projections dependent in such a way that if one state, say, $|\psi_1\rangle$, converges to the dominant state, the other converges to the subdominant state. It is through these two equations that $|\psi_2\rangle$ gets subtracted out of $|\psi_1\rangle$ and vice versa. The claim is that there exist two numbers, η_1 and η_2, that make this happen.

In the middle two equations, we want to form an eventual eigenstate from the linear combination of two states. Clearly the values of η_1 and η_2 cannot be arbitrary. To find them, we observe that if λ is an eigenvalue of A and the components of the associated eigenstate are ψ_i, then for any nonzero ψ_i[6]

$$\lambda = \frac{\sum_j A_{ij}\psi_j}{\psi_i}.$$

If this is true for one component, then a similar relation is true for a sum over any subset of components.

There are many ways of choosing such a subset, which we call a *region*. For any two regions R_1 and R_2, we must have

$$\lambda = \frac{\sum_{j\in R_1} A_{ij}\psi_j}{\sum_{i\in R_1}\psi_i} = \frac{\sum_{j\in R_2} A_{ij}\psi_j}{\sum_{i\in R_2}\psi_i}. \tag{10.26}$$

Now if $|\psi\rangle = |\psi_1\rangle + \eta|\psi_2\rangle$ is an eigenvector, and we substitute this state into the above equation and cross-multiply the numerators and denominators on opposite sides of the equal sign, we find that η must satisfy a quadratic equation

$$a\eta^2 + b\eta + c = 0,$$

whose coefficients are sums of products of the region sums:

$$\begin{aligned}
a &= \sum_{i\in R_2}\psi_{1i}\sum_{i\in R_1}\sum_j A_{ij}\psi_{2j} - \sum_{i\in R_1}\psi_{2i}\sum_{i\in R_2}\sum_j A_{ij}\psi_{1j},\\
b &= \sum_{i\in R_2}\psi_{2i}\sum_{i\in R_1}\sum_j A_{ij}\psi_{1j} - \sum_{i\in R_1}\psi_{2i}\sum_{i\in R_2}\sum_j A_{ij}\psi_{1j}\\
&\quad + \sum_{i\in R_2}\psi_{1i}\sum_{i\in R_1}\sum_j A_{ij}\psi_{2j} - \sum_{i\in R_1}\psi_{1i}\sum_{i\in R_2}\sum_j A_{ij}\psi_{2j},\\
c &= \sum_{i\in R_2}\psi_{1i}\sum_{i\in R_1}\sum_j A_{ij}\psi_{1j} - \sum_{i\in R_1}\psi_{1i}\sum_{i\in R_2}\sum_j A_{ij}\psi_{1j}.
\end{aligned}$$

[6] Until a deterministic power method converges, what we really should define is $\lambda(i) = \frac{\sum_j A_{ij}\psi_j}{\psi_i}$, an estimate of the eigenvalue that varies from component to component. As the deterministic power method converges, these estimates for different i converge to the same value. In a Monte Carlo simulation, while the global eigenvalue estimator converges, this local estimate still shows small variations.

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

At each step of the iteration, we compute these coefficients and solve the quadratic equation for its two roots. When its roots are both real,[7] we use each root in $|\psi\rangle = |\psi_1\rangle + \eta|\psi_2\rangle$, compute its eigenvalue estimate (10.26) using one of the regions, and then associate one root η_1 with the largest eigenvalue estimate $|\psi_1\rangle$ and the other root η_2 with the subdominant estimate. Doing this consistently causes the modified power method to converge to the dominant and subdominant states as claimed. By shifting the spectrum of A, we can enforce convergence to the ground state and first excited states.

For a deterministic implementation, whether we are seeking a few dominant eigenvalues of a matrix or a continuous operator, just about any choice of regions works. Additionally, the mixed signs of the elements of the matrix or of the components of its eigenvectors cause few problems because we do the matrix-vector multiplication exactly. A Monte Carlo implementation, which uses a sample of just a few components of the vector, misses important sign cancellations, and we are thus confronted with a sign problem.

To address the sign problem, we start by associating with each walker a state and two weights representing the components of the two eigenstates. If the matrix is nonnegative, one weight is positive while the other may have either sign. As the number of walkers becomes minuscule with respect to the number of available states, the likelihood that the Monte Carlo algorithm puts two walkers of opposite sign into the same state is small. We therefore add to the Monte Carlo sampling some procedures that promote these events. Several ways to do this have been developed (Booth and Gubernatis, 2009b; Rubenstein et al., 2010; Gubernatis, 2012). One way was actually developed for quantum Monte Carlo problems in the continuum (Arnow et al., 1982). We now describe it.

We are using a Monte Carlo algorithm to sample the sum $\sum_j A_{ij}x_j$. We start by expressing this product as a sum $\sum_{j\in\{k\}} A_{ij}x_j$ over the set of indices $\{k\}$ representing the current population of M walkers, which of course is only a small subset of the available states. Instead of sampling an i for a given j in $\{k\}$, we involve the entire population in the following way:

$$\sum_{j\in\{k\}} A_{ij}w_j = \sum_{j\in\{k\}} \frac{A_{ij}w_j}{\sum\limits_{j\in\{k\}} P_{ij}} \sum_{j\in\{k\}} P_{ij}$$

$$= \sum_{j\in\{k\}} \frac{A_{ij}w_j}{\sum_{j\in\{k\}} P_{ij}} \left[P_{ij_1} + P_{ij_2} + \cdots + P_{ij_M}\right].$$

[7] We could simply use the complex roots if we were to use complex arithmetic.

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

From the terms in the square brackets, we now sample an i_1 from P_{ij_1} to obtain a walker in state $|i_1\rangle$ with weight

$$\sum_{j\in\{k\}} A_{i_1 j} w_j \Bigg/ \sum_{j\in\{k\}} P_{i_1 j} \tag{10.27}$$

and i_2 from P_{ij_2} to obtain a walker in state $|i_2\rangle$ with weight

$$\sum_{j\in\{k\}} A_{i_2 j} w_j \Bigg/ \sum_{j\in\{k\}} P_{i_2 j} \ , \tag{10.28}$$

and so on. Consequently, each of the current walkers contributes multiple new walkers, instead of the usual one per walker. Each new walker typically receives contributions from multiple old walkers. This increase in population increases the likelihood of new walkers with opposite sign sharing the same state. It is easy to combine walkers in the same state and thus cancel signs that otherwise would be missed.

This importance sampling procedure scales as M^2 and proves computationally expensive. Instead of using the entire population at once, we can take all pairs of walkers, or all triplets, and so on. Generally there is little gain in grouping beyond a certain number. Or we may choose to use just subsets of the population most likely to have walkers of mixed sign. We illustrate the pair-wise sampling in the right-hand panel of Fig. 10.1. In this figure, the solid lines connecting the old and new states are the connections we sample with the standard Monte Carlo power method sampling. This sampling defines the new population. Then, for a given pair of old states, the dotted lines represent additional possible contributions to the new pair of states. We modify the weights of the new state according to (10.27).

The modified power method with proactive sign cancellation has been successfully applied to a number of classical problems and is just being explored for quantum problems. More details are given in a series of papers (Gubernatis and Booth, 2008; Booth and Gubernatis, 2009a; Rubenstein et al., 2010; Gubernatis, 2012). To date, its success surprisingly has been without the use of importance sampling. Much work remains before its full potential for quantum Monte Carlo is known.

Booth has proposed power methods for computing one excited state at a time without the knowledge of other eigenstates and methods to assess convergence (Booth, 2003a,b,c, 2010, 2011a,b,c).

10.7 Comments

As mentioned at the beginning of this chapter, most Monte Carlo techniques described here were originally developed for classical problems. In fact, we can

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

easily argue that the problem of neutron transport through fissile material, the original application that inspired the development of the Monte Carlo method, is the father of ground state quantum Monte Carlo methods in the continuum. The materials in these nuclear engineering problems are typically piecewise homogeneous. Each piece is characterized by scattering cross sections that determine the mean free path of a neutron and the probability of it being exited, absorbed, scattered, and fissioned. There are two classes of neutron transport problems, called *fixed source* and *criticality*. Fixed-source problems map onto solving a linear system of equations whose right-hand side (the known) is the source of the radiation. The criticality problem maps onto an eigenvalue problem. When the material produces as many neutrons by fission as those lost by absorption and escape, the system is critical with a dominant eigenvalue of unity. In a subcritical system, this eigenvalue is less than unity, while in a supercritical system, it is greater than unity.

In these transport problems, the neutron needs to move from one spatial point in a continuum to another. This movement is relatively easy to sample. Sampled is the distance to the next collision whose probability is an exponential that has the path lengths weighted by the reciprocal of the mean free path for the pieces of material transversed.

Diffusion Monte Carlo in the continuum borrowed many of its sampling methods from the neutron transport problem. A point of difference is the sampling of the movement of a particle (walker) through the continuum. In the quantum problem the Hamiltonian is exponentiated. The Trotter approximation expresses this exponential as a product of the exponential of the kinetic energy and the exponential of the potential energy (Section 7.1.1). In a position basis, the latter is diagonal. The former moves the particle and requires the computation of terms such as $e^{D\nabla^2}\psi(r)$. We can execute this operation via a Hubbard-Stratonovich transformation in the following manner (Zhang et al., 1997):

$$\begin{aligned} e^{D\nabla^2}\psi(r) &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} dx\, e^{-\frac{1}{2}x^2} e^{-i\sqrt{D}x\cdot\nabla}\psi(r) \\ &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} dx\, e^{-\frac{1}{2}x^2}\psi\left(r+\sqrt{D}x\right). \end{aligned}$$

Sampling x from the Gaussian produces movement from r to $r+\sqrt{D}x$, a diffusive motion. Accordingly, for problems in the continuum, diffusion Monte Carlo provides a convenient and almost necessary means to sample particle movement.

A curious difference between the quantum and nuclear engineering applications is the way importance sampling, in the sense of (10.4.3), is used. In the engineering problems different importance functions may be used for specific processes in different pieces of the material. The overall problem is too complex to have one

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press

importance function for the entire problem. Accordingly, the emphasis has been on the generation of variance reduction methods for the difficult pieces. In general, multiple techniques are used in multiple parts of the material.

The stochastic reconfiguration methods we discussed owe their existence to the neutron transport problem. In the quantum problems, however, we have means to generate a global importance function, for example, by the variational Monte Carlo method or a mean field theory. As a consequence, there has been less emphasis on exploring a range of methods for variance reduction. Usually, one stochastic reconfiguration method is used for the entire problem, making its intent more about controlling the size of the population. Compared with the spectrum of techniques used in the neutron transport simulations, the spectrum of techniques used in quantum Monte Carlo power method simulations appears somewhat monochromatic.

Ground state quantum Monte Carlo on a lattice in turn borrowed many techniques from ground state quantum Monte Carlo simulations in the continuum. In lattice models, the walkers generally move from one discrete state to another, whether or not we are using a discrete Hubbard-Stratonovich transformation (Section 7.1.1). Often the states available from any one state are very limited in number. To some extent this feature of lattice problems is underexploited. While diffusion Monte Carlo maintains advantages, the disadvantages of using the linear method and the exact Green's function method in the continuum are mitigated. Designing a ground state quantum Monte Carlo solution for lattice problems has more options. The elegance of the exact Green's function method becomes harder to keep bypassing.

The methods in this chapter bring us back to where Fermi started us in Chapter 1. We have illustrated that the imaginary-time Schrödinger equation has a Monte Carlo realization as a collection of walkers (particles) in which each independently performs a random walk while at the same time being subjected to multiplication of their weights at their locations in configuration space by parts of the Hamiltonian. In the Monte Carlo simulation the weight of the walkers decays exponentially in imaginary time at a rate controlled by the eigenvalue E_0 and the distribution of walkers $\langle C|\psi\rangle$. This eigenpair corresponds to the ground state solution of the Hamiltonian.

Suggested reading

G. Goertzel and M. H. Kalos, "Monte Carlo methods in transport problems," in *Progress in Nuclear Energy*, vol. 2 (New York: Pergamon Press, 1958), p. 315.

J. Spanier and E. M. Gelbard, *Monte Carlo Principles and Neutron Transport Problems* (Reading, MA: Addison-Wesley, 1969), chapter 2.

T. E. Booth, "Particle transport applications," in *Rare Event Simulation Using Monte Carlo Methods*, ed. G. Rubino and B. Tuffin (Chichester: John Wiley, 2009), p. 215.

Exercises

10.1 For a homogeneous, real, symmetric, tridiagonal matrix of order N whose nonzero elements are positive, find analytically the eigenpair associated with the largest and smallest eigenvalue.

10.2 For the matrix chosen in Exercise 10.1, use Algorithm 33 to find the eigenpair associated with the largest and smallest eigenvalue for several different values of N.

10.3 For the matrix chosen in Exercise 10.1, use Algorithm 35 to find the eigenpair associated with the largest and smallest eigenvalue for several different values of N. Use branching for stochastic reconfiguration.

10.4 Propose a quantum Monte Carlo algorithm for the transverse-field Ising model by adapting the steps for the linear method in Section 10.4.1 using the basis in which H_1 is diagonal. Do the analogous development for the transverse field Ising model for the diffusion Monte Carlo method described in Section 10.4.2.

10.5 For the matrix chosen in Exercise 10.1, use the modified power method deterministically to find the two eigenpairs associated with the two largest and the two smallest eigenvalues for several different values of N.

10.6 Propose a Monte Carlo method to solve the linear system of equations $\sum_j A_{ij}x_j = b_i$.

https://doi.org/10.1017/CBO9780511902581.011 Published online by Cambridge University Press