Perfect Stochastic Summation in High Order Feynman Graph Expansions[*]

J. N. Corcoran[†], U. Schneider[†], and H. -B. Schüttler[‡]
[†]University of Colorado and [‡]University of Georgia

April 1, 2003

Abstract

We describe a new application of an existing perfect sampling technique of Corcoran and Tweedie to estimate the self energy of an interacting Fermion model via Monte Carlo summation. Simulations suggest that the algorithm in this context converges extremely rapidly and results compare favorably to true values obtained by brute force computations for low dimensional toy problems. A variant of the perfect sampling scheme which improves the accuracy of the Monte Carlo sum for small samples is also given.

1 Introduction

The interacting fermion problem is of fundamental importance in a wide range of research areas, including fields as diverse as electronic structure theory of solids, strongly correlated electron physics, quantum chemistry, and the theory of nuclear matter [6, 11, 15]. The ultimate objective of this project, which is a massive collaborative effort between physicists, statisticians, and computer scientists, is to combine Monte Carlo summation techniques with self-consistent high-order Feynman diagram expansions into an efficient tool for the controlled approximate solution of interacting fermion models.

The concept of "self energy" is one of the most important examples of the power of Feynman diagram resummation. Suppose, for example that we have a lattice of atoms (such as in the case of a crystal) where electrons are almost localized in atomic orbitals at each site. Suppose further that we create a particle at one site and destroy a particle at another site

[*]Work supported in part by NSF Grants DMR-0081789 and DMR-9970291.

[†]Postal Address: Department of Applied Mathematics, The University of Colorado, Box 526 Boulder CO 80309-0526, USA; email: corcoran@colorado.edu, Uli.Schneider@colorado.edu; phone: 303-492-0685

[‡]Postal Address: Center for Simulational Physics, Department of Physics and Astronomy, The University of Georgia, Athens GA 30602, USA; email: hbs@physast.uga.edu; phone: 706-542-3886

[0]Keywords: invariant measures, backwards coupling, coupling from the past, exact sampling, perfect sampling, IMH algorithm, Monte Carlo summation, self energy, Feynman diagram, resummation, Hubbard model
AMS Subject classification: 60J10, 60K30, 82B80, 82D25

thereby adding and removing a quanta of vibrational energy to the system. Electrons of opposite spin occupying a single atom give rise to a Coulomb repulsion energy which causes particles to hop between sites. There is a contribution to the energy of the particle due to the virtual emission and absorption of particles. In other words, the particle interacts with its surrounding medium which, in turn, acts back on the particle. Essentially, a “wake” of energy is created around the movement of the particle. It is this “self energy”, described in more detail in Section 2.1, that we would like to quantify.

Self energy is represented as a large and complicated sum (“outer sum”) of terms that are, in themselves, large and complicated sums (“inner sums”). The objective of this paper is to describe the evaluation of these inner sums via a Monte Carlo approach. Monte Carlo summation techniques, detailed in Section 3, involve evaluating an approximating sum at random points drawn from a particular distribution. While there is much flexibility in the choice of this distribution, some choices are better than others in terms of minimizing the (first level)error in the approximating sum. The downside to choosing a nearly optimal distribution is that we often cannot sample points from it directly and are faced with a second level of error (sampling error) resulting from a Monte Carlo sampling approach. In this paper we choose a nearly optimal complicated distribution, yet sample from it using “perfect” simulation techniques, described in Section 4, that completely eliminate the second level error.

Simulation results are given in Sections 6 and 7.

2 Self Energy, Feynman Diagrams, and Computer Friendly Representation

The Hubbard model was originally developed in the early sixties in order to understand the magnetic properties of electron motion in transition metals. The original model remains a subject of active research today, and it is within this context that we describe our numerical approach. (The approach applies, with minor adaptations, to all interacting fermion models which possess a Feynman diagram expansion in terms of a two-body interaction potential.)

Suppose we have a lattice of atoms, representing atoms in a crystal, where electrons are almost localized in atomic orbitals at each site. We consider the following model Hamiltonian governing the motion and interactions between the particles.

$$H = \sum_{j',j} \frac{1}{2} V_{j',j}\, n_{j'}\, n_j - \sum_{j',j} \sum_{\sigma} t_{j',j}\, c^{\dagger}_{j'\sigma} c_{j\,\sigma} \tag{1}$$

Here

$c^\dagger_{j'\sigma}$	$=$	creation operator that creates an electron of spin $\sigma = \uparrow, \downarrow$ at site j'
$c_{j\,\sigma}$	$=$	destruction operator that destroys an electron of spin $\sigma = \uparrow, \downarrow$ at site j
$c^\dagger_{j'\sigma}\, c_{j\,\sigma}$	$=$	transfers an electron of spin σ from site j to site j'
n_j	$=$	the number of electrons at site j.

$V_{j'j}$ is called the model *Coulomb potential* and it includes the on-site ($j' = j$) Coulomb repulsion energy (U) generated when two electrons of opposite spin occupy a single atom as well as possibly extended (1st, 2nd, ... neighbor) repulsions.

$t_{j'j}$ is a parameter that controls an electron's ability to tunnel or "hop" between sites. If orbitals are highly localized, then this tunneling amplitude will decay exponentially with distance between sites. $t_{j'j}$ includes the on-site ($j' = j$) energy of the electron, a nearest neighbor tunneling amplitude (t) and possibly extended (2nd, 3rd ... neighbor) tunneling amplitudes. A depiction is given in Figure 1.

The total energy, given by (1), of this Hubbard model is thus represented as a sum of *potential* and *kinetic* energy given by the first and second terms, respectively, of (1). Both $V_{j'j}$ and $t_{j'j}$ are assumed to obey period boundary conditions on a finite lattice defined in Section 2.1.

Figure 1: The Hubbard Model

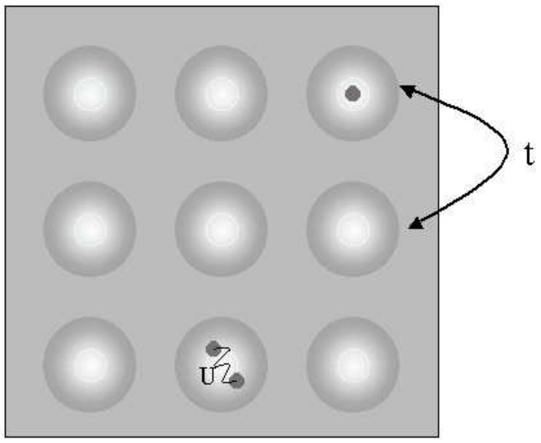


Illustration of the Hubbard model. The amplitude for an electron to hop from site to neighboring site is t and the Coulomb repulsion energy generated by two electrons of opposite spins at the same site is U.

2.1 Self Energy and Feynman Diagrams

The concept of self energy enables us to understand the feedback of the interacting environment on a propagating particle. Physically, it describes the cloud of particle-hole excitations that form the wake which accompanies a propagating electron.

The following paragraphs describe how to obtain the self energy $\Sigma(k)$. Basically, we will depict $\Sigma(k)$ through Feynman graph expansions and represent it as is a sum (of sums) involving Green's functions. (These Green's functions will in turn involve the self energy $\Sigma(k)$ we are trying to compute. This issue is addressed in Section 5.)

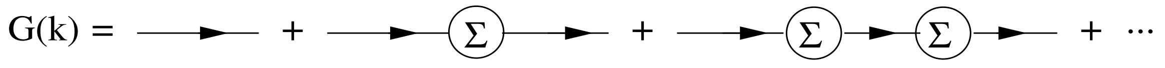
Green's function $G(k)$

The single-fermion Green's function $G(k)$ is the most basic physical quantity which can be obtained via a Feynman graph expansion. As it is expressed in terms of self energy, which we will denote by $\Sigma(k)$, we will describe our approach within this context. Here $k \equiv (\vec{k}, i\nu)$ denotes a $D+1$-dimensional "momentum-energy" variable (referred to as "momentum" hereafter), where $\vec{k}$ is a D-dimensional "wavevector" and $i\nu$ represents a frequency. These variables will be described in more detail later in this Section.

In general, a Green's function is the response of a linear system to a point input. Diagrammatically, we can represent the single-fermion Green's function, or *propagator*, as a sum of *free propagators* with various inserted scattering processes. (Mathematically, a propagator is simply an integral kernel.) The self energy represents all scattering processes combined into a single quantity. We depict the Green's function in terms of the self energy in Figure 2.

The self energy is in turn represented as a sum of *Feynman diagrams* as shown in Figure 3. These diagrams, developed by physicist Richard Feynman, give a convenient shorthand for representing complex mathematical quantities. The "wavy" lines represent photons that are emitted or absorbed by an electron traveling along the straight lines. Assignments of values to the momentum are made on various sections of the diagrams which correspond to arguments of functions in the algebraic self energy expression.

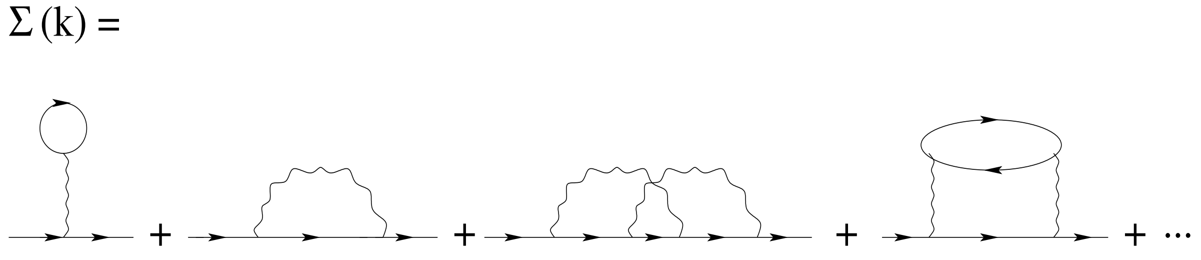
Figure 2: Representing a Green's Function as a Sum of Free Propagators with Inserted Scattering Processes



Algebraically, the free propagator has the form

$$G_0(k) = G_0(\vec{k}, i\,\nu) = \frac{1}{i\,\nu - \epsilon(\vec{k})}$$

Figure 3: Representing Self Energy as a Sum of Feynman Diagrams



where $\epsilon(\vec{k})$ is the *electron energy band*

$$\epsilon(\vec{k}) = -\mu - 2t \sum_{d=1}^{D} \cos(k^{(d)}),$$

and D is the lattice dimension. Here, μ and t are, respectively, the on-site electron energy (chemical potential) and nearest neighbor tunneling amplitude, discussed above in the description of the Hubbard model, and $k^{(d)}$ is the dth element of $\vec{k}$. In general, $\epsilon(\vec{k})$ is simply the lattice Fourier transform of $t_{j'j}$, which reduces to the above form in the case that only the first neighbor tunneling t and the on-site energy $-\mu$ are taken and included in $t_{j'j}$.

Following the Green's function representation from Figure 2, we may write

$$\begin{aligned} G(k) &= G_0 + G_0\Sigma G_0 + G_0\Sigma G_0\Sigma G_0 + \cdots \\ &= G_0 + G_0\Sigma G_0 + G_0(\Sigma G_0)^2 + \cdots \\ &= \tfrac{1}{G_0^{-1} - \Sigma} \end{aligned}$$

by summing the Taylor series expansion in powers of (ΣG_0).

This is known as the *Dyson equation* [11, 15]:

$$G(k) = G(\vec{k}, i\,\nu) = [i\,\nu - \epsilon(\vec{k}) - \Sigma(k)]^{-1}. \tag{2}$$

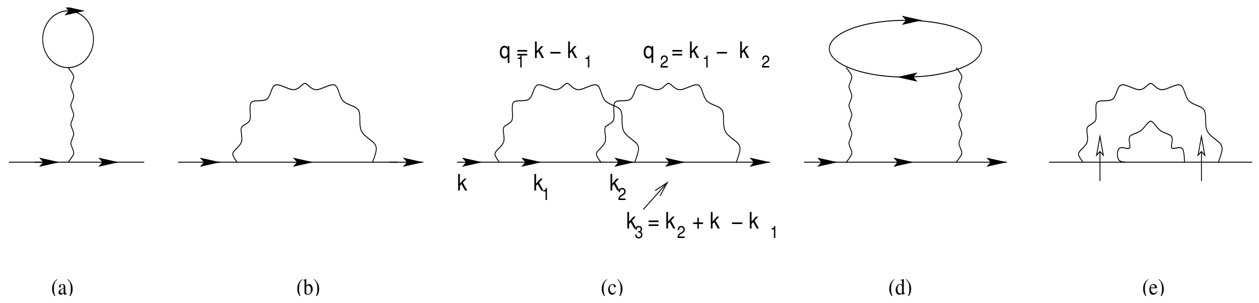
Self energy

The self energy $\Sigma(k)$, in turn, is obtained self-consistently via a Feynman graph expansion in terms of G and the interaction potential V. As illustrated in Figure 4 for orders $n = 1$ and 2, an *nth order* Σ*-graph* consists of n non-directional wavy lines (referred to as "V-lines" hereafter) and of 2 external and $2n - 1$ internal directed straight lines, (referred to as "G-lines" hereafter). One incoming and one outgoing G-line is attached to each endpoint ("vertex") of each V-line. The $\Sigma(k)$-contribution for each graph is given by the Feynman rules [11, 15] so that for $k \equiv (\vec{k}, i\,\nu)$ and $k_v \equiv (\vec{k}_v, i\,\nu_v)$ $(v = 1, 2, 3, \ldots)$,

$$\Sigma(k) = \sum_{n=1}^{n_{max}} \sum_{g \in \mathcal{G}_n} \left(\frac{-T_P}{N}\right)^n \sum_{k_1,\ldots,k_n \in \mathcal{K}} F_g^{(n)}(k, k_1, \ldots, k_n). \tag{3}$$

Here, $\mathcal{G}_n$ denotes the set of all topologically distinct "G-irreducible" Σ-graphs g of order n, where g is defined to be G-irreducible (G-reducible) if and only if it cannot (can) be severed into two disjoint pieces by cutting no more than two internal G-lines, as illustrated in Figure 4.

Figure 4: Feynman Diagrams



Shown are all G-irreducible Σ-graphs of order $n = 1$, in (a) and (b), and of order $n = 2$, in (c) and (d), as well as a selected G-reducible graph of order $n = 2$, in (e). In (c), a possible k-assignment is also shown. In (e), vertical arrows indicate the "cuts" which separate the graph into two disjoint pieces.

Summation Domains and Parameters

The $\vec{k}_v$-summation domain is the set $\mathcal{B}$ of $\vec{k}$- grid points in the *first Brillouin zone*:

$$\mathcal{B} := \{\vec{k} = (k^{(1)}, \ldots, k^{(D)}) \mid k^{(d)} = \frac{2\pi m_d}{L_d}, m_d \in \mathcal{L}_d \text{ for } d \in 1, \ldots, D\} \tag{4}$$

with

$$\mathcal{L}_d := \{-\lfloor (L_d - 1)/2 \rfloor, -\lfloor (L_d - 1)/2 \rfloor + 1, \ldots, \lfloor L_d/2 \rfloor\}$$

where $\lfloor \cdot \rfloor$ is the greatest integer function and L_d denotes the integer sidelength of the D-dimensional finite lattice prism in the d-th coordinate direction. N is the total number of sites j in the lattice, hence

$$N = \prod_{d=1}^{D} L_d.$$

The $i\,\nu$-summation domain is the set $\mathcal{M}$ of *odd Matsubara frequencies*:

$$\mathcal{M} := \{i\,\nu \mid \nu = (2m_0 + 1)\pi T_P \,,\ m_0 \text{ an integer}\} \tag{5}$$

where T_P denotes the temperature of the physical system.

The inner sums in (3) are therefore taken over $\mathcal{K} := \mathcal{B} \times \mathcal{M}$, and the summand $F_g^{(n)}(k, k_1, \ldots, k_n)$, for $n \geq 1$, contains the internal G- and V-line factors of graph g:

$$F_g^{(n)}(k, k_1, \ldots, k_n) = (-(2s_f + 1))^{l g} \exp(\delta_{n,1}\, i\,\nu\, 0^+) \prod_{u=1}^{2n-1} G(k_u) \times \prod_{x=1}^{n} V(q_x). \tag{6}$$

Here, the momenta k_u and q_x associated with the G- and V-lines, respectively, are determined by the graph's topology, via momentum conservation rules at each vertex, as illustrated in Figure 4(c). Only the first n of the internal G-line k-variables, $k_1, \ldots, k_n$ can be chosen and summed over independently; the remaining k-variables, $k_{n+1}, \ldots, k_{2n-1}$, and all q-variables, $q_1, \ldots, q_n$, are linear combinations of the external k and of $k_1, \ldots, k_n$.

Finally, $V(q)$ denotes the Fourier transform of the interaction potential V of our lattice model:

$$V(q) = V(\vec{q}) := N^{-1} \sum_{j'\,j} e^{-i\vec{q}\cdot(\vec{r}_{j'} - \vec{r}_j)} V_{j'\,j} \tag{7}$$

where $\vec{r}_j$ denotes the position vector of site j. Note that q is a momentum variable with a wavevector component in $\mathcal{B}$ and a frequency component ($q = (\vec{q}, i\,\omega)$). The frequency component is not in the set of odd Matsubara frequencies (i.e. $q \notin \mathcal{K}$), however as $V(q)$ is independent of the frequency component, we will not describe the domain here.

In equation 6, s_f is the "single-fermion spin", taking the value 1/2 for electrons and l^g is the number of closed G-loops in graph g. Remaining parameters and are described in Table 1 where all parameters and quantities addressed thus far are also summarized.

The purpose of this paper is to give a "perfect" Monte Carlo approach that will perform the **innermost** summation in (3). Our collaborators address the other summations in [20].

2.2 Computer Friendly Representation

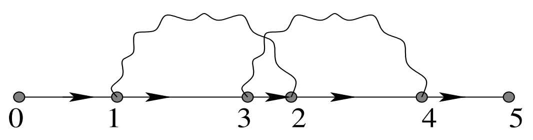
Computationally, each Feynman graph $g \in \mathcal{G}_n$ in (3) can be conveniently represented by labeling each vertex by an integer $n \in \{1, 2, \ldots, 2n\}$, in such a manner that v and $v+1$ denote endpoints of the same V-line if v is odd. Additionally, the origin of the incoming external G-line and the terminus of the outgoing external G-line are labeled $v = 0$ and $v = 2n+1$, respectively. The complete topology (connectivity) of the graph can then be specified by a "linkage list", $[w(v)]_{v=0}^{2n}$, where $w(v)$ is the integer label of the terminating vertex of the G-line that originates at vertex v. The integer v then also serves as a convenient label of all G-lines: "v" is that G-line which originates from vertex v for $v \in \{1, 2, \ldots 2n\}$, with $v = 0$ labeling the incoming external G-line. Using the notation $[w^{-1}(v)]_{v=1}^{2n+1}$ for the inverse assignment, an example for a graph of order $n = 2$ is shown in Figure (5).

The assignment of k and q variables corresponding to the internal G- and V-lines in equation (6) can be represented by a pair of "k-assignment" lists, $\sigma_G(u, v)$ and $\sigma_V(x, v)$, such that

$$k_u = \sum_{v=0}^{n} \sigma_G(u, v)\, k_v \qquad \text{and} \qquad q_x = \sum_{v=0}^{n} \sigma_V(x, v)\, k_v \tag{8}$$

for $u \in \{1, 2, \ldots, 2n-1\}$ and $x \in \{1, 2, \ldots, n\}$ with $k_0 \equiv k$. Given the graph's linkage list (and its inverse w^{-1}), the σ_G's and σ_V's are constructed to satisfy momentum conservation

Figure 5: Representing a Feynman Diagram as a Linkage List



v	w(v)	$w^{-1}(v)$
0	1	–
1	3	0
2	4	3
3	2	1
4	5	2
5	–	4

at each vertex. For the endpoints of V-line $x \in \{1, 2, \ldots, n\}$, that is, for vertices $v = 2x - 1$ and $v = 2x$, respectively,

$$k_{2x-1} + q_x = k_{w^{-1}(2x-1)} \qquad \text{and} \qquad k_{2x} = q_x + k_{w^{-1}(2x)}. \tag{9}$$

We have adopted the convention that the (usually non-directional) V-line x carries momentum q_x *from* vertex $2x - 1$ *towards* vertex $2x$ and $k_0 = k_{2n} \equiv k$. The resulting $\sigma_G(u, v)$ and $\sigma_V(x, v)$ take on values 0, +1, and −1 only, with $\sigma_G(u, v) = \delta_{u,v}$ for $u = 0, 1, \ldots, n$, as illustrated in Figure 4(c).

Preliminary steps for this labeling procedure consists of setting up tables with topologies of all irreducible diagrams. This task has been carried out using a deep first search algorithm program developed by Robert Robinson and coworkers [18] at the University of Georgia.

3 Monte Carlo Summation

We are often faced with integrals (or sums) that cannot be computed analytically. In the case of multidimensional integrals, even the simplest methods of approximation by discretization can become prohibitively expensive. Monte Carlo integration provides an alternative approximation technique.

Suppose we wish to evaluate the integral

$$I(g, A) = \int_A g(x)dx \tag{10}$$

for some integrable function $g : \mathbb{R}^n \to \mathbb{R}$ and some set $A \subseteq \mathbb{R}^n$. Let $\pi(x)$ be any probability density function whose support contains A. Then we may write

$$I(g, A) = \int_A g(x)dx = \int_{\mathbb{R}^n} g(x)\mathbb{1}_A(x)dx = \int_{\mathbb{R}^n} \frac{g(x)\mathbb{1}_A(x)}{\pi(x)}\pi(x)dx$$

where $\mathbb{1}_A(x)$ is the indicator function. We shall refer to $\pi(x)$ as the *weight function* and the remaining part of the integrand as the *score function*.

Now if X is a random variable with density $\pi(x)$, we find that we have written the integral as an expected value

$$I(g, A) = \mathsf{E}\left[\frac{g(X)\mathbb{1}_A(X)}{\pi(X)}\right]. \tag{11}$$

A basic Monte Carlo integration technique is to simulate independent and identically distributed (i.i.d.) values drawn from the distribution with density $\pi(x)$, say

$$X_1, X_2, \ldots, X_n \overset{iid}{\sim} \pi$$

and to estimate the probability weighted average given in (11) by

$$\hat{I}(g, A) = \frac{1}{n}\sum_{i=1}^{n}\frac{g(X_i)\mathbb{1}_A(X_i)}{\pi(X_i)}. \tag{12}$$

Clearly, by (11), this is an unbiased estimator of $I(g, A)$.

Since the variance of this estimator is

$$\begin{aligned} V[\hat{I}(g, A)] &= \tfrac{1}{n} V\left[\tfrac{g(X)\mathbb{1}_A(X)}{\pi(X)}\right] \\ &= \tfrac{1}{n}\mathsf{E}\left[\left(\tfrac{g(X)\mathbb{1}_A(X)}{\pi(X)} - I(g, A)\right)^2\right], \end{aligned}$$

the variance is minimized by choosing

$$\pi(x) = \frac{|g(x)|\mathbb{1}_A(x)}{\int_{\mathbb{R}^n}|g(x)|\mathbb{1}_A(x)\,dx}. \tag{13}$$

Specifically, we choose

$$\pi(x) = \begin{cases} \frac{|g(x)|}{\int_A |g(x)|\,dx} & , \quad x \in A \\ 0 & , \quad x \notin A \end{cases} \tag{14}$$

Of course, if we could compute the denominator for this optimal "weight", it is likely that we could have solved our original problem and there is no need to use a Monte Carlo approach to compute the integral in (10). Hence, we may consider taking a different, non-optimal candidate weight, keeping in mind that we would like to at least choose something that attempts to mimic the shape of g. We point out that while we may usually sample from (14) without knowing the constant of proportionality, (for example using the Metropolis-Hastings algorithm), we ultimately require the constant in order to evaluate (12).

4 Perfect Simulation

There has been considerable recent work on the development and application of "perfect sampling" algorithms that will enable the simulation of the invariant (or stationary) measure π of a Markov chain, either exactly (that is, by drawing a random sample known to be from π) or approximately, but with computable order of accuracy. These were sparked by the seminal paper of Propp and Wilson [17], and several variations and extensions of this idea have appeared since – see [3, 4, 5, 7, 8, 12], and [14]. These ideas have proven effective in areas such as statistical physics, spatial point processes and operations research, where they provide simple and powerful alternatives to methods based on iterating transition laws, for example.

The essential idea of most of these approaches is to find a random epoch $-T$ (not to be confused with the physical temperature of the system being considered in this paper) in the past such that, if we construct sample paths (according to a transition law $P(x, y)$ that is invariant for π) from every point in the state space starting at $-T$, then all paths will have coupled successfully by time zero. The common value of the paths at time zero is a draw from π. Intuitively, it is clear why this result holds with such a random time T. For consider a chain starting at $-\infty$ with the stationary distribution π. At every iteration it maintains the distribution π. But at time $-T$ it must pick *some* value x, and from then on it follows the trajectory from that value. But of course it arrives at the same place at time zero no matter what value x is picked at time $-T$: so the value returned by the algorithm at time zero must itself be a draw from π.

Perfect sampling algorithms can be particularly efficient if the chain is *stochastically monotone* in the sense that paths from lower starting points stay below paths from higher starting points. In this case, one need only couple sample paths from the "top" and "bottom" of the space, as all other paths will be sandwiched in between. It is possible to generalize one step further to monotone chains on an unbounded state space by considering *stochastically dominating* processes to bound the journeys of sample paths. For example, see [8].

For this to be practicable we need to ensure that T is indeed finite. In [17] it is shown that this occurs for irreducible aperiodic finite space chains, and for a number of stochastically monotone chains possessing maximal and minimal elements. Indeed, in [2] it is shown that if the distribution at $-\infty$ is any fixed (or even random) value x_0, then under fairly standard conditions the value at time zero is still distributed according to π, and this observation is crucial for perfect sampling algorithms.

There are several easy-to-read perfect sampling tutorials available, and we refer interested readers to [1]. In this paper we only wish to emphasize that the key idea in the search successively further and further back in time for the so-called *backward coupling time* T requires that one reuse random number streams. That is, if sample paths run forward to time 0 from time -1 using a random number (or random vector) U_{-1} have not coalesced by time 0, then one must go back further, say to time -2 and run paths forward for two steps using a random number U_{-2} and then the **previously used** U_{-1}.

We now describe a particular specific perfect sampling scheme that we will use for our self energy calculations.

4.1 The IMH Algorithm

Certain monotonicity properties of the "independent" Metropolis-Hastings (IMH) scheme, which we review briefly in this section, are such that perfect sampling is feasible [2]. We use the term "independent" to describe the Metropolis-Hastings algorithm where candidate states are generated by a distribution that is independent of the current state of the chain. In other words, suppose we have a given candidate distribution Q which we will assume to have a density q, positive everywhere for convenience, with which we can generate potential values of an i.i.d. sequence. A "candidate value" generated according to q is then accepted with probability $\alpha(x,y)$ given by

$$\alpha(x,y) = \begin{cases} \min\left\{\frac{\pi(y)}{\pi(x)}\frac{q(x)}{q(y)},\ 1\right\} & \pi(x)q(y) > 0 \\ 1 & \pi(x)q(y) = 0 \end{cases}$$

where x is the current state and y is the candidate state.

Thus, actual transitions of the IMH chain take place according to a law P with transition density

$$p(x,y) = q(y)\alpha(x,y), \qquad y \neq x$$

and with probability of remaining at the same point given by

$$P(x,\{x\}) = \int q(y)[1-\alpha(x,y)]\mu(dy)$$

where μ is Lebesgue measure. With this choice of α we have that π is invariant for P.

The perfect IMH algorithm uses the ratios in the acceptance probabilities $\alpha(x,y)$ to reorder the states in such a way that we always accept moves to the left (or downwards). That is, if we write $\pi(x) = c\,h(x)$ where k is unknown, we define the IMH ordering,

$$x \succeq y \qquad \Leftrightarrow \qquad \frac{\pi(y)q(x)}{\pi(x)q(y)} \geq 1 \qquad \Leftrightarrow \qquad \frac{h(y)}{q(y)} \geq \frac{h(x)}{q(x)} \tag{15}$$

With this ordering, we can (hopefully) attain a "lowest state" l. Essentially, one can think of l as the state that is hardest to move away from when running the IMH algorithm. Thus, if we are able to accept a move from l to a candidate state y drawn from the distribution Q with density q, then sample paths from every point in the state space will also accept a move to y, so all possible sample paths will coalesce. The perfect sampling algorithm is formally described as follows:

IMH (Backward Coupling) Algorithm

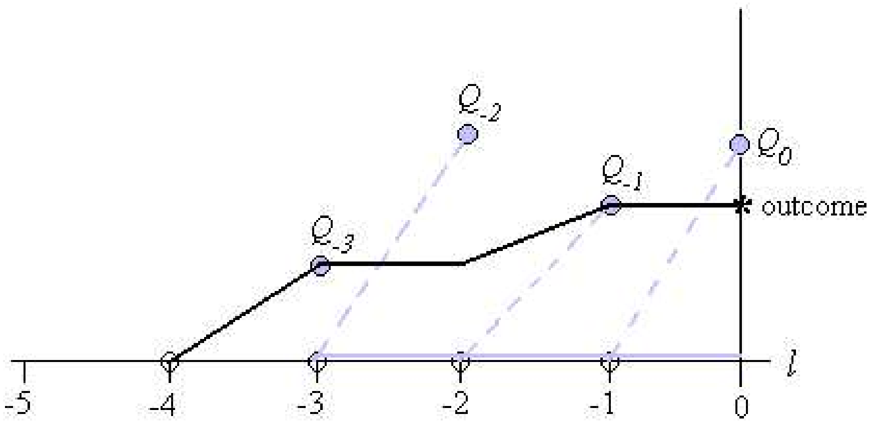
1. Draw a sequence of random variables $Q_n \sim Q$ for $n = 0, -1, -2, \ldots$, and a sequence $\alpha_n \sim \text{unif}(0,1)$ for $n = -1, -2, \ldots$.

2. For each time $-n = -1, -2, \ldots$, start a lower path L at l, and an upper path, U at Q_{-n}.

3. (a) For the lower path: Accept a move from l to Q_{-n+1} at time $-n+1$ with probability $\alpha(l, Q_{-n+1})$, otherwise remain at state l. That is, accept the move from l to Q_{-n+1} if $\alpha_{-n} \leq \alpha(l, Q_{-n+1})$.
   (b) For the upper path: Similarly, accept a move from Q_{-n} to Q_{-n+1} at time $-n+1$ if $\alpha_{-n} \leq \alpha(Q_{-n}, Q_{-n+1})$; otherwise remain at state Q_{-n}.
4. Continue until T defined as the first n such that at time $-n+1$ each of these two paths accepts Q_{-n+1}. (Continue the Metropolis-Hastings algorithm foward to time zero using the α's and Q's from steps 1-3 to get the draw from π at time zero.)

By monotonicity, the upper path will accept a candidate point whenever the lower path will, and the two paths will be the same from that time forward. Consequently, our description of the upper process is a formality only, and, indeed, the upper process need not be run at all. Figure 6 illustrates a realization of the perfect IMH algorithm.

Figure 6: A Realization of the Perfect IMH Algorithm With Backward Coupling Time at -4



Dashed grey lines represent potential but unrealized arcs of the sample path. Solid grey lines represent the sample paths started at times -1, -2, and -3 that did not achieve the coupling. The solid black line represents the path whose outcome is ultimately observed in the perfect sampling algorithm.

For sampling from complicated densities, we may wish to take advantage of the observation that neither the lowest state, l, nor the maximum value $\pi(l)/q(l)$ need be attained explicitly. If we are able to find a constant C such that

$$C \geq \frac{\pi(x)}{q(x)}, \qquad \text{for all } x$$

then we know that

$$\alpha(l,y) \geq \frac{\pi(y)}{C\,q(y)}$$

so we could modify step 3(a) of the IMH algorithm to read

3. (a)$'$ For the lower path: Accept a move from l to Q_{-n+1} at time $-n+1$ with probability $\alpha(l, Q_{-n+1})$, otherwise remain at state l. That is, accept the move from l to Q_{-n+1} if $\alpha_{-n} \leq \frac{\pi(y)}{C\,q(y)}$.

We refer the reader to [2] for details on how one's choice of Q will affect the expected backward coupling time for the perfect IMH algorithm.

5 Perfect IMH in the Context of this Self Energy Problem

Earlier work on the Monte Carlo evaluation of individual Feynman diagrams has employed a variety of non-perfect sampling techniques [9, 10, 19]. We now describe the perfect IMH algorithm for computing the innermost sum in (3) for the second order diagram depicted in Figures 4(c) and 5. Ignoring the physical constants, we would like to compute

$$\Sigma_2(k) := \sum_{k_1,k_2\in\mathcal{K}} \prod_{u=1}^{3} G(k_u) \;\times\; \prod_{x=1}^{2} V(q_x). \tag{16}$$

Using momentum conserving assignments, described in Section 2.2 and shown in Figure 4(c), this becomes

$$\Sigma_2(k) = \sum_{k_1,k_2\in\mathcal{K}} G(k_1)\cdot G(k_2)\cdot G(k_2+k-k_1)\cdot V(k-k_1)\cdot V(k_1-k_2). \tag{17}$$

Recall that each k, k_1, k_2 is a $D+1$-dimensional with the vector of the first D components coming from $\mathcal{B}$ as decribed in (4) and the third component taking values in

$$\mathcal{M} := \{i\,\nu\,|\,\nu = (2m_0+1)\pi T_P\,,\ m_0 \text{ an integer}\}.$$

As a computational simplification, we will truncate $\mathcal{M}$ to a finite space $\mathcal{M}_T$. (We address alternatives in Section 5.4.) This will allow us to choose a trivial candidate distribution for the perfect IMH algorithm where, more importantly, we can easily maximize the h/q ratio described in (15) in order to identify the "lowest state" needed to carry out the algorithm. For physical reasons, it is necessary to truncate $\mathcal{M}$ in a symmetric way, say

$$\mathcal{M}_T = \{i\,\nu\,|\,\nu = (2m_0+1)\pi T_P\,,\ m_0 \in \{-l, -l+1, \ldots, l-1\}\}$$

for some positive integer l.

The astute reader may have noticed by now that we are trying to compute $\Sigma_2(k)$ which is a term of $\Sigma(k)$, given by (3), which is itself involved in the Green's function (2), which is in our expression for $\Sigma_2(k)$. This is handled with standard feedback methods described in [20], [16], and [13] where one

1. starts by setting $\Sigma(k) \equiv 0$ in (2) so that $G(k)$ becomes
$$G(k) = G(\vec{k}, i\,\nu) = [i\,\nu - \epsilon(\vec{k})]^{-1}, \tag{18}$$
2. carries out the entire Monte Carlo procedure to compute (3) which consists of the procedures of this paper for the "inner (single-diagram) sum" and the procedures of [20] for the "outer sums" which move through diagram orders and diagrams within an order,
3. uses the resulting $\Sigma(k)$ to update the Green's function before returning to step 2. and repeating the entire combined Monte Carlo procedure.

This approach is repeated until consecutive updates of the Green's function coincide within a given limit of accuracy. As this paper is concerned with computing the self energy contribution for single diagrams only, we will use a fixed Green's function. For simplicity, we use (18).

5.1 The Monte Carlo Sum

For the Monte Carlo summation, we propose as a weight function the product of the magnitudes of the Green's functions:
$$|G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k - k_1)|. \tag{19}$$
However, since this involves the input external momentum k, it would be inefficient to focus exclusively on this weight. In order to have a single weight function for any input momentum, we again rewrite our summation goal:
$$\Sigma_2(k) = \sum_{k_1,k_2,k_0 \in \mathcal{K}} V(k - k_1) \cdot V(k_1 - k_2) \cdot \delta_{k,k_0} G(k_1) \cdot G(k_2) \cdot G(k_2 + k_0 - k_1) \tag{20}$$
where δ_{k,k_0} is equal to 1 when $k_0 = k$ and equal to 0 otherwise.

Now we may write
$$\Sigma_2(k) = \sum_{k_1,k_2,k_0 \in \mathcal{K}} S(k, k_0, k_1, k_2)\, W(k_0, k_1, k_2) \tag{21}$$
where $W(k_0, k_1, k_2)$ is the (unnormalized) weight function
$$W(k_0, k_1, k_2) = |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)| \tag{22}$$
and $S(k, k_0, k_1, k_2)$ is the score function
$$S(k, k_0, k_1, k_2) = \delta_{k,k_0}\, V(k - k_1) \cdot V(k_1 - k_2) \frac{G(k_1) \cdot G(k_2) \cdot G(k_2 + k_0 - k_1)}{|G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|}. \tag{23}$$
Our "target distribution" from which to draw values for the Monte Carlo summation described in Section 3 is
$$\pi(k_0, k_1, k_2) = \frac{1}{N_W} W(k_0, k_1, k_2) = \frac{1}{N_W} |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)| \tag{24}$$

where

$$N_W := \sum_{k_0,k_1,k_2 \in \mathcal{B}\times\mathcal{M}_T} |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|. \tag{25}$$

Following Section 3 we write

$$\begin{aligned} \Sigma_2(k) &= N_W \textstyle\sum_{k_0,k_1,k_2 \in \mathcal{B}\times\mathcal{M}_T} \; S(k,k_0,k_1,k_2)\,\pi(k_0,k_1,k_2) \\ &= N_W \; \mathsf{E}\,[S(K_0,K_1,K_2)] \end{aligned}$$

where (K_0, K_1, K_2) is a random vector with density $\pi(k_0, k_1, k_2)$.

5.2 The IMH Algorithm

We will now describe how to use the perfect IMH algorithm to draw n values, (k_{0i}, k_{1i}, k_{2i}), from $\pi(k_0, k_1, k_2)$. We will use these values both to estimate N_W and to plug them into the Monte Carlo approximation

$$\hat{\Sigma}_2(k) := N_W \frac{1}{n} \sum_{i=1}^{n} S(k, k_{0i}, k_{1i}, k_{2i}). \tag{26}$$

(Note: Due to the large number of symbols in use, we are reclaiming the use of the letter n throughout Section 5 where we are considering a fixed diagram of order 2. In previous sections, n was reserved for a diagram order.)

In the notation of Section 4.1, the unnormalized target density is

$$h(k_0, k_1, k_2) \equiv W(k_0, k_1, k_2)$$

which is given by (22).

We choose a simple uniform candidate distribution with density $q(k_0, k_1, k_2)$ which gives equal weight to all points $(k_0, k_1, k_2) \in (\mathcal{B} \times \mathcal{M}_T)^3$.

In order to implement the IMH algorithm, it remains to identify the "lowest point", (k_0^*, k_1^*, k_2^*), which is the point in $(\mathcal{B} \times \mathcal{M}_T)^3$ that maximizes the ratio h/q. Since q is constant for all $(k_0, k_1, k_2) \in (\mathcal{B} \times \mathcal{M}_T)^3$, we simply want to maximize

$$h(k_0, k_1, k_2) = |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|. \tag{27}$$

Consider first a single factor $|G(k)|$. Since

$$\begin{aligned} |G(k)| &= |G(k^{(1)}, \ldots, k^{(D)}, i\nu)| = |i\,\nu + \mu + 2t \textstyle\sum_{d=1}^{D} cos(k^{(d)})|^{-1} \\ &= \left[\nu^2 + \left(\mu + 2t \sum_{d=1}^{D} cos(k^{(d)}) \right)^2 \right]^{-1/2}, \end{aligned}$$

we can maximize $|G(k)|$ by minimizing

$$\nu^2 + \left(\mu + 2t\sum_{d=1}^{D} cos(k^{(d)})\right)^2 .$$

Regardless of where we truncate $\mathcal{M}$, ν^2 is minimized for $\nu = \pm\pi T_P$. The IMH algorithm does not require a unique "lowest point", so we may take either value for ν.

We minimize

$$\left(\mu + 2t\sum_{d=1}^{D} cos(k^{(d)})\right)^2$$

with brute force by searching over the (finite) space $\mathcal{B}$. It turns out that we only need to do this once for the extire self energy calculation. This does not depend on the particular graph we are considering at any moment and also therefore, just as importantly, does not depend on the number of k's associated with a graph.

At this point we have identified a $k^* = (\vec{k}^*, i\,\pi T_P)$ that maximzes $|G(k)|$. We may simultaneously maximize all factors in (27) by setting each of k_0, k_1, and k_2 to be k^*. Due to the relationships given by (9), we will always be able to simultaneously maximize all factors in the weight function associated with any Feynman graph in this way.

As $\Sigma_2(k)$ and ultimately $\Sigma(k)$ is complex-valued, the denominator of the updated, self-consistent Green's function obtained through the feedback procedure described in Section 5 will have an imaginary component that is not simply in the set $\mathcal{M}_T$. In this case, we will have to maximize $|G(k)|$ by searching over values for $\vec{k}$ and ν together. This does *not* require searching over the entire space $(\mathcal{B} \times \mathcal{M}_T)$ but rather it can be done with some finite upper-limit cut-off on $|i\,\nu|$ since for very large $i\,\nu$, $\Sigma(k)$ is bounded by a constant independent of $i\,\nu$. This fact will be especially useful when we want to use the entire set $\mathcal{M}$ (described in Section 5.4) as opposed to the truncated set $\mathcal{M}_T$. The foregoing procedure (of choosing W and then maximizing W by individually maximizing each contributing $|G(k_i)|$-factor) can be straightfowardly generalized to higher order diagrams. The method is thus applicable at all diagram orders with arbitrarily updated input Green's function $G(k)$.

5.3 Estimating N_W

In order to estimate the normalization constant N_W, we take advantage of computations already performed within the IMH algorithm. In executing the algorithm, we are drawing values uniformly from the space $(\mathcal{B} \times \mathcal{M}_T)^3$. Hence, letting $M := \#\left((\mathcal{B} \times \mathcal{M}_T)^3\right)$ denote the number of points in the space, we write N_W as

$$\begin{aligned} N_W &= \textstyle\sum_{(\mathcal{B}\times\mathcal{M}_T)^3} h(k_0, k_1, k_2) \\ &= \textstyle\sum_{(\mathcal{B}\times\mathcal{M}_T)^3} \frac{h(k_0,k_1,k_2)}{1/M}\frac{1}{M} \\ &= \mathsf{E}\left[\frac{h(K_0,K_1,K_2)}{1/M}\right] = M \cdot \mathsf{E}\left[\frac{h(K_0,K_1,K_2)}{1/M}\right] \end{aligned}$$

where (K_0, K_1, K_2) is a random vector with uniform density on $(\mathcal{B} \times \mathcal{M}_T)^3$.

We estimate this expectation by drawing n points uniformly over $(\mathcal{B} \times \mathcal{M}_T)^3$ and computing

$$\hat{N}_W := \frac{M}{n} \sum_{i=1}^{n} h(k_{0i}, k_{1i}, k_{2i})$$

where (k_{0i}, k_{1i}, k_{2i}) are the uniform draws.

5.4 Extending the $\mathcal{M}$- Space

In order to execute the perfect IMH algorithm, we need to be able to identify the point that maximizes h/q where h is the (possibly unnormalized) target density and q is a candidate density with the same support as h from which we are able to simulate values. The truncation of $\mathcal{M}$ is a common practice, irrespective of the algorithm to be used, that simplifies computational cost. This turned out to be very convenient for us as well since it provided us with a finite state space that allowed us to choose a very simple uniform candidate density which exchanged our task of maximizing h/q for the simpler task of maximizing only h. Another nice feature of this model is that we are able to make $\mathcal{M}_T$ arbitrarily large (but finite) without any adjustment, as h is always maximized at $\nu = \pm i\,\pi T_P$.

It is possible that one can use the full space $\mathcal{M}$ provided one chooses a candidate density q on $\mathcal{B} \times \mathcal{M}$ for which the maximizing point for h/q can be identified. If this is not the case, one alternative is to approximate the maximum either with traditional deterministic optimizers or a stochastic search approach. We are currently investigating the effect of this approximation on the perfect IMH algorithm and preliminary results suggest this "imperfect perfect simulation algorithm" is still superior to traditional Monte Carlo methods in terms of both accuracy and speed.

6 Simulation Results

For each of the following two examples, we use the specific second order diagram of Section 5. To verify our algorithm, we start with a very low dimensional toy example where our results can be compared to those from brute force calculations.

For both examples we take $\mu = 0.5$, $t = 1.0$, and the physical temperature to be $T_P = 2.0$.

Example 1:

We take $D = 2$, $L_1 = L_2 = 2$, and only two odd Matsubara frequencies. This means that all of the k's in (17) have 3 components with the first two taking values in

$$\mathcal{A} := \{0, \pi\}\,.$$

and the third taking values in

$$\mathcal{M}_T := \{-i\,\pi T_P, i\pi, T_P\}.$$

(Note that $\mathcal{B} = \mathcal{A} \times \mathcal{A}$.)

We simulated 100,000 (9-dimensional) values from

$$\pi(k_0, k_1, k_2) \propto h(k_0, k_1, k_2) = |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|.$$

The results were as follows.

1. Exact N_W Versus the Estimate

   By brute force summation, we find that the exact value of the normalizing constant N_W is 1.349741. Since there are $(2 \cdot 2 \cdot 2)^3 = 512$ possible values for (k_0, k_1, k_2), we write

   $$\begin{aligned} N_W &= \textstyle\sum_{(k_0,k_1,k_2)} h(k_0, k_1, k_2) \\ &= 512 \cdot \textstyle\sum_{(k_0,k_1,k_2)} h(k_0, k_1, k_2) \cdot \frac{1}{512} \\ &= 512 \cdot \mathsf{E}[h(K_0, K_1, K_2)] \end{aligned}$$

   where (K_0, K_1, K_2) is a random vector distributed uniformly over $(\mathcal{B} \times \mathcal{M}_T)^3$.

   Our estimate is then

   $$\hat{N}_W = 512 \cdot \frac{1}{m} \sum_{i=1}^{m} h(k_{0i}, k_{1i}, k_{2i})$$

   where (k_{0i}, k_{1i}, k_{2i}) is a particular value generated from the uniform candidate distribution in our simulation and m is the number of candidates generated. (To generate 100,000 outomes we generate more than 100,000 candidates.)

   Our simulation produced the estimate

   $$\hat{N}_W = 1.349738.$$

2. Comparing Probability Estimates for Various Regions

   Using the true value of N_W, we computed the 512 probabilites associated with the 512 points in $(\mathcal{B} \times \mathcal{M}_T)^3$. We can then compute, for example

   $$P(k^1_{(1)} = 0) = \frac{1}{N_W} \sum_{(k_0,k_1,k_2),k_1^{(1)}=0} h(k_0, k_1, k_2) = 0.4963337.$$

   In Table 2, we compare the true probabilities of seeing value in a given region of points with the our estimated probabilities. Although we are sampling "perfectly" from the target distribution, we are estimating probabilities based on a finite number of sampled values. All estimates consistently remain well within two standard errors of the target values.

3. Backward Coupling Times

   In 100,000 draws, the average backward coupling time (the number of time steps required to achieve a draw from the target distribution) was 1.2 time steps. The minimum and maximum backward coupling times were 1 and 8, respectively. Drawing 100,000 values from $\pi(k_0, k_1, k_2)$ took less than a second on a 450 MHz Sun SPARC processor.

4. $\Sigma_2(k)$

   It remains only to evaluate the score function

$$S(k, k_0, k_1, k_2) = \delta_{k,k_0} V(k - k_1) \cdot V(k_1 - k_2) \frac{G(k_1) \cdot G(k_2) \cdot G(k_2 + k_0 - k_1)}{|G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|}.$$

   at the values we have drawn from $\pi(k_0, k_1, k_2)$ and a fixed external input k in order to produce the estimate for $\Sigma_2(k)$ given by (26).

   As a simple model potential, we consider

$$V_{j'j} = \begin{cases} 4t & , \text{ for } j = j' \\ t & , \text{ for } j \text{ and } j' \text{ nearest neighbors} \\ t/\sqrt{2} & , \text{ for } j \text{ and } j' \text{ second nearest neighbors} \\ 0 & , \text{ otherwise} \end{cases}.$$

   Here, j and j' label the sites of the 4 points in

$$\mathcal{B} = \mathcal{L}_1 \times \mathcal{L}_2$$

   and "nearest neighbors" of a site are the four nearest neighbors with periodic boundary conditions. Recall that our score function contains $V(\vec{k})$ factors where $V(\vec{k})$ denotes the Fourier transform of the interaction potential:

$$V(\vec{k}) := N^{-1} \sum_{j'j} e^{-i\vec{k}\cdot(\vec{r}_{j'} - \vec{r}_j)} V_{j'j} \tag{28}$$

   where $\vec{r}_j$ denotes the position vector of site j and N is the total number of points in $\mathcal{B}$.

   In Table 3 we compare the true value of $\Sigma_2(0, 0, i\pi T_P)$ with simulated values based on various sample sizes. In Table 4 we compare the true value of $\Sigma_2(k)$ for each $k \in \mathcal{B} \times \mathcal{M}$ with simulated values based on a sample of size 100,000. It should be pointed out that, for this example, the δ_{k,k_0} term in the score function allowed only approximately 12.5% of the simulated values drawn from $\pi(k_0, k_1, k_2)$ to contribute to our estimate for $\Sigma_2(k)$.

Example 2:

We take $D = 2$, $L_1 = L_2 = 64$, and 50 odd Matsubara frequencies. Thus, our 9-dimensional state space has approximately 8.6×10^{15} points, making a brute force approach rather unappealing.

Drawing 100,000 values from $\pi(k_0, k_1, k_2)$ took approximately 12 seconds on a 450 MZ Sun Sparc processor.

Our simulation resulted in the estimate

$$\hat{N}_W = 3.483866904037023 \times 10^{10}.$$

It would be of little value in this case to attempt an estimate of $\Sigma_2(0,0,i\,\pi T)$ as the δ_{k,k_3} factor would take the value 1 with probability

$$\sum_{k_1,k_2} \pi((0,0,i\,\pi T_P),k_1,k_2) \approx \frac{1}{\hat{N}_W} \sum_{k_1,k_2} h((0,0,i\,\pi T_P),k_1,k_2).$$

A very rough upper bound on this is given by

$$\begin{aligned} \tfrac{1}{\hat{N}_W}\,(64\cdot 64\cdot 50)^2\, \max_{k_1,k_2} h((0,0,i\,\pi T_P),k_1,k_2) \quad &\leq \quad \tfrac{1}{\hat{N}_W}\,(64\cdot 64\cdot 50)^2\,(\max_k |G(k)|)^3 \\ &\approx \quad \tfrac{1}{\hat{N}_W}\,(64\cdot 64\cdot 50)^2\, 0.1591549^3 \approx 0.004854. \end{aligned}$$

Here, $\max_k |G(k)|$ was obtained numerically.

However, Monte Carlo simulations give the more realistic estimate

$$\frac{1}{\hat{N}_W}\ 645603.228730 \approx 0.0000186$$

In other words, almost none of the values drawn from $\pi(k_0,k_1,k_2)$ contribute to the sum. Clearly another approach is needed here, and we provide two alternatives that address this "low-score problem" by removing the δ_{k,k_0} factor. In Section 7.1 we use conditional averaging to change the score function while still using the current weight. In Section 7.2 we change the weight and at the same time reduce the dimensionality of the problem using the modified perfect IMH algorithm given by step 3(a)$'$ in Section 4.1.

7 Addressing the Low-Score Problem

7.1 Conditional Averaging

In this section, we change the score function (23) in order to remove the δ_{k,k_0} "spike" while retaining the original weight function.

Recall from (21) that

$$\begin{aligned} \Sigma_2(k) \quad &= \quad \textstyle\sum_{k_0,k_1,k_2\in\mathcal{K}} S(k,k_0,k_1,k_2) W(k_0,k_1,k_3) \\ &= \quad N_W \textstyle\sum_{k_0,k_1,k_2\in\mathcal{K}} S(k,k_0,k_1,k_2) \pi(k_0,k_1,k_3) \end{aligned}$$

where

$$N_W = \sum_{k_0,k_1,k_2} W(k_0,k_1,k_2).$$

Using the conditional density

$$\pi(k_0|k_1,k_2) := \frac{\pi(k_0,k_1,k_2)}{\pi(k_1,k_2)}$$

where $\pi(k_1,k_2)$ is the marginal density

$$\pi(k_1,k_2) := \sum_{k_0} \pi(k_0,k_1,k_2), \tag{29}$$

we may write

$$\begin{aligned} \Sigma_2(k) &= N_W \textstyle\sum_{k_0,k_1,k_2} S(k,k_0,k_1,k_2)\pi(k_0|k_1,k_2)\pi(k_1,k_2) \\ &= N_W \textstyle\sum_{k_1,k_2} \left[\sum_{k_0} S(k,k_0,k_1,k_2)\pi(k_0|k_1,k_2)\right] \pi(k_1,k_2) \\ &= N_W \textstyle\sum_{k_1,k_2} S'(k,k_1,k_2)\pi(k_1,k_2) \end{aligned} \tag{30}$$

where

$$S'(k,k_1,k_2) := \sum_{k_0} S(k,k_0,k_1,k_2)\pi(k_0|k_1,k_2).$$

Since we are already drawing values from $\pi(k_0,k_1,k_2)$, we take advantage of the relationship given by (29) to write (30) as

$$\Sigma_2(k) = N_W \sum_{k_0,k_1,k_2} S'(k,k_1,k_2)\pi(k_0,k_1,k_2). \tag{31}$$

Note that we have removed k_0 from the score function and, in particular, we have removed the troublesome δ_{k,k_0} factor.

Replacing k by k_0 in (23), the original score function has the form

$$S(k,k_0,k_1,k_2) = \delta_{k,k_0}\, R(k_0,k_1,k_2).$$

Thus, the conditionally averaged score function $S'(k,k_1,k_2)$ may be written as

$$\begin{aligned} S'(k,k_1,k_2) &= R(k,k_1,k_2)\tfrac{\pi(k,k_1,k_2)}{\pi(k_1,k_2)} \\ &= R(k,k_1,k_2)\tfrac{W(k,k_1,k_2)}{W(k_1,k_2)} \end{aligned} \tag{32}$$

where

$$W(k_1,k_2) := \sum_{k'} W(k',k_1,k_2). \tag{33}$$

By implementing this conditional averaging approach, we have eliminated the δ_{k,k_0} factor which enables us to make use of all values drawn from $\pi(k_0,k_1,k_2)$ while simultaneously reducing the variance of the Monte Carlo estimate for $\Sigma_2(k)$. The evaluation of $W(k_1,k_2)$ requires a CPU effort on the order of $N \times |\mathcal{M}_T|$ steps, but needs to be performed only once to accumulate the scores for all external k-points, per given (k_1,k_2). Hence, the evaluation of $W(k_1, k_2)$ will, at most, double the amount of CPU effort for score accumulation in a full simulation where $\Sigma(k)$ must be evaluated for all $N \times |\mathcal{M}_T|$ external k-points. For large lattice sizes and/or higher order diagrams, the CPU effort for perfect sample generation will likely exceed the effort for score accumulation.

Example 1 Revisited:

Once again, we take $D = 2$, $L_1 = L_2 = 2$ and only two odd Matsubara frequencies. The parameters are $\mu = 0.5$, $t = 1.0$, and the physical temperature is $T_P = 2.0$.

From (31) we have that

$$\Sigma(k) = N_W\, \mathsf{E}[S'^{(k,K_1,K_2)}].$$

We use the previous estimate

$$\hat{N}_W = 1.349738$$

and average values of $S'(k, k_1, k_2)$ at values for k_1 and k_2 from draws (k_1, k_2, k_3) from $\pi(k_1, k_2, k_3)$ for various sample sizes. Results for 6 different simulation runs are shown in Table 5.

Example 2 Revisited:

Again, we take $D = 2$, $L_1 = L_2 = 64$ and 50 odd Matsubara frequencies. The parameters are $\mu = 0.5$, $t = 1.0$, and the physical temperature is $T_P = 2.0$.

Using the conditionally averaged score function, we arrive at the estimate $\hat{\Sigma}_2(0, 0, i\,\pi T_P) \approx -285.6363 - 2868.7525\,i$, which, when multipled by the constants we have ignored to this point, contributes

$$\left(\frac{2.0}{64 \cdot 64}\right)^2 [-285.6363 - 2868.7525\,i] \approx -0.000068 - 0.000684\,i$$

to the self energy $\Sigma(k)$ given by (3).

7.2 Using IMH Step 3(a)$'$

In Section 6 we used the basic perfect IMH algorithm to draw values from the weight function given by (22) as opposed to the somewhat more natural weight function given by (19):

$$W(k, k_1, k_2) = |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k - k_1)|$$

where k is the external user-input momentum vector that is fixed for any given $\Sigma_2(k)$ calculation. The purpose of using (22) was so that it would be unnecessary to change the sampling procedure each time we wanted to change the input k. However, there are two obvious drawbacks to this approach:

1. We increase the size of our state space. In this example, we go from having to sample values for (k_1, k_2) to having to sample values for (k_0, k_1, k_2).

2. Even with an efficient approach to sampling from the weight function, the δ_{k,k_0} in the score function causes us to effectively discard most of the sampled values. In Section 6, the score function turned out to be zero approximately 87.5% of the time in Example 1 and almost 100% of the time in Example 2.

In this section, we consider sampling directly from (19) in order to avoid the two drawbacks listed above. We can actually write down a single procedure that is efficient in the sense that will work for any k if we can accept the trade-off that it may become inefficient in some cases due to increased backward coupling times. (In some cases, however they may decrease since we will be making the state space smaller.)

For any fixed k, we write $\Sigma_2(k)$ as

$$\begin{aligned}
\Sigma_2(k) &= \textstyle\sum_{k_1,k_2\in\mathcal{K}} V(k-k_1)\cdot V(k_1-k_2)\cdot G(k_1)\cdot G(k_2)\cdot G(k_2+k-k_1) \\
&= \textstyle\sum_{k_1,k_2\in\mathcal{K}} V(k-k_1)\cdot V(k_1-k_2)\frac{G(k_1)\cdot G(k_2)\cdot G(k_2+k-k_1)}{|G(k_1)|\cdot|G(k_2)|\cdot|G(k_2+k-k_1)|} \\
&= S(k,k_1,k_2)\, W_k(k_1,k_2)
\end{aligned}$$

where $S(k,k_1,k_2)$ is the score function

$$S(k,k_1,k_2) := \frac{V(k-k_1)\cdot V(k_1-k_2)\cdot G(k_1)\cdot G(k_2)\cdot G(k_2+k-k_1)}{|G(k_1)|\cdot|G(k_2)|\cdot|G(k_2+k-k_1)|}$$

and $W_k(k_1,k_2)$ is the weight function

$$W_k(k_1,k_2) := |G(k_1)|\cdot|G(k_2)|\cdot|G(k_2+k-k_1)|.$$

We wish to use the perfect IMH algorithm to draw values from

$$\pi(k_1,k_2)\;\propto\; h(k_1,k_2)\equiv W_k(k_1,k_2).$$

We again choose a simple uniform candidate distribution this time with density $q(k_1,k_2)$ which gives equal weight to all points $(k_1,k_2)\in(\mathcal{B}\times\mathcal{M}_T)^2$.

In order to implement the IMH algorithm, it remains to identify the "lowest point", (k_1^*, k_2^*), which is the point in $(\mathcal{B}\times\mathcal{M}_T)^2$ that maximizes the ratio h/q. Since q is constant for all $(k_1,k_2)\in(\mathcal{B}\times\mathcal{M}_T)^2$, we simply want to maximize

$$h(k_1,k_2) = |G(k_1)|\cdot|G(k_2)|\cdot|G(k_2+k-k_1)|. \tag{34}$$

As in Section 5.2 we can, after a limited search, identify a point $k^* = (\vec{k}^*, i\,\pi T)$ (where $\vec{k}^*$ is two dimensional) that maximizes $|G(k)|$. This time however, it is more difficult to simultaneously maximize all factors in (34) as we will not necessarily be able to simply make all three arguments equal to k^* at the same time. It is not desirable to search through all k_1 and k_2 for a fixed k in order to maximize (34) for three reasons:

1. The search would need to be performed again each time we change the external k.
2. The search would need to be performed again for every graph included in the overall summation defining $\Sigma(k)$.
3. The search space will increase in size with higher order graphs and, with higher dimensions, (i.e. large D, $L_1, \ldots, L_D$), it will increase much more quickly than for the limited search described in Section 5.2.

Given that we do not want to maximize (34), we appeal to Step 3(a)$'$ (Section 4.1) of the IMH algorithm.

Recall that in the IMH algorithm we move backwards in time through time steps $-1, -2, -3, \ldots$. At each time step, we start a sample path at the "low point", l, and, looking forward in time, decide whether or not the sample path is able to accept a move to the next candidate point drawn from the candidate density q. If it will not accept the move, we go back one more time step and try again. If it will accept the move, we have achieved a coupling and, after following this sample path forward to time zero, we have a draw from π.

Suppose that we are starting a sample path at the low point l and have drawn a candidate value x from the distribution with density q. We accept a move from l to x with probability $\pi(x)/\pi(l) = h(x)/h(l)$. That is, we simulate a value u from the Uniform[0,1] distribution and accept the move if

$$u \leq \frac{h(x)}{h(l)}.$$

Suppose now that we know a value C such that $h(l) \leq C$. Since

$$\frac{h(x)}{C} \leq \frac{h(x)}{h(l)}$$

we may choose to accept the move from l to x whenever

$$u \leq \frac{h(x)}{C}$$

since this would imply that

$$u \leq \frac{h(x)}{h(l)}.$$

In this case, we will accept moves away from l less often which result in a higher backward coupling time, but we may still perform the IMH algorithm despite not being able to identify l provided we can identify a C.

In the context of this problem, we have that

$$\max_{k_1,k_2} |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k - k_1)| \leq |G(k^*)| \cdot |G(k^*)| \cdot |G(k^*)|,$$

so we take C to be $|G(k^*)| \cdot |G(k^*)| \cdot |G(k^*)|$.

Example 1 Revisited:

Once again, we take $D = 2$, $L_1 = L_2 = 2$ and only two odd Matsubara frequencies. The parameters are $\mu = 0.5$, $t = 1.0$, and $T_P = 2.0$. We report estimates for $\Sigma_2(\pi, 0, i\pi T)$ in Table 6. As expected, we have improved accuracy for small sample sizes since we are able to make use of all values drawn from the weight when evaluating the score function. The average backward coupling time for 100,000 draws was still 1.2 time steps in this case with a minimum of 1 and a maximum of 7.

Example 2 Revisited: We again consider the case where $D = 2$, $L_1 = L_2 = 64$ with 50 odd Matsubara frequencies. Using the modified IMH algorithm described at the beginning of this Section, we arrive at the estimate $\hat{\Sigma}_2(0, 0, i\,\pi T)_P \approx -286.0717 - 2876.0375\,i$ which, when multiplied by the constants we have ignored to this point, contributes

$$\left(\frac{2.0}{64 \cdot 64}\right)^2 [-286.0717 - 2876.0375\,i] \approx 0.0000068 - 0.000686$$

to the self energy $\Sigma(k)$ given by (3).

It is interesting to note that even though the state space was of size $(64 \cdot 64 \cdot 50)^2 = 41,943,040,000$, roughly one-third of our 100,000 draws from the weight function, required that we only go back in time 1 step to acheive the coupling. In 50% of the draws, we had to go back 3 steps or less and 75% of the time the backward coupling time was less than or equal to 9. The average backward coupling time in 100,000 draws was 20.5. This was skewed upwards by a few backward coupling times of just over 2,000, but 90% of the draws were made in less than 33 time steps.

References

[1] G. Casella, M. Lavine, and C. Robert. Explaining the perfect sampler. Working Paper 00-16, State University of New York at Stony Brook, Duke University, Durham., 2000.

[2] J.N. Corcoran and R.L. Tweedie. Perfect sampling from Independent Metropolis-Hastings Chains. *Journal of Statistical Planning and Inference*, 104:297–314, 2002.

[3] J.A. Fill. An interruptible algorithm for perfect sampling via Markov chains. *ANNAP*, 8:131–162, 1998.

[4] S.G. Foss and R.L. Tweedie. Perfect simulation and backward coupling. *Stochastic Models*, 14:187–203, 1998.

[5] S.G. Foss, R.L. Tweedie, and J.N. Corcoran. Simulating the invariant measures of Markov chains using horizontal backward coupling at regeneration times. *Prob. Eng. Inf. Sci.*, 12:303–320, 1998.

[6] P. Fulde. *Electron Correlations in Molecules and Solids, Solid State Sci., 3rd edn.* Springer, Berlin, Heidelberg, 1995.

[7] O. Häggström, M.N.M. van Liesholt, and J. Møller. Characterisation results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes. *Bernoulli*, 5:641–659, 1999.

[8] W.S. Kendall. Perfect simulation for the area-interaction point process. In L. Accardi and C.C. Heyde, editors, *Probability Towards the Year 2000*, pages 218–234. Springer, New York, 1998.

[9] R. Kreckel. Parallelization of adaptive mc integrators. *Comput. Phys. Commun.*, 106:258, 1997.

[10] G.P. Lepage. A new algorithm for adaptive multidimensional integration. *J. Comput. Phys.*, 27:192, 1978.

[11] G.D. Mahan. *Many Particle Physics, 2nd edn.* Plenum Press, New York, 1990.

[12] J. Møller. Perfect simulation of conditionally specified models. *JRSSB*, 61(1):251–264, 1999.

[13] P. Monthoux and D.J. Scalapino. Self-consistent $d_{x^2-y^2}$ pairing in a two-imensional hubbard model. *Phys. Rev. Lett.*, 72:1874, 1994.

[14] D.J. Murdoch and P.J. Green. Exact sampling from a continuous state space. *Scandinavian Journal of Statistics*, 25:483–502, 1998.

[15] J.W. Negele and H. Orland. *Quantum Many-Particle Systems.* Addison Wesley, Redwood City, CA, 1988.

[16] C.-H. Pao and N.E. Bickers. Renormalization group acceleration of self- consistent field solutions: two-dimensional hubbard model. *Phys. Rev. B*, 49:1586, 1994.

[17] J.G. Propp and D.B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9:223–252, 1996.

[18] R. Robinson. Private communication. 2002.

[19] S. Veseli. Multidimensional integration in a heterogeneous network environment. *Comput. Phys. Commun.*, 108:258, 1998.

[20] A. Voight, C. Liu, Q. Wang, R.W. Robinson, and H.B. Schüttler. Stochastic feynman diagram summation for the anderson impurity model. *Submitted for publication. Preprint at: http://www.csp.uga.edu/publications/HBS/*, 2002.

Table 1: Summary of Symbols

Notation	Meaning	Additional Information
D	number of dimensions of the lattice	
L_d	integer sidelength of the lattice in dimension d, $(d = 1, 2, \ldots, D)$	
$\mathcal{B}$	set of points in the 1st Brillouin zone	given by equation (4)
N	number of points in $\mathcal{B}$	
$\mathcal{M}$	set of odd Matsubara frequencies	given by equation (5)
$k \equiv (\vec{k}, i\,\nu)$ $\in \mathcal{K} = \mathcal{B} \times \mathcal{M}$	momentum-energy variable (momentum)	consists of a wavevector $\vec{k} \in \mathcal{B}$ and a frequency $i\,\nu \in \mathcal{M}$; associated with "G-lines" (straight lines) in the Feynman diagrams
$k_1, k_2, \ldots$	elements of $\mathcal{K}$	
$k^{(1)}, k^{(2)}, \ldots, k^{(D)}$	components of a vector $\vec{k} \in \mathcal{B}$	
$\Sigma(k)$	self energy	given by equation (3)
$G(k)$	Green's function	given by equation (2)
$q \equiv (\vec{q}, i\,\nu)$, $\vec{q} \in \mathcal{B}$	momentum-energy variable (momentum)	consists of a wavevector $\vec{q} \in \mathcal{B}$ and a frequency $i\,\nu$; associated with "V-lines" (wavy lines) in the Feynman diagrams
$V_{j'j}$	model interaction potential	repulsion energy generated by electrons
$V(\vec{q})$	Fourier transform of interaction potential	given by equation (7)
T_P	physical temperature of the system	
l^g	number of closed G-loops in graph g	Figure 4, $l^g = 1, 0, 0, 1, 0$ for (a), (b), (c), (d), and (e), respectively
s_f	single-fermion spin quantum number	$s_f = \frac{1}{2}$ for non-spin-polarized electrons
δ_{ij}	Kronecker delta	1 when $i = j$, zero otherwise
n_{max}	the highest order graphs that will be considered in computing $\Sigma(k)$	
$F_g^{(n)}(k, k_1, \ldots, k_n)$	the summand of $\Sigma(k)$	given by equation (6)

Table 2: Selected True Versus Estimated Probabilities Based on a Sample of Size 100,000: Example 1 in Section 6

Region	True p	Estimated p
$k_1^{(1)} = 0$	0.4944148	0.49886
$k_1 = 0, \nu_2 = \pi T$	0.2472074	0.24799
$k_0^{(1)} = 0$ or $k_1^{(2)} = 0$	0.7472078	0.74707
Set of 50 randomly generated (k_0, k_1, k_2)-points	0.0962621	0.09632
Set of 70 randomly generated (k_0, k_1, k_2)-points	0.1325202	0.13166
Set of 100 randomly generated (k_0, k_1, k_2)-points	0.1996107	0.20006
Set of 300 randomly generated (k_0, k_1, k_2)-points	0.5827408	0.58275
Set of 375 randomly generated (k_0, k_1, k_2)-points	0.7396903	0.73867
Set of 400 randomly generated (k_0, k_1, k_2)-points	0.7885402	0.78757

Model Parameters: $D = 2$, $L_1 = L_2 = 2$, $|\mathcal{M}_T| = 2$, $\mu = 0.5$, $t = 1.0$, $T_P = 2.0$

Table 3: Estimated Value of $\Sigma_2(0, 0, i\pi T_P)$ Based on Various Sample Sizes: Example 1 in Section 6

True value of $\Sigma_2(0,0,i\pi T_P) = 0.010953844 + 0.997002312i$				
	Sample Size			
Sim	100	1000	10000	100000
1	$0.070568 + 1.894689$	$0.040577 + 1.325870i$	$0.012238 + 1.046736i$	$0.010843 + 0.997868i$
2	$0.083460 + 1.485778$	$0.050968 + 1.285695i$	$0.013005 + 0.968438i$	$0.010914 + 0.997350i$
3	$0.030653 + 0.848791$	$-0.002457 + 0.973494i$	$0.010843 + 0.995879i$	$0.010672 + 0.996325i$
4	$-0.020880 + 1.232835$	$0.045756 + 0.993246i$	$0.010967 + 0.991003i$	$0.010946 + 0.997358i$
5	$0.154370 + 1.345225$	$0.011286 + 0.963402i$	$0.011794 + 1.134048i$	$0.010957 + 0.996579i$
6	$-0.036580 + 0.946579$	$0.010258 + 1.323729i$	$0.020001 + 1.003477i$	$0.010144 + 0.998348i$

Model Parameters: $D = 2$, $L_1 = L_2 = 2$, $|\mathcal{M}_T| = 2$, $\mu = 0.5$, $t = 1.0$, $T_P = 2.0$

Table 4: True Versus Estimated Value of $\Sigma_2(k)$ Based on a Sample of Size 100,000: Example 1 in Section 6

k	$\Sigma_2(k)$	Estimated $\Sigma_2(k)$
$(0,\ 0,\ -i\,\pi T_P)$	0.010954 - 0.997002 i	0.010933 - 0.996102 i
$(0,\ 0,\ i\,\pi T_P)$	0.010954 + 0.997002 i	0.010991 + 1.015137 i
$(0,\ \pi,\ -i\,\pi T_P)$	-0.053717 - 2.295094 i	-0.054136 - 2.297783 i
$(0,\ \pi,\ i\,\pi T_P)$	-0.053717 + 2.295094 i	-0.053234 + 2.284396 i
$(\pi,\ 0,\ -i\,\pi T_P)$	-0.053717 - 2.295094 i	-0.054249 - 2.268983 i
$(\pi,\ 0,\ i\,\pi T_P)$	-0.053717 + 2.295094 i	-0.053113 + 2.300121 i
$(\pi,\ \pi,\ -i\,\pi T_P)$	0.015501 - 1.199040 i	0.010938 - 1.183866 i
$(\pi,\ \pi,\ i\,\pi T_P)$	0.015501 + 1.199040 i	0.016491 + 1.195241 i

Model Parameters: $D = 2,\ L_1 = L_2 = 2,\ |\mathcal{M}_T| = 2,\ \mu = 0.5,\ t = 1.0,\ T_P = 2.0$

Table 5: Estimated Value of $\Sigma_2(0, 0, i\pi T_P)$ Based on Various Sample Sizes Using Conditional Averaging: Example 1 in Section 7.1

True value of $\Sigma_2(0, 0, i\pi T_P) = 0.010953844 + 0.997002312i$

	Sample Size			
Sim	100	1000	10000	100000
1	$-0.078502 + 0.877613\,i$	$0.016364 + 0.938827\,i$	$0.010553 + 0.995451\,i$	$-0.002656 + 0.990609\,i$
2	$-0.020618 + 1.258647\,i$	$-0.132466 + 0.832302\,i$	$-0.017643 + 1.006430\,i$	$0.010735 + 1.004080\,i$
3	$0.283950 + 1.347881\,i$	$0.007313 + 0.952197\,i$	$0.038507 + 1.007398\,i$	$0.011389 + 1.010014\,i$
4	$-0.010512 + 1.054748\,i$	$0.046003 + 1.104158\,i$	$0.014029 + 0.956924\,i$	$0.015495 + 0.997494\,i$
5	$0.050348 + 0.758642\,i$	$0.138418 + 1.078661\,i$	$0.017016 + 1.023525\,i$	$-0.010569 + 0.980516\,i$
6	$-0.030622 + 0.924121\,i$	$0.010811 + 1.088295\,i$	$0.024671 + 0.978156\,i$	$0.010921 + 0.996255\,i$

Model Parameters: $D = 2,\ L_1 = L_2 = 2,\ |\mathcal{M}_T| = 2,\ \mu = 0.5,\ t = 1.0,\ T_P = 2.0$

Table 6: Estimated Value of $\Sigma_2(0, 0, i\pi T_P)$ Based on Various Sample Sizes Using the Modified IMH Algorithm: Example 1 in Section 7.2

True value of $\Sigma_2(0, 0, i\pi T_P) = 0.010953844 + 0.997002312i$

	Sample Size			
Sim	100	1000	10000	100000
1	$0.017980 + 1.199001i$	$-0.119150 + 0.877788i$	$0.013816 + 0.963932i$	$0.010960 + 0.996932i$
2	$-0.323498 + 1.058913i$	$-0.039490 + 1.017291i$	$0.013452 + 0.999321i$	$0.010744 + 0.997154i$
3	$-0.001811 + 0.915970i$	$-0.164076 + 0.952820i$	$0.026331 + 0.991237i$	$0.010955 + 0.997030i$
4	$-0.013037 + 0.897021i$	$-0.011933 + 0.972742i$	$0.023496 + 0.940430i$	$0.010601 + 0.998827i$
5	$-0.025634 + 1.009884i$	$0.013173 + 1.093038i$	$-0.012007 + 1.024184i$	$0.010821 + 0.995304i$
6	$0.072457 + 1.096260i$	$0.003531 + 0.932560i$	$0.020840 + 0.967423i$	$0.015686 + 0.992164i$

Model Parameters: $D = 2$, $L_1 = L_2 = 2$, $|\mathcal{M}_T| = 2$, $\mu = 0.5$, $t = 1.0$, $T_P = 2.0$