Solving the parquet equations for the Hubbard model beyond weak coupling

Ka-Ming Tam,[1] H. Fotso,[1] S.-X. Yang,[1] Tae-Woo Lee,[2,3] J. Moreno,[1,2] J. Ramanujam,[2,4] and M. Jarrell[1,2]
[1]*Department of Physics and Astronomy, Louisiana State University, Baton Rouge, Louisiana 70803, USA*
[2]*Center for Computation and Technology, Louisiana State University, Baton Rouge, Louisiana 70803, USA*
[3]*Seagate, 7801 Computer Avenue South, Bloomington, Minnesota 55435, USA*
[4]*Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, Louisiana 70803, USA*
(Received 1 September 2011; revised manuscript received 6 August 2012; published 29 January 2013)

We find that imposing crossing symmetry in the iteration process considerably extends the range of convergence for solutions of the parquet equations for the Hubbard model. When crossing symmetry is not imposed, the convergence of both simple iteration and more complicated continuous loading (homotopy) methods is limited to high temperatures and weak interactions. We modify the algorithm to impose the crossing symmetry without increasing the computational complexity. We also imposed time reversal and a subset of the point group symmetries, but they did not further improve the convergence. We elaborate the details of the latency hiding scheme which can significantly improve the performance in the computational implementation. With these modifications, stable solutions for the parquet equations can be obtained by iteration more quickly even for values of the interaction that are a significant fraction of the bandwidth and for temperatures that are much smaller than the bandwidth. This may represent a crucial step towards the solution of two-particle field theories for correlated electron models.

DOI: 10.1103/PhysRevE.87.013311 PACS number(s): 02.60.−x, 71.10.Fd, 71.27.+a

I. INTRODUCTION

A natural step to extend most of the existing many-body single-particle self-consistent methods is to include the full momentum and energy dependence of the vertex corrections. Historically, the self-consistent approach for vertex corrections was first considered by Landau, Abrikosov, and Khalatnikov in the context of the high-energy behavior of quantum electrodynamics [1]. The original goal was to develop a nonperturbative method which encodes the information in terms of a system of closed integral equations. The parquet equations, in principle, provide a framework for self-consistent determination of the self-energy and the vertex corrections. They were proposed for both boson-boson scattering and fermion-fermion scattering during the 1950s [2,3]. Methods similar to the parquet equations were first introduced in the context of many-body theory by de Dominicis and Martin [4,5]. One of the early practical applications was on the x-ray absorption and emission problem by Roulet, Gavoret, and Nozières [6]. Since then, various problems have been studied by the parquet summation approach, most notably, the Fermi liquid in a strong magnetic field [7–9], the disordered electron gas in a strong transverse magnetic field [10], the Anderson impurity model [11–14], random potential problems [15–17], the Hubbard model [18–24], helium-4 [25,26], helium-3 [27], local moment formation [28,29], the vortex liquid model [30–33], the matrix models [34,35], and nuclear structure calculations [36]. While these applications of the parquet formulation provide a lot of important insights, most of the calculations are based on various approximated forms of the parquet equations.

It is obvious that going from a one-particle to a two-particle self-consistent calculation represents a significant increase in the computational effort, as each two-particle vertex contains three independent momentum and frequency indices. From the point of view of practical calculation, the number of elements for each index is around a few thousands. Therefore, the number of elements for the vertices is of the order of tens of millions to a few billions. Moreover, all the information is encoded in integral equations with a complicated structure, in which simplifications do not seem to be immediately possible. Indeed, in the past, the most successful application using the full parquet equations was largely limited to the single-impurity Anderson model [11]. With recent advances in computational infrastructure where petascale performance has become available, calculations for lattice models such as the Hubbard model are now feasible. For example, the solution of the parquet equations for a 4×4 Hubbard cluster with on-site coupling $U = 2t$ and temperature $T = 0.3t$ was recently obtained [37].

However, limitations on computer performance and storage are apparently not the sole obstacle for obtaining the solution of the parquet equations. Another major barrier is the stability of the solvers. The simple iteration method, which is widely adapted for the dynamical mean-field method, often fails to provide a stable solution for the parquet equations. In most cases, a damping scheme has to be employed. Even with the damping scheme, when the temperature is low or the coupling is large, finding a stable solution still seems to be rather difficult [37].

Given the large number of variables and the complexity of the parquet equations, instabilities in their solution may not be unexpected. Methods based on the local gradient are not likely to be suitable as the Hessian cannot be readily calculated. Most of the nonlinear solvers have only local convergence properties. This may not pose a problem if we have a reasonable guess which is close enough to the true solution. Unfortunately, it is not easy to obtain a good initial guess for the parquet equations. Methods that in principle allow "global" convergence, for example, the homotopy method or continuous loading method, have been proposed as possible ways to improve the calculations [38]. While these tend to improve convergence, many steps are required for the

solution to move along the homotopy path. Moreover, practical experience seems to suggest that convergence may still not be achieved when the temperature is low or the coupling is strong. It is clear that a better solver is definitely required for the practical application of the parquet method within the context of strongly correlated systems.

One of the most prominent differences between the parquet formulation and most of the other approximation schemes, such as the random phase approximation (RPA) [39,40], the self-consistent spin fluctuations approach [41], and the fluctuation exchange approach [42], is that the so-called crossing symmetry is obeyed by construction of the parquet equations. The crossing symmetry [43] implies that a vertex in one channel can also produce a vertex in all other channels by pulling or crossing the vertex legs and multiplying by appropriate constants. It also implies that the Pauli exclusion principle is automatically satisfied. However, in the course of the iteration process, as long as the exact solution of the parquet equations is not obtained, crossing symmetry is violated. The main point in the present paper is to highlight that crossing symmetry is crucial for obtaining a stable solution. We devise a modified iteration scheme which can obtain a stable solution for the parquet equations at lower temperature and stronger coupling than in the previous schemes [37]. This is achieved primarily by restoring the crossing symmetry at each step of the iteration.

It is important to notice that because of the large number of vertex functions, for production runs, massively parallel machines are absolutely necessary. Since the vertex functions in different channels are mixed in the parquet formulation, an efficient scheme to transform the vertex function storage in different nodes is critical to improving the overall efficiency of the calculations. Some of the computational details have been explained in a previous publication [37]. We have further improved the scheme, which allows us to hide the communication latency across different nodes behind the local calculations within the nodes, effectively further speeding up the calculations.

The model used for testing the computational scheme for solving the parquet equations is the standard Hubbard model at half filling. The Hamiltonian is

$$H = -t \sum_{\langle i,j\rangle,\sigma} (c^{\dagger}_{i,\sigma} c_{j,\sigma} + \text{H.c.}) + U \sum_{i} n_{i,\uparrow} n_{i,\downarrow}, \qquad (1)$$

where U is the on-site repulsion and $t = 1$ is the hopping matrix which establishes the unit of energy.

The paper is organized as follows. In Sec. II, we reproduce the parquet equations, which also allows us to fix the notation. In Sec. III, we describe the iteration scheme for solving the parquet equations. In Sec. IV, we discuss the violation of crossing symmetry. In Sec. V, we present a modified iteration scheme which explicitly restores the crossing symmetry. We also discuss the limitation of the modified scheme and the possible directions for further developments. In Sec. VI, we present the leading eigenvalues of the antiferromagnetic channel as a function of the temperature and coupling strength and find that the parameter region of stable solutions is greatly increased when crossing symmetry is enforced. A brief summary is contained in Sec. VII. In the Appendix, we present a latency hiding scheme which allows substantial increase of the efficiency for solving the parquet equations.

II. PARQUET EQUATIONS

A. Derivation of the parquet equations

The parquet equations have been elaborated in the literature [19,20,24,37,44,45]. For the completeness of the paper we outline their derivation in this section. We intend to highlight the structure of the parquet equations and to fix the notation needed for the discussion of the numerical implementation.

The parquet equations [1] are essentially a generalization of the Bethe-Salpeter equation [43,46,47] that was originally proposed to obtain a bound state solution by summing up an infinite series of diagrams. From the perspective of an iterative process, it is similar to finding the bound state solution by scattering an infinite number of times. The scattering processes included in the Bethe-Salpeter equation are two-particle scatterings; therefore, the diagrams generated are always reducible. *Reducible* diagrams, in the present context, are defined as the ones that can be separated into two pieces by cutting two fermion lines. The first term of the scattering can be considered as the input, or from an iteration perspective, the initial condition of the iteration process. This initial term is called the *fully irreducible* vertex, and is denoted by Λ.

The *full* vertex F contains the sum of the *fully irreducible* part, given by the input Λ and the *reducible* part from the repeated scattering terms. For the two-body interaction there are three different channels for the reducible vertices, one particle-particle channel, and two particle-hole channels. These can be understood simply from perturbation theory with two-body coupling at one-loop level. Consider the action

$$S = \int d(1)\bar{\Psi}(1)G_0(1)\Psi(1) + \int d(1)d(2)d(3)d(4)u^{12}_{34}\bar{\Psi}(1)\bar{\Psi}(2)\Psi(3)\Psi(4), \qquad (2)$$

where Ψ and $\bar{\Psi}$ are Grassmann fields. The momentum and Matsubara frequency indices are denoted collectively as $(1,2,\ldots)$. Following the standard perturbation expansion of the vertex function (see, e.g., Refs. [48,49]), at one-loop level (second-order expansion with respect to u) the three diagrams for the vertex corrections are the particle-particle BCS channel; and the particle-hole zero-sound [50–54] and Peierls channels. In the literature of parquet formulation they are sometimes denoted as the particle-particle (PP), "vertical" particle-hole (PH), and "horizontal" particle-hole ($\overline{\text{PH}}$) channels [19]. We reproduce these three one-loop terms here (see Fig. 1 for the diagrammatic representation):

BCS:

$$\text{PP}_{1\text{ loop}}(u,u') = \frac{1}{2}\int d(5)d(6)u^{15}_{26}u'^{53}_{64}G(5)G(6); \qquad (3)$$

zero sound:

$$\text{PH}_{1\text{ loop}}(u,u') = \int d(5)d(6)u^{56}_{13}u'^{65}_{24}G(5)G(6); \qquad (4)$$

Peierls:

$$\overline{\text{PH}}_{1\text{ loop}}(u,u') = -\int d(5)d(6)u^{56}_{14}u'^{65}_{23}G(5)G(6). \qquad (5)$$

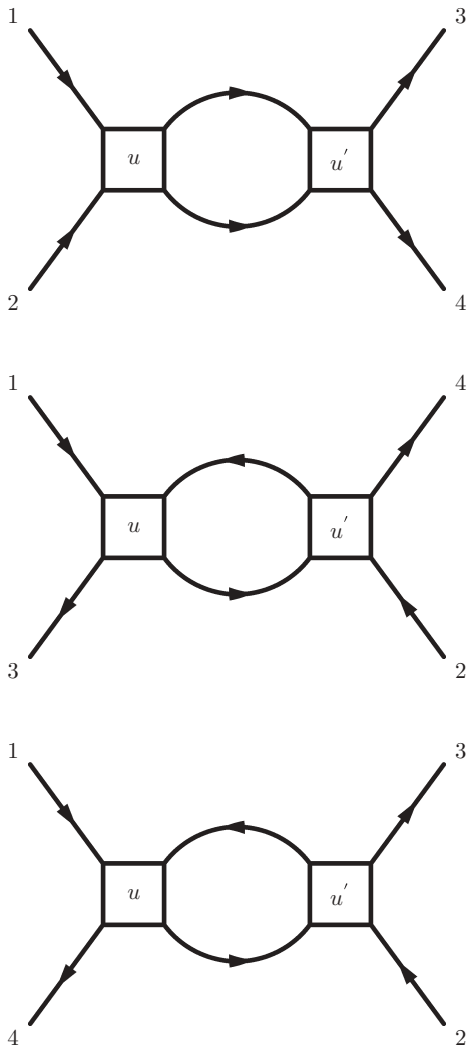

FIG. 1. Feynman diagrams for the second-order vertex correction: the BCS diagram–particle-particle channel (top), the zero-sound diagram–vertical particle-hole channel (middle), and the Peierls diagram–horizontal particle-hole channel (bottom). These diagrams are reducible in the sense that they can be cut into two pieces by cutting two fermion lines.

We introduced the functions $\mathrm{PP}_{1\,\mathrm{loop}}(u,u')$, $\mathrm{PH}_{1\,\mathrm{loop}}(u,u')$, and $\overline{\mathrm{PH}}_{1\,\mathrm{loop}}(u,u')$ with the kernels u and u' to facilitate the following dicussion of the parquet equation. Clearly, at the one-loop level $u' = u$; however, it need not be the case in general. It is important to note that at the one-loop level the particle-particle channel itself is crossing symmetric. On the other hand, the two particle-hole channels are not crossing symmetric by themselves, although their sum is. The full vertex, which is defined as the sum of all three channels at the one-loop level ($F_{1\,\mathrm{loop}}$), can be written as

$$F_{1\,\mathrm{loop}} = u + \mathrm{PP}_{1\,\mathrm{loop}}(u,u) + \mathrm{PH}_{1\,\mathrm{loop}}(u,u) + \overline{\mathrm{PH}}_{1\,\mathrm{loop}}(u,u). \tag{6}$$

One of the methods to generate higher-order diagrams is the so-called vertex insertion method. The parquet diagrams are the diagrams which can be generated by replacing one of the vertices with the one-loop diagram. For example, two-loop diagrams can be obtained from $\mathrm{PP}_{1\,\mathrm{loop}}(u,u')$ (where u' contains the one-loop correction), and similarly from the $\mathrm{PH}_{1\,\mathrm{loop}}$ and $\overline{\mathrm{PH}}_{1\,\mathrm{loop}}$ one-loop diagrams. $(N+1)$th-order diagrams can be generated iteratively, by replacing one of the vertices in an Nth-order diagram with a one-loop diagram. The parquet formulation is essentially a systematic way to sum up all the parquet diagrams that can be generated by this iterative scheme. Therefore, ultimately all the diagrams generated within the parquet formulation are two-particle reducible in at least one of the three channels.

The central idea is to organize the parquet diagrams in a systematic manner to allow the sum to be calculated. In principle, the idea is the same as that for deriving the usual Bethe-Salpeter equation. The difficulty here is that more than one channel has to be considered. First, it is convenient to introduce the irreducible vertices Γ for every channel. This includes all the parquet diagrams which cannot be cut into two pieces in their own channel. The reducible bubbles $(\Psi_{PP}, \Phi_{PH}, \Phi_{\overline{PH}})$ (see Fig. 2) are formed by replacing one of the vertex in the one-loop diagram by Γ and the other by the full vertex F (which includes all the parquet diagrams), we define them as

$$\Psi_{\mathrm{PP}} = \mathrm{PP}_{1\,\mathrm{loop}}(\Gamma_{\mathrm{PP}}, F), \tag{7}$$

$$\Phi_{\mathrm{PH}} = \mathrm{PH}_{1\,\mathrm{loop}}(\Gamma_{\mathrm{PH}}, F), \tag{8}$$

$$\Phi_{\overline{\mathrm{PH}}} = \overline{\mathrm{PH}}_{1\,\mathrm{loop}}(\Gamma_{\overline{\mathrm{PH}}}, F). \tag{9}$$

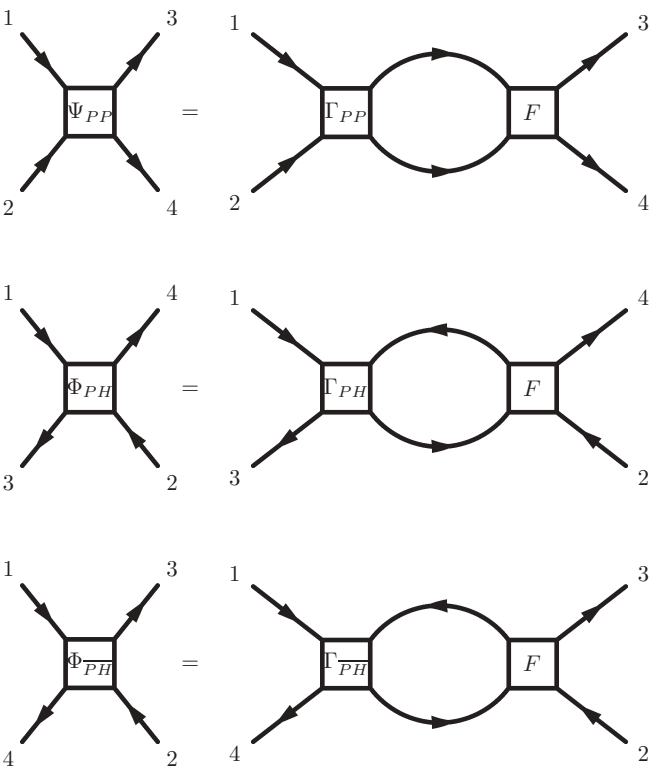

FIG. 2. Diagrams for the reducible bubbles for the three different channels. The BCS diagram–particle-particle channel (top), the zero-sound diagram–vertical particle-hole channel (middle), and the Peierls diagram–horizontal particle-hole channel (bottom). These bubbles are reducible in the sense that they can be cut into two pieces by cutting two fermion lines.

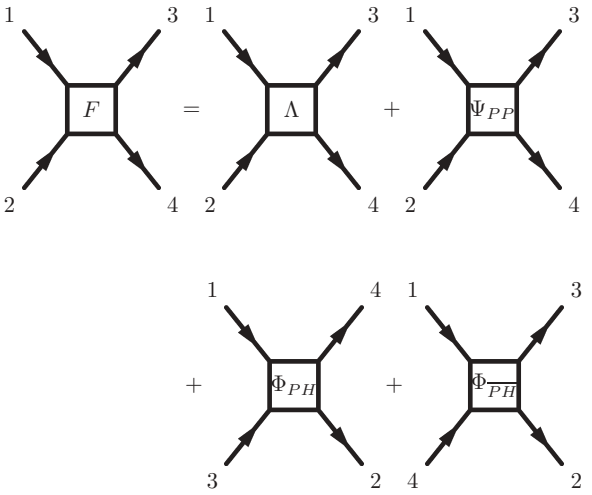

FIG. 3. Diagrams for the full vertex containing the fully irreducible vertex and the contribution from all three channels. The fully irreducible in most practical calculations is choosen as the bare vertex. It is often said that the above equation for the full vertex is "exact", however in practice it is almost never possible to calculate the fully irreducible vertex. In essence, this equation separates the total contribution for the full vertex into two parts: parquet diagrams and non-parquet diagrams.

In principle, we can replace the bare vertices used in the one-loop perturbation by any corrected vertex obtained by methods that include nonparquet diagrams, for example the vertex from the dynamical mean-field approximation. Therefore instead of the bare u we can have the corrected $u_{\rm corr}$. In this sense, it is appropriate to call $u_{\rm corr}$ the input of the perturbation. In the context of parquet formulation, this input is usually denoted as the fully irreducible vertex Λ, as it contains diagrams which cannot be separated into two pieces by cutting any two Green function lines.

The full vertex F (see Fig. 3) should be given by the sum of the fully irreducible vertex and the contribution from three different channels, that is,

$$F = \Lambda + \Psi_{\rm PP} + \Phi_{PH} + \Phi_{\overline{\rm PH}}. \tag{10}$$

This is essentially Eq. (6) but the kernel of the bubbles is not the bare coupling, u.

There is an overcounting due to the reducible bubbles. Consider the particle-particle bubble $\Phi_{\rm PP} = {\rm PP}_{1\ {\rm loop}}(\Gamma_{\rm PP}, F)$, since the full vertex F already contains the diagrams which are reducible in the particle-particle channel, they should be subtracted from $\Gamma_{\rm PP}$ (see Fig. 4). Similarly, for the other two particle-hole channels,

$$\Gamma_{\rm PP} = \Lambda + \Phi_{\rm PH} + \Phi_{\overline{\rm PH}}, \tag{11}$$

$$\Gamma_{\rm PH} = \Lambda + \Psi_{\rm PP} + \Phi_{\overline{\rm PH}}, \tag{12}$$

$$\Gamma_{\overline{\rm PH}} = \Lambda + \Psi_{\rm PP} + \Phi_{\rm PH}. \tag{13}$$

These are the parquet equations which take into consideration channel mixing. The parquet equations together with the Bethe-Salpeter equations complete the closed system of equations for the vertex functions. The remaining part of the self-consistent calculation at the two-particle level is

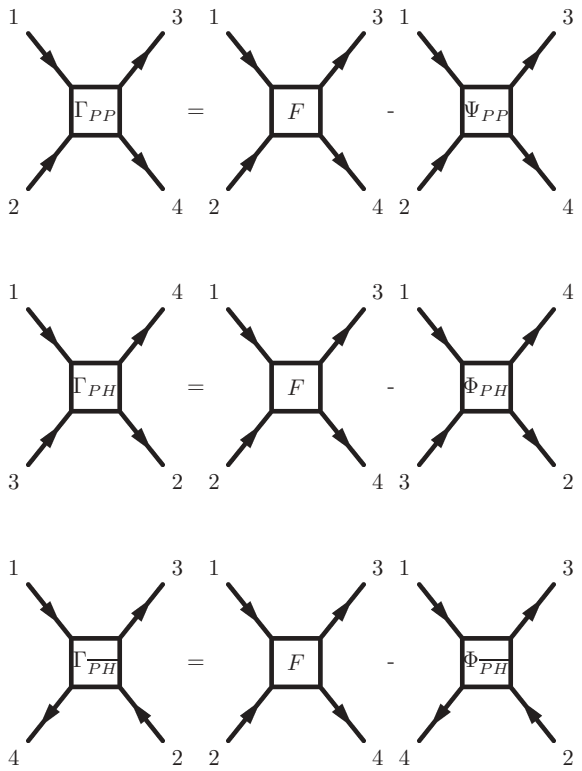

FIG. 4. Diagrams for the irreducible bubbles for the three different channels. The BCS diagram–particle-particle channel (top), the zero-sound diagram–vertical particle-hole channel (middle), and the Peierls diagram–horizontal particle-hole channel (bottom). These bubbles are reducible in the sense that they can be cut into two pieces by cutting two fermion lines.

to connect the self-energy with the vertex function via the Dyson-Schwinger equation [55,56]

$$\Sigma(1) = \int d(2) \int d(3) \int d(4) G(2) G(3) G(4) F^{24}_{13} u^{42}_{31}. \tag{14}$$

The self-energy is related to the dressed Green function by the Dyson equation

$$G^{-1} = G_0^{-1} - \Sigma, \tag{15}$$

where G_0 is the bare Green function. The corrected self-energy or the one-particle Green function can be fed back into the parquet equations and the Bethe-Salpeter equation for the fermion lines. Therefore, the self-consistency loop is completed for the two-particle vertex and one-particle self-energy.

The Bethe-Salpeter equations, the parquet equations, the Dyson-Schwinger equation, and the Dyson equation form a closed set of integral equations which in principle allow the vertex correction and the self-energy correction to be calculated in a self-consistent manner given the input of the fully irreducible vertex. One of the advantages of the parquet formulation is that the vertex obtained from the solution of these equations satisfies crossing symmetry exactly. This is fundamentally due to the channel mixing in the parquet equations which enforces this symmetry explicitly. Most of the one-particle approximations and biased two-particle approximations fail to fulfill this symmetry.

B. Parquet equations in spin-diagonalized form

Note that the particle-hole channel can be obtained from either the irreducible vertex $\Gamma_{\rm PP}$ or the irreducible vertex $\Gamma_{\rm PH}$, using the crossing symmetry we can eliminate one of the irreducible particle-hole channels [19,20]. Since we are mostly interested in the models which possess SU(2) spin rotation symmetry, it is convenient to preserve this symmetry, as this is an exact symmetry for our two-dimensional calculations at nonzero temperature, by decomposing the vertices in the so-called spin-diagonalized representation [19,20]. In this representation the three different channels with the additional spin degree of freedom can be decomposed into four different channels; they are the the spin channel, charge channel, spin-singlet channel, and spin-triplet channel, which we denote as the m channel, d channel, s channel, and t channel, respectively.

We reproduce the full set of equations with all the indices for the parquet formulation in the spin-diagonalized representation in the following [19,24,37]. First we define the different channels:

$$\Gamma_d = \Gamma^{\rm PH}_{\uparrow\uparrow;\uparrow\uparrow} + \Gamma^{\rm PH}_{\uparrow\uparrow;\downarrow\downarrow}, \tag{16}$$

$$\Gamma_m = \Gamma^{\rm PH}_{\uparrow\uparrow;\uparrow\uparrow} - \Gamma^{\rm PH}_{\uparrow\uparrow;\downarrow\downarrow}, \tag{17}$$

$$\Gamma_s = \Gamma^{\rm PP}_{\uparrow\downarrow;\uparrow\downarrow} - \Gamma^{\rm PP}_{\uparrow\downarrow;\downarrow\uparrow}, \tag{18}$$

$$\Gamma_t = \Gamma^{\rm PP}_{\uparrow\downarrow;\uparrow\downarrow} + \Gamma^{\rm PP}_{\uparrow\downarrow;\downarrow\uparrow}, \tag{19}$$

and similarly for F and Λ.

The Schwinger-Dyson equation is

$$\begin{aligned}\Sigma(P) = -\frac{UT^2}{4N}\sum_{P',Q}\{&G(P')G(P'+Q)G(P-Q)\\ &\times[F_d(Q)_{P-Q,P'} - F_m(Q)_{P-Q,P'}]\\ &+ G(-P')G(P'+Q)G(-P+Q)\\ &\times[F_s(Q)_{P-Q,P'} + F_t(Q)_{P-Q,P'}]\},\end{aligned} \tag{20}$$

where G is the single-particle Green function, which itself can be calculated from the self-energy using the Dyson equation,

$$G^{-1}(P) = G_0^{-1}(P) - \Sigma(P), \tag{21}$$

where G_0 is the bare Green function. Here, the indices P, P', and Q combine the momentum $\mathbf{k}$ and Matsubara frequency $i\omega_n$, i.e., $P = (\mathbf{k}, i\omega_n)$.

The reducible and the irreducible vertices in a given channel are related by the Bethe-Salpeter equation,

$$F_r(Q)_{P,P'} = \Gamma_r(Q)_{P,P'} + \Phi_r(Q)_{P,P'}, \tag{22}$$

$$F_{r'}(Q)_{P,P'} = \Gamma_{r'}(Q)_{P,P'} + \Psi_{r'}(Q)_{P,P'}, \tag{23}$$

where $r = d$ or m for the density and magnetic channels and $r' = s$ or t for the spin-singlet and spin-triplet channels. The vertex ladders are defined as

$$\Phi_r(Q)_{P,P'} \equiv \sum_{P''} F_r(Q)_{P,P''}\chi_0^{\rm PH}(Q)_{P''}\Gamma_r(Q)_{P'',P'}, \tag{24}$$

$$\Psi_{r'}(Q)_{P,P'} \equiv \sum_{P''} F_{r'}(Q)_{P,P''}\chi_0^{\rm PP}(Q)_{P''}\Gamma_{r'}(Q)_{P'',P'}, \tag{25}$$

where χ_0 is the product of two single-particle Green functions.

The parquet equations in the spin-diagonalized representation are

$$\begin{aligned}\Gamma_d(Q)_{PP'} = \Lambda_d(Q)_{PP'} &- \tfrac{1}{2}\Phi_d(P'-P)_{P,P+Q}\\ &- \tfrac{3}{2}\Phi_m(P'-P)_{P,P+Q}\\ &+ \tfrac{1}{2}\Psi_s(P+P'+Q)_{-P-Q,-P}\\ &+ \tfrac{3}{2}\Psi_t(P+P'+Q)_{-P-Q,-P},\end{aligned} \tag{26}$$

$$\begin{aligned}\Gamma_m(Q)_{PP'} = \Lambda_m(Q)_{PP'} &- \tfrac{1}{2}\Phi_d(P'-P)_{P,P+Q}\\ &+ \tfrac{1}{2}\Phi_m(P'-P)_{P,P+Q}\\ &- \tfrac{1}{2}\Psi_s(P+P'+Q)_{-P-Q,-P}\\ &+ \tfrac{1}{2}\Psi_t(P+P'+Q)_{-P-Q,-P},\end{aligned} \tag{27}$$

$$\begin{aligned}\Gamma_s(Q)_{PP'} = \Lambda_s(Q)_{PP'} &+ \tfrac{1}{2}\Phi_d(P'-P)_{-P',P+Q}\\ &- \tfrac{3}{2}\Phi_m(P'-P)_{-P',P+Q}\\ &+ \tfrac{1}{2}\Phi_d(P+P'+Q)_{-P',-P}\\ &- \tfrac{3}{2}\Phi_m(P+P'+Q)_{-P',-P},\end{aligned} \tag{28}$$

$$\begin{aligned}\Gamma_t(Q)_{PP'} = \Lambda_t(Q)_{PP'} &+ \tfrac{1}{2}\Phi_d(P'-P)_{-P',P+Q}\\ &+ \tfrac{1}{2}\Phi_m(P'-P)_{-P',P+Q}\\ &- \tfrac{1}{2}\Phi_d(P+P'+Q)_{-P',-P}\\ &- \tfrac{1}{2}\Phi_m(P+P'+Q)_{-P',-P}.\end{aligned} \tag{29}$$

It is important to note at this point that if we substitute the irreducible vertices Γ [Eqs. (26), (27), (28), and (29)] into the Bethe-Salpeter equation [Eqs. (22) and (23)] the crossing symmetry in the full vertex F is automatically satisfied regardless of the numerical values of the vertex ladders Φ and Ψ, assuming that the fully irreducible vertices Λ are crossing symmetric. We write all the full vertices explicitly in the following using only the vertex ladders Φ and Ψ and the fully irreducible vertices Λ:

$$\begin{aligned}F_d(Q)_{P,P'} = \Lambda_d(Q)_{PP'} &- \tfrac{1}{2}\Phi_d(P'-P)_{P,P+Q}\\ &- \tfrac{3}{2}\Phi_m(P'-P)_{P,P+Q}\\ &+ \tfrac{1}{2}\Psi_s(P+P'+Q)_{-P-Q,-P}\\ &+ \tfrac{3}{2}\Psi_t(P+P'+Q)_{-P-Q,-P}\\ &+ \Phi_d(Q)_{P,P'};\end{aligned} \tag{30}$$

$$\begin{aligned}F_m(Q)_{P,P'} = \Lambda_m(Q)_{PP'} &- \tfrac{1}{2}\Phi_d(P'-P)_{P,P+Q}\\ &+ \tfrac{1}{2}\Phi_m(P'-P)_{P,P+Q}\\ &- \tfrac{1}{2}\Psi_s(P+P'+Q)_{-P-Q,-P}\\ &+ \tfrac{1}{2}\Psi_t(P+P'+Q)_{-P-Q,-P}\\ &+ \Phi_m(Q)_{P,P'};\end{aligned} \tag{31}$$

$$\begin{aligned}F_s(Q)_{P,P'} = \Lambda_s(Q)_{PP'} &+ \tfrac{1}{2}\Phi_d(P'-P)_{-P',P+Q}\\ &- \tfrac{3}{2}\Phi_m(P'-P)_{-P',P+Q}\\ &+ \tfrac{1}{2}\Phi_d(P+P'+Q)_{-P',-P}\\ &- \tfrac{3}{2}\Phi_m(P+P'+Q)_{-P',-P}\\ &+ \Psi_s(Q)_{P,P'};\end{aligned} \tag{32}$$

$$
\begin{aligned}
F_t(Q)_{P,P'} = {} & \Lambda_t(Q)_{PP'} + \tfrac{1}{2}\Phi_d(P'-P)_{-P',P+Q} \\
& + \tfrac{1}{2}\Phi_m(P'-P)_{-P',P+Q} \\
& - \tfrac{1}{2}\Phi_d(P+P'+Q)_{-P',-P} \\
& - \tfrac{1}{2}\Phi_m(P+P'+Q)_{-P',-P} \\
& + \Psi_t(Q)_{P,P'}. \qquad (33)
\end{aligned}
$$

These relations allow us to restore the crossing symmetry for the full vertices without heavy computational overhead.

The prominent technical problem at hand is whether or not we can solve this set of equations efficiently without resorting to any approximation scheme. An obvious difficulty is to handle the large number of variables. On going from the one-particle-level calculation to a two-particle-level calculation, the number of variables which has to be monitored grows as the third power of the linear dimension of the system. If N_t is the number of lattice sites times the number of discrete Matsubara frequencies, i.e., $N_t = N_k \times N_\omega$, the largest N_t that can be handled is in the range 2000–3000, i.e., the number of variables can be over 1×10^9. One can immediately see that practical calculations for reasonably large system sizes pose a serious problem, although not insurmountable with modern computational facilities where a large number of computer nodes are accessible. However, in addition to the large number of computations associated with solving the parquet equations, they also require a complex communication pattern between the different processes, as we discuss in more detail in the Appendix. Moreover, a numerical instability is not unexpected, especially when the system is in the proximity of a phase transition.

III. NUMERICAL IMPLEMENTATION

The parquet formulation consists of two sets of equations. The first set, e of the parquet equations and the Bethe-Salpeter equation, determines the full vertex F and the irreducible vertex Γ given the one-particle self-energy Σ and the fully irreducible vertex Λ as the input. The second set of equations determines the one-particle quantities given the full vertex F; it includes the Schwinger-Dyson equation and the Dyson equation.

Since the method is iteration based, the initial guess is crucial for obtaining a converged solution. In principle, the initial guess can be approximated, for example, by second-order perturbation theory. However, in practice, this is not the optimal choice, especially when the self-energy from second-order perturbation theory is small. In this case the Green function will quickly destabilize the calculation. This may relate to the fact that the damping from the imaginary part of the self-energy is quickly reduced. Since we are supposing that we know the fully irreducible vertex Λ, a practical scheme is to choose the irreducible, Γ, and full, F, vertices, equal to Λ, and a large value (a few times the bandwidth) for the imaginary part of the self-energy.

Figure 5 is an illustration of the flow diagram of the algorithm, where the fully irreducible vertices are the initial input for the calculation. The algorithm can be described as follows:

(1) Set the initial conditions for the irreducible and full vertices, and the self-energy.

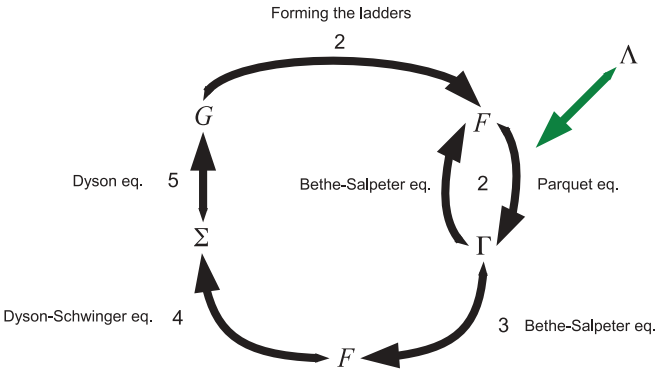

FIG. 5. (Color online) Flow diagram of the algorithm for solving the parquet equations. See the text for the description of each step. The major computational bottleneck is in the self-consistent loop of step 2. The cross-channel rotations of the vertex ladders from the form required by the Bethe-Salpeter equations to that in the parquet equations require expensive communications across different nodes in the parallel implementation.

(2) Update the Green functions and calculate the bare susceptibility χ_0, which is given by the product of two Green functions. Solve the parquet and the Bethe-Salpeter equations for the irreducible vertices Γ. Simple iteration is used until the convergence criteria are met for the irreducible vertices.

This completes the update of the vertices. The next step is to use the irreducible vertices obtained from the parquet equations to construct the full vertices.

(3) Solve the Bethe-Salpeter equation to obtain the full vertices F using the irreducible vertices from the previous step; this is executed exactly by calling the LAPACK routines for the inverse of the matrices [57].

With the full vertices obtained, we can update the self-energy.

(4) Solve the Dyson-Schwinger equation to obtain the self-energy from the full vertices. Simple iteration is used until certain convergence criteria are met for the self-energy.

(5) Solve the Dyson equation for the fully dressed Green function from the self-energy.

This completes the iteration loop, and the procedure is repeated from step 2, until convergence is reached for both the self-energy and the irreducible vertices.

In practice, step 2 which attempts to obtain the irreducible vertices needs to be iterated for a few times to get a reasonable convergence, even in the case where the coupling is weak and the temperature is high. On the other hand, step 4 which attempts to obtain the self-energy from the updated full vertices is not iterated more than once at each cycle of the loop, so as to avoid instability (we define instability here as the failure to obtain a converged solution from the iterative solver). Attempting to solve for the self-energy at the early stage of the iterative procedure where the full vertices are not well converged can generally lead to instability. Although in the present paper we focus only on the Hubbard model, instabilities in the iteration process have also been observed in solving the parquet equations for nuclear structure calculations [36].

A widely used method to avoid the instability in the iterative process is to introduce a damping factor in the updates of the

variables. The updates are modified as follows:

$$\Sigma = (\alpha)\Sigma_{\text{new}} + (1-\alpha)\Sigma_{\text{old}}; \tag{34}$$

$$\Gamma = (\alpha)\Gamma_{\text{new}} + (1-\alpha)\Gamma_{\text{old}}. \tag{35}$$

With this damping scheme, the solution for the half-filled Hubbard model on a 4×4 cluster has been successfully obtained for $U = 2t$ and temperature $T = 0.3t$ [37]. However, in the strong-coupling regime, obtaining a stable solution still seems to be difficult, even though a rather heavy damping is employed.

We will demonstrate the instability problem of the simple iterative process by monitoring the leading eigenvalues λ defined as

$$\lambda_r \phi_r = \Gamma_r(P,P')G(P')G(P'+(\pi,\pi))\phi_r \tag{36}$$

for $r = d$ and m; similarly

$$\lambda_{r'}\phi_{r'} = (-1/2)\Gamma_{r'}(P,P')G(-P')G(P')\phi_{r'} \tag{37}$$

for $r' = s$ and t. In principle, these leading eigenvalues signal a phase transition by going through 1, expressing the divergence of the susceptibilities in the corresponding channel.

In Fig. 6, we plot the leading eigenvalues of the density, magnetic, spin-singlet, and spin-triplet channels as functions of the number of iteration steps calculated with this simple iteration (SI) method. The calculation is done on a 2×2 cluster with 32 frequencies and temperature $T = 0.4t$; the damping parameter is $\alpha = 0.1$. A converged solution is obtained for $U = 2t$; however, for $U = 4t$ and $6t$ the iterative solutions diverge. Changing the damping or the initial self-energy does not help in obtaining a converged solution for the larger values of U. These are illustrative examples which show the problem of using the simple iteration method for solving the parquet equations. For weak coupling and not too low temperature, converged solutions can be obtained. Beyond weak coupling the iteration becomes divergent.

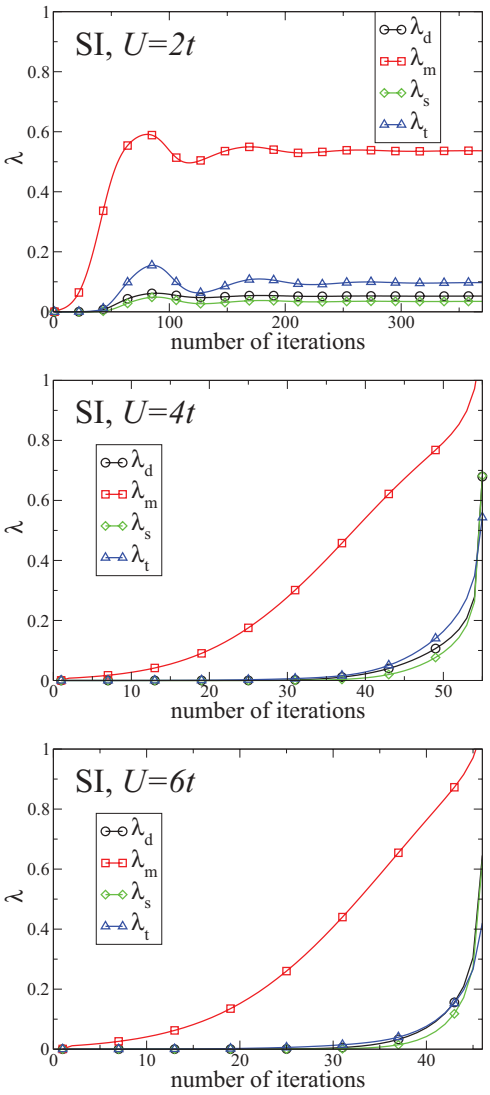

FIG. 6. (Color online) The leading eigenvalues of various channels [density (d), magnetic (m), spin singlet (s), and spin triplet (t)] as functions of the number of iteration steps calculated with the simple iteration (SI) method. The calculations are for the half-filled Hubbard model on a 2×2 cluster at temperature $T = 0.4t$. The initial condition for the self-energy is set at $0 + i320t$, and that for the irreducible vertex is set at the bare Hubbard coupling. The damping factor α is set at 0.1. The solution is well converged for $U = 2t$. However, divergence occurs for $U = 4t$ at the 55th iteration step. For $U = 6t$, divergence occurs at the 46th iteration step.

A. Continuous loading method

The fixed point iteration method is widely used to solve the self-consistent equations arising in the context of many-body physics, e.g., from the dynamical mean-field approximation. Generically the fixed point iteration method can be used to solve the equation $f(x) = x$, where $f : \mathbb{R}^n \to \mathbb{R}^n$. For example, in the dynamical mean-field approximation, the function f is the self-consistency equation, x is the self-energy, and n is equal to $2N_\omega$, where N_ω is the number of Matsubara frequencies.

However, this simple fixed point iteration method provides a convergent solution for the parquet formulation only in the weak-coupling and high-temperature regime. A possible method to alleviate the divergence in the nonlinear solver is the so-called continuous loading or homotopy method. The continuous loading method constructs an auxiliary problem, which is easier to solve than the original one, and then gradually deforms the auxiliary problem into the original one. For the purpose of illustration, consider that we want to solve an equation $f(x) = x$, where $f(x) : \mathbb{R}^1 \to \mathbb{R}^1$ with a single variable x. Instead of solving the equation $f(x) = x$, we devise a function $g(x,\nu) : \mathbb{R}^2 \to \mathbb{R}^1$ which is defined as $g(x,\nu) \equiv \nu f(x) + (1-\nu)x_0$, where x_0 is the solution for the equation $g(x,\nu = 0) = x$, and ν is a tuning parameter used to deform the function $g(x,\nu)$. One can easily see that $g(x,\nu = 1) = f(x)$, while the solution of $g(x,\nu = 0) = x$ is trivially given by $x = x_0$. There are a few different methods to deform the function g. One of the strategies is to first solve $g(x,\nu_0) = x$, where ν_0 is small and thus the equation is easy to solve by the fixed point iteration. The solution obtained is used as the initial condition to solve $g(x,\nu_1) = x$, where $\nu_1 > \nu_0$, and this process is continued until the $g(x,\nu)$ is deformed to the original function $f(x)$. Therefore, a series of ν_i is needed

to gradually deform the auxiliary problem $g(x,\nu) = x$ into the original problem $f(x) = x$.

When applying the continuous loading method to solve the parquet formulation, we write down the set of parquet equations as $\mathbf{f}_{\text{parquet}}(\{\Sigma,\Gamma\}) = \{\Sigma,\Gamma\}$, where $\mathbf{f}_{\text{parquet}} : \mathbb{R}^n \to \mathbb{R}^n$ is a large set of equations with dimension $n = 2(4N_t^3 + N_t)$, where the $4N_t^3$ is due to the four different channels of irreducible vertices (Γ), N_t is due to the self energy (Σ), and the factor of 2 is due to the complex variables. We define the auxiliary function as $\mathbf{g} = \nu\mathbf{f}_{\text{parquet}} + (1-\nu)\mathbf{f}_0 : \mathbb{R}^{n+1} \to \mathbb{R}^n$, where $\mathbf{f}_0$ is a function with a trivial solution. In our study we choose $\mathbf{f_0}$ as all the elements of Γ_0 and Σ_0, where Γ_0 and Σ_0 are the initial guesses for the irreducible vertices (bare vertices) and the self-energy (second-order perturbation result), respectively.

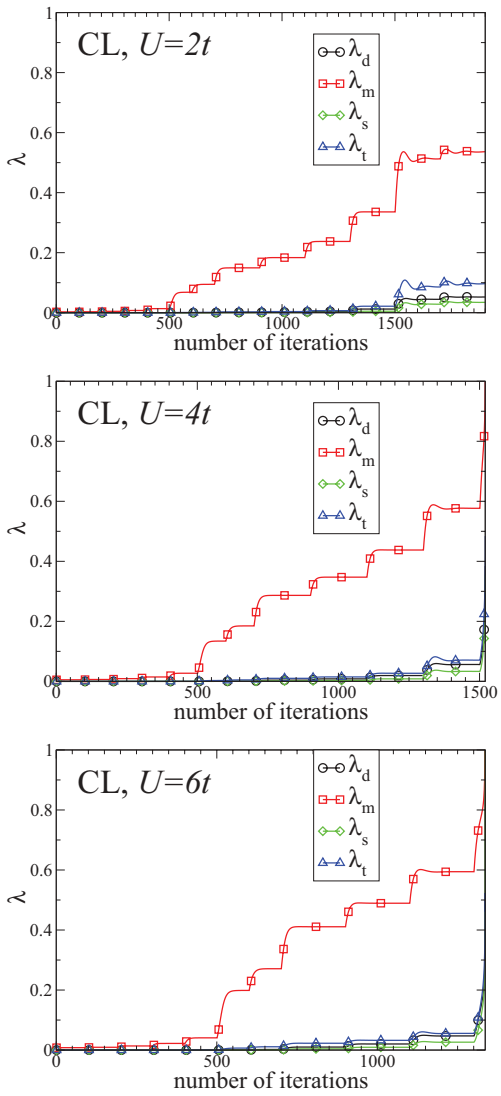

FIG. 7. (Color online) The leading eigenvalues obtained by the continuous loading (CL) method. (See Sec. III A for details.) Symbolically we write the parquet equations as $\mathbf{f}_{\text{parquet}}(\{\Sigma,\Gamma\}) = \{\Sigma,\Gamma\}$ and use them to define the auxiliary function as $\mathbf{g} = \nu\mathbf{f}_{\text{parquet}} + (1-\nu)\mathbf{f}_0$, where $\mathbf{f}_0$ is a function with a trivial solution. We choose $\mathbf{f}_0$ as a vector containing all the elements of Γ_0 and Σ_0. The iteration method is used to solve the function $\mathbf{g}(\nu)$, instead of $\mathbf{f}_{\text{parquet}}$. The solution of $\mathbf{f}_{\text{parquet}}$ is recovered when $\nu = 1$. A series of ν values are needed, which we denote as ν_i. The function $\mathbf{g}(\nu_i)$ is solved by the simple iteration method with the initial conditions given by the converged solution of the function $\mathbf{g}(\nu_{i-1})$. For $U = 2t$, we can push the value of ν to 1 to obtain the converged solution for $g(\nu = 1)$, and the solution of the parquet equations is recovered. However, the iteration procedure diverges for the cases of $U = 4t$ and $U = 6t$; they diverge at $\nu = 0.9999$ and 0.999, respectively. These examples show that for the cases where the simple iteration method diverges, the continuous loading method may not eliminate the divergence, even though the value of ν is pushed fairly close to 1.

We plot the leading eigenvalues from the continuous loading method in Fig. 7. The values of ν used are $\nu =$ 0.0, 0.5, 0.8, 0.9, 0.95, 0.99, 0.993, 0.996, 0.997, 0.998, 0.999, 0.9999, 1.0. 100 and 200 iterations are performed for $\nu \leqslant 0.993$ and $\nu \geqslant 0.996$, respectively. For $U = 2t$, a converged solution is obtained for $\nu = 1$. However, for $U = 4t$ and $6t$, the iterative solution diverges. The divergences appear before the homotopy parameter is pushed to $\nu = 1$. These examples illustrate the generic behavior when the parquet equations are solved by the simple iteration method beyond weak coupling. They also illustrate that the continuous loading method may not be sufficiently robust to solve the problem. The damping factor α used in these calculations is 0.1, which we believe is a fairly small value, although it may still not be sufficient. A rule of thumb for choosing the damping parameter is that it should be close to the value of the inverse of the residual between two consecutive iterations; unfortunately, with the huge number of variables, this choice will result in a very small step and may not be a practical option [58].

IV. CROSSING SYMMETRY VIOLATION

The exact solution of the parquet equations automatically satisfies crossing symmetry. It is one of the most important differences between the parquet formulation and most other perturbative methods. However, within the iteration scheme presented in the last section, the crossing symmetry is not satisfied unless the iteration converges to an exact solution. From the above section, we clearly find that the iteration method, even with the help of the continuous loading scheme, is not robust enough to obtain a converged solution beyond weak coupling. It is desirable to quantify the violation of crossing symmetry. The following six equalities are the consequence of crossing symmetry [see Fig. 8 for a diagrammatic representation of these crossing symmetry (CS) operations] [19,24]:

$$L_1 \equiv F_t(Q)_{P,P'} = [-(1/2)F_m - (1/2)F_d](P + P' + Q)_{-P',-P} \equiv R_1, \tag{38}$$

$$L_2 \equiv F_s(Q)_{P,P'} = [-(3/2)F_m + (1/2)F_d](P + P' + Q)_{-P',-P} \equiv R_2, \tag{39}$$

$$L_3 \equiv F_m(Q)_{P,P'} = [(1/2)F_t - (1/2)F_s](P + P' + Q)_{-P-Q,-P} \equiv R_3, \tag{40}$$

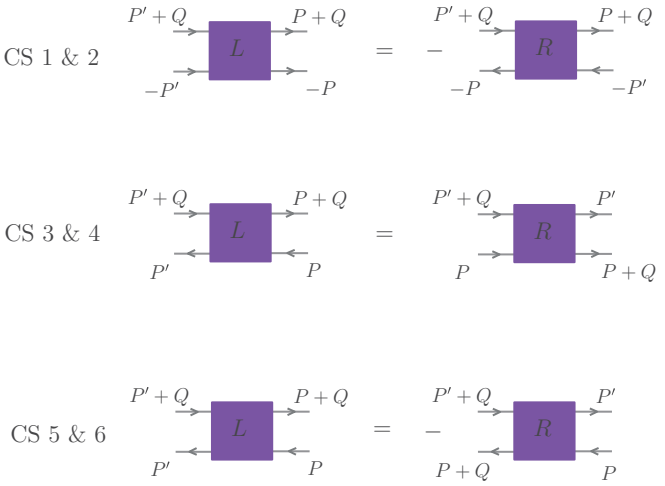

FIG. 8. (Color online) Diagrammatic representation of the six crossing symmetry operations. Note that spin indices are hidden for the purpose of clarity. For the first two operations (CSs 1 and 2), we exchange the lower two external legs. For the third and fourth operations (CSs 3 and 4), we exchange the lower two legs and then the right two legs. The first four crossing symmetry relationships (CSs 1–4) relate the particle-particle vertices to the particle-hole vertices. The last two (CSs 5 and 6) are for the particle-hole vertices only, where we exchange the lower left and upper right legs.

$$L_4 \equiv F_d(Q)_{P,P'} = [(3/2)F_t + (1/2)F_s](P + P' + Q)_{-P-Q,-P} \equiv R_4, \quad (41)$$

$$L_5 \equiv F_m(Q)_{P,P'} = [(1/2)F_m - (1/2)F_d](P' - P)_{P,P+Q} \equiv R_5, \quad (42)$$

$$L_6 \equiv F_d(Q)_{P,P'} = [-(3/2)F_m + (1/2)F_d](P' - P)_{P,P+Q} \equiv R_6. \quad (43)$$

In Fig. 9 we plot the violation of crossing symmetry versus the number of iterations. It can be seen clearly that the crossing symmetry cannot be perfectly restored; even for the case of $U = 2t$, the measures of violation of crossing symmetries show oscillatory decreasing behavior, and the rate of decrease is quite slow even though the leading eigenvalues seem to be well converged. Obviously, for the cases of $U = 4t$ and $6t$ the crossing symmetry is severely violated and at the verge of the divergence there are sharp increases in the crossing symmetry violation. This may suggest that if the crossing symmetry can be restored, the divergence may be avoided beyond weak coupling.

In Fig. 10, we plot the crossing symmetry violation as a function of iteration steps with the continuous loading method and the same parameters as in Fig. 7. It is important to note that the homotopy function $g(\nu)$ does not respect the crossing symmetry except at $\nu = 1$. Therefore, although the solution is converged, as long as $\nu \neq 1$, the crossing symmetry is violated. The data for $U = 2t, 4t$, and $6t$ are shown in the top, the middle, and the bottom panels, respectively. The data for $U = 2t$ show a peak at the beginning of the iteration procedure when ν is increased and gradually converge to a finite value. For ν close to 1, the data show a similar oscillatory behavior as that from the simple iteration method. Similar behaviors are also observed for $U = 4t$ and $U = 6t$; however, the iterations fail to converge for ν close to 1; the crossing symmetry is strongly violated. Just as in the simple iteration method, at the verge of the divergence, there are sharp increases of the violation of crossing symmetry.

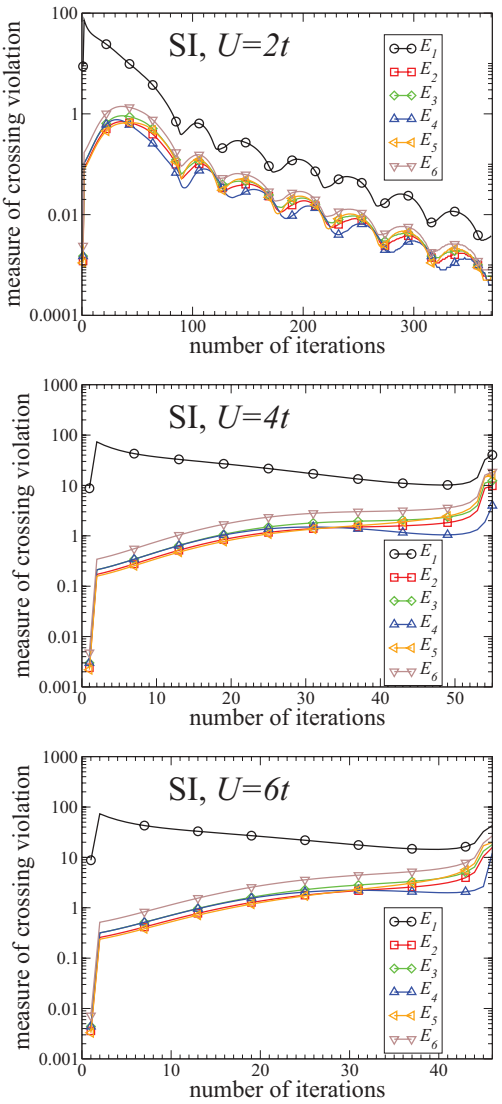

FIG. 9. (Color online) Crossing symmetry violation E_i versus the number of iterations for the simple iteration (SI) method with the same parameters as in Fig. 6. The six measures of crossing symmetry violation are defined as $E_i = |L_i - R_i|/|L_i + R_i|$, where $i = 1,2,\ldots,6$; L_i and R_i are defined respectively as the left- and the right-hand sides of Eqs. (38)–(43). The data for $U = 2t$ show an oscillatory but decreasing trend. This is expected for the case where the iteration provides a well-converged solution. One should note that although the leading eigenvalues seem to be converged, the crossing symmetry is not perfectly constructed from the iteration. For $U = 4t$ and $U = 6t$, the iteration fails to provide converged solutions, and the crossing symmetry is strongly violated. In particular, at the verge of the divergence, there is a sharp increase of the violation of crossing symmetry.

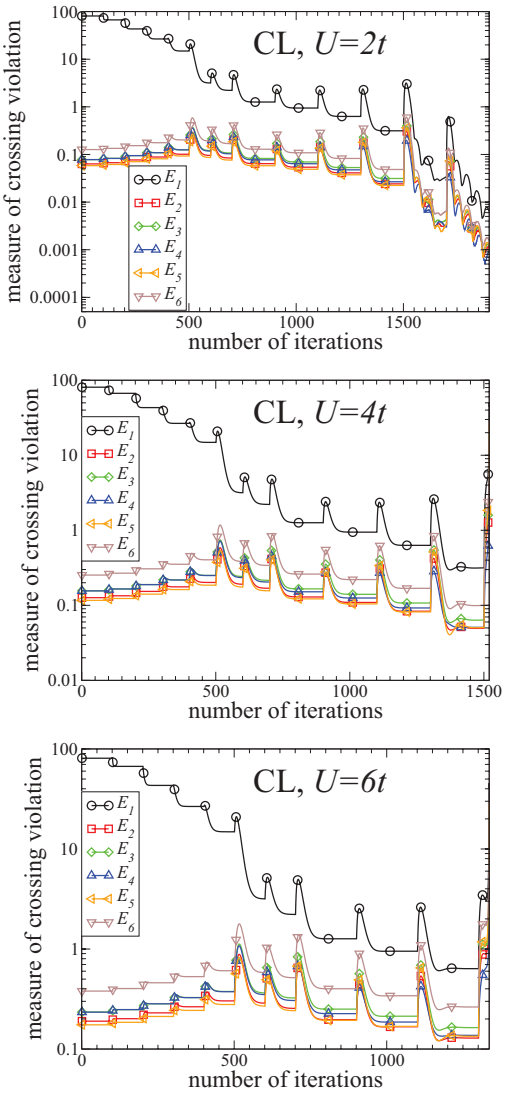

FIG. 10. (Color online) Crossing symmetry violation E_i versus the number of iterations for the continuous loading (CL) method with the same parameters and the same definition of E_i as in Fig. 9. The homotopy function $g(\nu)$ does not respect the crossing symmetry except at $\nu = 1$. Therefore, although the solution may be converged for some values of ν, as long as $\nu \neq 1$, the violation of crossing symmetry is nonzero. For $U = 2t$ the crossing symmetry violations peak near the beginning of the iteration procedure where ν is small and gradually converge to a finite value when $\nu = 1$. Similar behaviors are also observed for $U = 4t$ and $U = 6t$; however, since the iteration fails to converge for ν close to 1, the crossing symmetry is strongly violated. Similar to that observed in the simple iteration method, at the verge of the divergence, there is a sharp increase of the violation of crossing symmetry.

V. SYMMETRY RESTORATION

A. Crossing symmetry

Although it does not seem to be easy to analyze all the causes of the instability in the iteration, based on the discussion in the above section, our conjecture is that one of the possible reasons for the instability is that certain symmetries are violated in the course of the iterative process. A possible strategy to improve the iteration scheme is to impose those symmetries explicitly in the iteration process, so that at each step of the iteration these symmetries are not violated. The full vertices F obtained by the solution of the Bethe-Salpeter equation cannot guarantee crossing symmetry, unless the exact solution is attained. Therefore, so as to preserve crossing symmetry, the simplest method is to use the full vertices obtained by solving the Bethe-Salpeter equation (that is, the full vertices obtained from step 3 of the algorithm presented in Sec. III), and feed them back into the parquet equation to reconstruct the crossing-symmetric full vertices. Figure 11 illustrates the flow diagram for solving the parquet equations with explicit restoration of the crossing symmetry in the full vertices. Here is the algorithm which explicitly preserves the crossing symmetry:

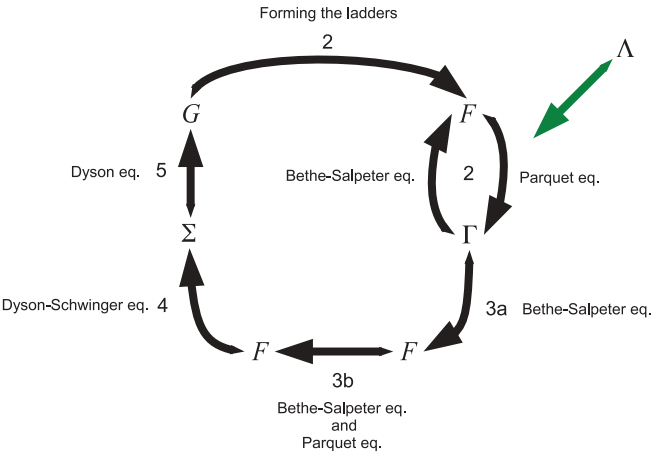

FIG. 11. (Color online) Flow diagram of the algorithm for solving the parquet equations with crossing symmetry restoration. See the text for the description of each step. The main difference compared to the previous algorithm is in step 3b where the crossing symmetry is restored explicitly in the full vertex F. Because of this explicit restoration of the crossing symmetry, in practice, step 2 is iterated only once as self-consistency is not required to generate the crossing symmetry.

(1) Set the initial conditions for the irreducible and full vertices, and the self-energy.

(2) Update the Green functions and calculate the bare susceptibility χ_0. Solve the parquet and the Bethe-Salpeter equations for the irreducible vertices Γ. Simple iteration is used until the convergence criteria are met for the irreducible vertices.

Since we will restore the crossing symmetry of the full vertices in the step 3b, we find that it is not necessary to attain the self-consistency for step 2. In practice, we iterate the parquet equations only once. The next step is to use the irreducible vertices obtained from the parquet equations to construct the full vertices.

(3a) Solve the Bethe-Salpeter equation to obtain the full vertices F, using the irreducible vertices from the previous step. This is executed exactly by calling the LAPACK routines for the inverse of the matrices [57].

(3b) Use the new irreducible vertices obtained in step 2 and the full vertices obtained in step 3a to form the vertex ladders. Construct the full vertices from Eqs. (22) and (23) using the

vertex ladders. Following these steps, the crossing symmetry is restored in the full vertex F.

With the full vertices obtained, we can update the self-energy.

(4) Solve the Dyson-Schwinger equation to obtain the self-energy from the full vertices. Simple iteration is used until the convergence criteria are met for the self-energy.

(5) Solve the Dyson equation for the fully dressed Green function from the self-energy.

This completes the iteration loop, and the procedure is repeated from step 2 until the criteria of convergence are met for both the self-energy and the irreducible vertices.

The main difference between the current algorithm and the previous algorithm we present in Sec. III is in step 3 where the full vertices are constructed. In the previous algorithm the Bethe-Salpeter equation is solved many times to attain convergence. When absolute convergence is attained, crossing symmetry will be satisfied. In the current algorithm, we just explicitly solve the Bethe-Salpeter equation in step 3a to refresh the full vertices. Once we obtain the full vertices, in step 3b, we construct the new vertex ladders and the new full vertices from the vertex ladders using the Bethe-Salpeter equation. By doing so, the crossing symmetry of the full vertices is satisfied; see Eqs. (30), (31), (32), and (33). In Fig. 12 we show the leading eigenvalues using the same set of parameters used in Fig. 6. While the simple iteration scheme without crossing symmetry fails to converge for the case of $U = 4$ and 6, it provides a converged solution when the crossing symmetry is explicitly restored.

B. Time-reversal and point group symmetry

Besides imposing crossing symmetry on the full vertices, some of the internal symmetries can also be imposed on the irreducible vertices and the self-energy without much computational overhead. We illustrate time-reversal symmetry in the self-energy

$$\Sigma(\mathbf{k}, i\omega) = \Sigma^*(\mathbf{k}, -i\omega) \tag{44}$$

and the vertices (spatial reflection symmetry and parity invariance are assumed)

$$F(Q)_{P,P'} = F(Q)_{P',P}. \tag{45}$$

As these symmetry operations do not mix vertices across different values of Q, and provided that the data is distributed with one or more Q at each node, the time-reversal symmetry of the vertices can be imposed without invoking communications across different nodes. Therefore, enforcing time-reversal symmetry will cause only a very minor computational overhead.

Other symmetries, such as the point group symmetry for the square lattice, can be rather cumbersome. An expensive scheme involving heavy internode communication would be required to impose the complete set of point group symmetries. However, we may impose an important subset of the operations R_α for which $R_\alpha(Q) = Q$ without expensive communications. In these cases, the vertices may be symmetrized by performing

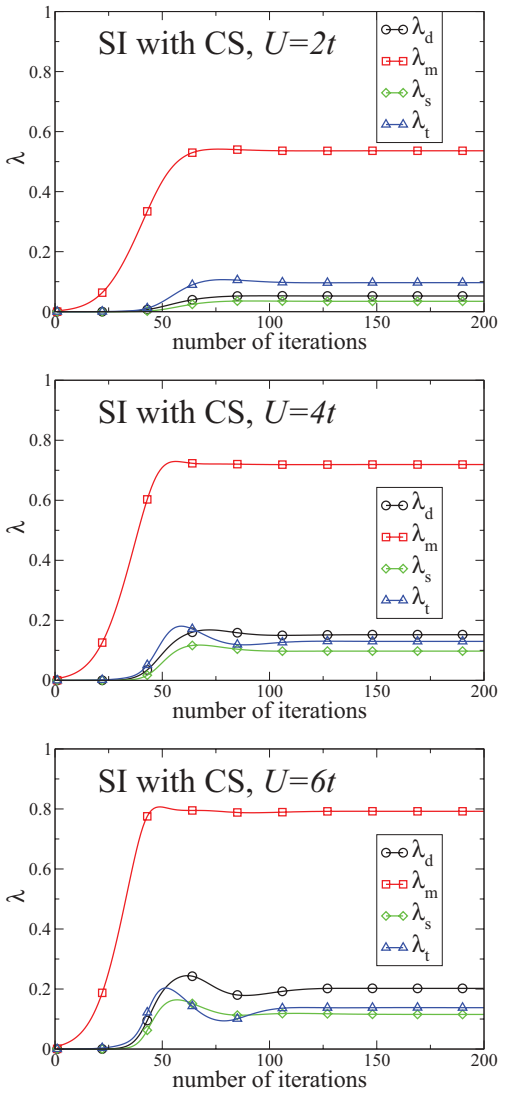

FIG. 12. (Color online) The leading eigenvalues of various channels [density (d), magnetic (m), spin singlet (s), and spin triplet (t)] versus the number of iterations with the simple interaction (SI) method with crossing symmetry (CS). The parameters used are the same as the data shown in Fig. 6. The only difference is that the crossing symmetry in the full vertex F is explicitly restored at each step of the iteration. This is easily achieved by constructing the full vertex directly from Eqs. (22) and (23). The simple iteration scheme without crossing symmetry fails for the case of $U = 4$ and 6. With the crossing symmetry explicitly restored, converged solutions are obtained.

the sum

$$F(Q)_{P,P'} = \frac{1}{N_{R_\alpha(Q)=Q}} \sum_{R_\alpha(Q)=Q} F(Q)_{R_\alpha(P),R_\alpha(P')}, \tag{46}$$

where $N_{R_\alpha(Q)=Q}$ is the number of elements in this subset of operations. For general Q in the cluster Brillouin zone there would be no α such that $R_\alpha(Q) = Q$ apart from the identity. However, for the points of high symmetry, $R_\alpha(Q) = Q$ for all α. Generally, the instabilities first occur here, so imposing the point group symmetries at these Q values should have the greatest impact.

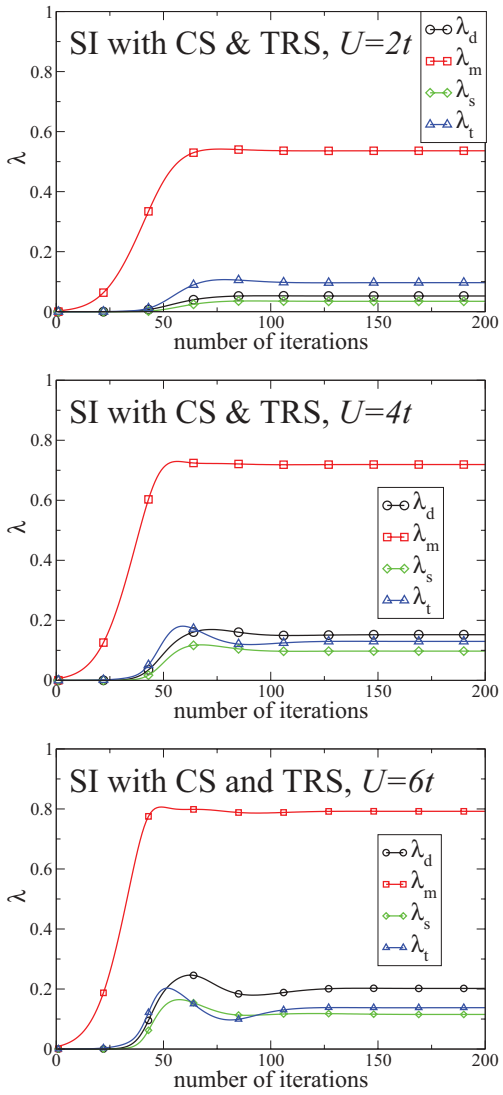

FIG. 13. (Color online) The leading eigenvalues of various channels [density (d), magnetic (m), spin singlet (s), and spin triplet (t)] as functions of the iteration. The parameters used are the same as for the data shown in Fig. 6. Two symmetries are explicitly restored at each step of the iteration: the crossing symmetry (CS) in the full vertex F and the time-reversal symmetry (TRS) for the self-energy Σ and both the irreducible vertex Γ and the full vertex F (that is $F(Q)_{P,P'} = F(Q)_{P',P}$ and similarly for Γ). Notice there is no substantial gain in the convergent rate compared to the case with only the crossing symmetry being restored.

In Fig. 13 we show the leading eigenvalues of various channels when both crossing and time-reversal symmetries are imposed for the same set of parameters that are used for the data in Figs. 6 and 12. Spatial reflection symmetry, parity invariance, and spin rotation invariance are assumed as appropriate for the two-dimensional Hubbard model at nonzero temperature. We can see that there is only very marginal improvement for the convergence compared to the results without explicit restoration of the time-reversal symmetry (see Fig. 12). We also used the scheme described above to partially impose the point group symmetries. However, these symmetries resulted in no additional improvements and therefore no results are shown.

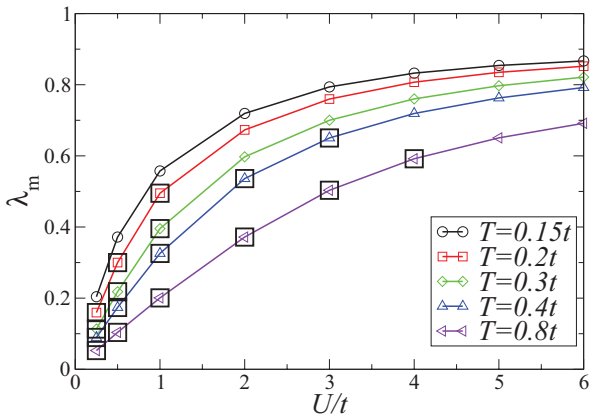

FIG. 14. (Color online) The leading eigenvalues of the antiferromagnetic magnetic channel λ_m as functions of the coupling U calculated with the simple iteration method. Different curves correspond to different temperatures. The data points enclosed in a black square correspond to the cases where the simple iteration without any symmetry restoration can provide a converged solution.

VI. LEADING EIGENVALUE OF THE ANTIFERROMAGNETIC CHANNEL

With the improved scheme proposed in this paper, we are able to explore a wider range of temperature and coupling strength for the half-filled Hubbard model. In Figure 14 we show the leading eigenvalue for the most singular channel, the antiferromagnetic channel λ_m as a function of U for a range of temperatures as low as $T = 0.15t$. The data points enclosed in a black square correspond to the cases where the simple iteration without any symmetry restoration provides a converged solution. For all temperatures, the λ_m values increase sharply at weak coupling ($U \sim 2t$), and they tend to saturate at strong coupling ($U \sim 6t$). They are most sensitive to temperature at intermediate coupling ($2t < U < 4t$). We emphasize that convergence is not possible without the improved scheme, unless a large number of iterations are used to attain the crossing symmetry.

VII. SUMMARY AND DISCUSSION

We present improvements of the numerical implementation of parquet equations for the Hubbard model. The main strategy is to enforce the symmetries in the iteration process. The most prominent advantage of the parquet formulation, compared to most approaches, is that the crossing symmetry is exactly fulfilled. However, in general, this is true only if the exact solution is found. With the simple iteration method, crossing symmetry is strongly violated prior to an instability, suggesting that the instability is due to these symmetry violations.

The continuous loading or homotopy method does not improve convergence significantly beyond the simple iteration method. We note that the solution of the continuous loading function does not preserve crossing symmetry. This may partly explain why the continuous loading method does not provide significant improvement over simple iteration.

We present a simple method to enforce the crossing symmetry at each step of the iteration which does not substantially increase the computational cost. The addition of these symmetry constraints can greatly improve the stability of the calculation, so that a wider range of parameters can be explored by the parquet formulation. Along this line of thought, one can expect that the stability may be further improved if other symmetries are also imposed, the obvious ones being time-reversal and point group symmetries. However, these additional symmetries did not improve the stability significantly beyond that obtained with crossing symmetry alone.

It has been suggested that the one-particle self-consistency cannot restore the three-peak structure of the spectral function for the single-impurity problem [12,13]. Therefore, in the future, it will be worthwhile to study in detail the effect of one-particle self-consistency for the lattice models. For the numerical implementation, we find that by keeping the Hartree propagator only, the iteration is generally more stable.

The main reason that the numerical solution of the parquet equations is limited to fairly modest system sizes is the memory allocation, and somewhat also computer cycles. In the current scheme the memory allocation scales with the third power of the number of sites (N_k) times the number of Matsubara frequencies (N_ω); this number ($N_k N_\omega$) is practically limited to around 3000 even in a supercomputer. However, we are currently investigating a scheme to compress the memory requirements by employing an inhomogeneous frequency grid. Once this is finished, we expect that we can study larger clusters.

The parquet formulation still remains one of the best approaches for calculating the two-particle vertex functions in a self-consistent manner. At present, solving the parquet equations for a large lattice size is still a very challenging task; however, with the continuous advances of computational facilities, it should become feasible in the foreseeable future. A promising direction, which allows immediate application of the parquet formulation, is to incorporate it as part of the multiscale many-body approach [59,60].

ACKNOWLEDGMENTS

We would like to acknowledge very useful discussions with Karen Tomko, and we thank Peter Reis for assistance. This work was supported in part by the DOE SciDAC Grant No. DE-FC02-06ER25792 (K.M.T., H.F., S.Y.Z., and M.J.) and the US National Science Foundation LA-SiGMA Grant No. EPS-1003897 (J.R., J.M., and M.J.). Supercomputer support was provided by the NSF TeraGrid under Grant No. TG-DMR100007. This research also used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

APPENDIX: PARALLEL IMPLEMENTATION WITH LATENCY HIDING

This Appendix describes a highly effective implementation of the symmetry-enforcing variant of the parquet formulation described earlier in the paper. The communication bottleneck in this implementation is caused by the expensive tensor rotations required to rotate the vertex ladders [Eqs. (24) and (25)] from the forms used in the Bethe-Salpeter equation, Eqs. (22) and (23), to those used in the parquet equations, Eqs. (26)–(29). If we distribute the equations between the processes executing on the compute nodes of a parallel machine using the transfer momenta Q, the tensor rotations are done with an expensive all-to-all communication among those processes, in which every node needs to communicate with all the other nodes. The message passing interface (MPI) implementation of the all-to-all communication [61] is a collective operation that is blocking, i.e., each process has to wait until the message has been sent out. The key aspect of our implementation is the decomposition of the required communication so that nonblocking communication primitives can be effectively utilized. The nonblocking communication enables latency hiding by overlapping computations and communications.

Four different forms of tensor rotation are required:

$$\text{rotation 1:}\Phi(Q)_{P,P'} \longleftarrow \Phi(P'-P)_{P,P+Q}, \tag{A1}$$

$$\text{rotation 2:}\Phi(Q)_{P,P'} \longleftarrow \Phi(P'-P)_{-P',P+Q}, \tag{A2}$$

$$\text{rotation 3:}\Phi(Q)_{P,P'} \longleftarrow \Phi(P+P'+Q)_{-P',-P}, \tag{A3}$$

$$\text{rotation 4:}\Phi(Q)_{P,P'} \longleftarrow \Phi(P+P'+Q)_{-P-Q,-P}. \tag{A4}$$

Note that the indices in subscripts and those in parentheses are equivalent, with the latter only distinguished by also labeling the nodes where data are distributed. The size of the tensors is $N_t \times N_t \times N_t$, where N_t is the number of momentum points times the number of discrete Matsubara frequencies, i.e., $N_t = N_k \times N_\omega$. All indices are in modulo arithmetic at each of the $D+1$ dimensions, where $D=2$ (the cluster dimension), and the "1" is for the Matsubara frequency. Because it takes many iterations (up to a few hundred for low temperatures and strong coupling) to obtain converged solutions, the total number of tensor rotations is significant and accounts for a large fraction of the computational time.

We use the hybrid MPI-OpenMP model for the computations. The rank-3 tensors are decomposed and evenly distributed into N virtual nodes. Each virtual node consists of a few cores. The size of a virtual node (i.e., the number of cores) is less than or equal to the size of a physical node. Specifically, we slice the rank-3 tensors to a set of two-dimensional arrays based on the index in parentheses, e.g., Q and $P-P'$ for the left and right sides of Eq. (A1), respectively. Then, each two-dimensional matrix is assigned to a virtual node. Since we have N_t layers of two-dimensional slices, the total number of virtual nodes also becomes $N = N_t$. In this scenario, every rotation requires data communications among all nodes. The following describes the data access patterns for our implementation of the tensor rotations.

Step 1. This step involves no MPI communication and is done before any data are sent between nodes. The tensor elements are locally rearranged in order to collect specific elements to be grouped and sent to designated destination nodes. The index in parentheses of the tensors on the right of Eqs. (A1)–(A4) represents the rank of a sending node in which a sliced two-dimensional matrix resides. For rotations 1 and 2,

the rank of the sending node S is

$$S = P' - P. \tag{A5}$$

For rotations 3 and 4, S is

$$S = P + P' + Q. \tag{A6}$$

Using Eqs. (A5) and (A6), and applying these to the corresponding rotations, the two-dimensional matrix elements are grouped based on the rank of the destination node Q from a given sending node S:

$$\text{rotation 1:} A_{P,Q} = \Phi(S)_{P,P+Q}, \tag{A7}$$

$$\text{rotation 2:} A_{P,Q} = \Phi(S)_{-(P+S),P+Q}, \tag{A8}$$

$$\text{rotation 3:} A_{P,Q} = \Phi(S)_{P+Q-S,-P}, \tag{A9}$$

$$\text{rotation 4:} A_{P,Q} = \Phi(S)_{-(P+Q),-P}. \tag{A10}$$

Note that, here, S is the node index (the index of the sender) and $P, Q \in \{0, \ldots, N_t - 1\}$, so the P and Q are the row and column indices of the matrix. We assume column-major order data access in MPI data communications which distribute columns of matrix **A** to nodes of rank Q in the next step.

Step 2. The columns of the two-dimensional matrix **A** are distributed among all nodes. At the sending nodes, each column of **A** is sent to a destination node labeled by Q. The standard approach is to use MPI_ALLTOALL. However, as we show later, this task can be done using different combinations of point-to-point communications [61]. In particular, nonblocking communication protocols can be applied to overlap communications and local computations. Overall, this procedure is applied to all the tensor rotations and can be written as

$$B_{P,S} \text{ at rank-}Q \text{ node: } \leftarrow A_{P,Q} \text{ at rank-}S \text{ node.} \tag{A11}$$

As shown in Eq. (A11), the rank of destination nodes is determined by the column index Q of **A** in sending nodes. The rank of the sending nodes becomes column index S of **B** in the receiving nodes. The rank of the sending nodes S must be provided to the receiving nodes in order to assign the correct column index to the received messages.

Step 3. Once messages have arrived at the destination nodes, the columns of the two-dimensional matrix **B** are rearranged to complete the tensor rotations. The column index of the rotated received matrix is related to the rank of the sending and receiving nodes by Eqs. (A5) and (A6).

Then, the rotations are finalized by using the following relations:

$$\text{rotation:1,2:} \Phi(Q)_{P,S+P} \longleftarrow B_{P,S}, \tag{A12}$$

$$\text{rotation:3,4:} \Phi(Q)_{P,S-(P+Q)} \longleftarrow B_{P,S}, \tag{A13}$$

where Q is the index of a given receiving node and $P, S \in \{0, \ldots, N_t - 1\}$. This step is a local process, i.e., no internode communication is necessary.

1. Improving the performance of tensor rotations

While steps 1 and 3 are strictly local processes, step 2 is the only stage involving nonlocal MPI communications. The nature of the collective communications among all nodes in step 2 makes it suited to the use of MPI_ALLTOALL. In such a case, step 2 can start only after the completion of step 1. Because MPI_ALLTOALL is a blocking communication, step 3 must wait to start until step 2 is finished. Therefore, the total elapsed time to complete a tensor rotation is the sum of elapsed times of the three steps. When the problem size is large, the communication efficiency of MPI_ALLTOALL is reduced significantly due to the increased network complexity associated with the bandwidth and latency among all participating nodes. Our approach to handling these rotations more efficiently is to implement a latency hiding strategy by overlapping message communications (step 2) and local computations (steps 1 and 3).

To enable this, we have developed our own version of a routine that performs communications from all nodes to all nodes. At a basic level, the functionality of this routine is identical to that of the generic MPI_ALLTOALL routine. However, our routine allows further data manipulations such that local computations are embedded between communications in the following way: On the sending node, the first column of **A** is computed from the equations of step 1. Then, MPI_ISEND sends out the first column of **A**. While this column is being sent out, the next column of **A** is prepared with step 1. This procedure is repeated until all N_t columns of **A**, the group of the selected elements from Φ, are sent out. This process overlaps steps 1 and 2. Latency hiding is also implemented in receiving nodes. We note that the sending nodes are also receiving nodes. They differ only in whether they are operating in the sending or the receiving *mode*. On the receiving nodes, MPI_IRECV is set to receive messages from arbitrary nodes by using MPI_ANY_SOURCE as a tag identifying the source of the message. For efficiency reasons, MPI_IRECV is posted before MPI_ISEND of the sending process. Then, MPI_TEST calls are used to check the completion of the arrival of the message. Once message arrival is confirmed, the rank of the node that sent this message can be identified by enquiring using MPI_STATUS. This provides S to assign to a corresponding column and to be used in step 3. Since the message arrival is column by column, the processing of each column of **B** continues to step 3 while the next column is traveling through the network. This process is repeated until all columns are completed. This procedure completely overlaps steps 2 and 3.

Depending on the size of problem, it is desirable to define a virtual node containing several cores (assuming multicore hardware architecture) based on the memory availability per node. Among the cores, MPI communications are assigned to one core. The other cores are utilized by implementing OpenMP [62] which parallelizes the local computational tasks in a node to all cores within the node. Thus, OpenMP thread depth is set to match with the total number of cores per virtual node. Specifically, we applied the DO directive of OpenMP for iterations of index P in the column selection processes of steps 1 and 3.

2. Experimental results

We test the efficiency of this latency hiding scheme using a nonblocking protocol against the standard MPI_ALLTOALL. All the experimental comparisons are conducted on the Cray XT5 (Jaguar) at the National Center for Computational

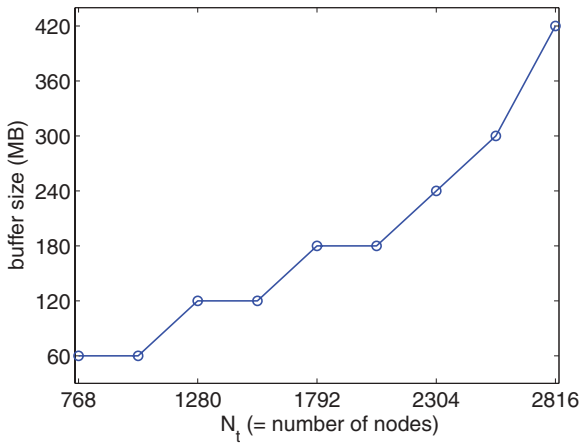

FIG. 15. (Color online) Required minimum buffer size to execute our all-to-all routine; each node has 12 cores.

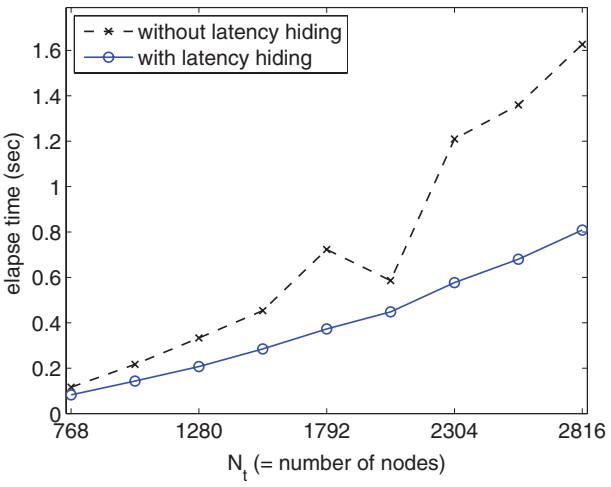

FIG. 16. (Color online) Time spent in data communication as a function of the number of computer nodes (12 processors per node). For large data sets each process sends messages to all the others, and the communication time scales linearly with the number of processes. Latency hiding techniques that overlap the interprocessor communication with local computations yields a factor of 2 speedup when compared with the standard MPI_ALLTOALL implementations as the number of processors increases beyond 30 000.

Sciences (NCCS) at the Oak Ridge National Laboratory. Jaguar consists of 12 cores per node, with six cores per nonuniform memory access (NUMA) node, and two NUMAs per node. First, we discuss hardware-driven constraints in implementing latency hiding. The nonblocking MPI_ISEND does not check for the arrivals of messages. With larger tensor size, the node usage and the size of individual columns becomes large. The MPI_ISEND from all participating nodes tries to dump a large column in each iteration. The next iteration starts regardless of message arrivals in the receiving nodes. As a consequence, a large amount of data rushes onto the network faster than the data can be absorbed by the receiving nodes. Eventually, this causes memory overflow to the system buffer assigned to the message processing unit. To avoid this we have allocated more memory space to the system buffer.

For simplicity, we assign one virtual node to a physical node. On the Jaguar Cray XT5, this means one virtual node containing 12 cores. To utilize all cores in a node, the value of OpenMP thread depth is set to 12. We gradually increase the problem size N_t until jobs end with error indicating buffer overflow. Then, we set a higher buffer size by controlling the environmental variable MPICH_UNEX_BUFFER_SIZE. For every incidence of error, we add 60 Mbytes buffer size. The default value of MPICH_UNEX_BUFFER_SIZE is 60 Mbytes on JAGUAR XT5 (the total number of cores is less than 50 000). The results are shown in Fig. 15. Up to $N_t = 1024$, the 60 Mbytes default buffer size is enough to handle the data traffic. Increasing N_t further forces us to use a larger buffer size. Overall, the amount of added buffer size increases for larger problem sizes. We note that the results presented in Fig. 15 are with the maximum number of cores per virtual node. Smaller core usage per node alleviates the buffer restriction. For example, hardware setup with a NUMA node per virtual node consumes less buffer memory due to the reduced total number of physical nodes participating in internode communication. We did not observe buffer memory overflow with the generic blocking MPI_ALLTOALL routine.

The performance of the latency hiding approach is evaluated in terms of wall-clock time spent on a single tensor rotation and compared with the case of the standard MPI_ALLTOALL applied for step 2. For this, the elapsed time to complete the tensor rotation is averaged over nine independent runs. Each run contains 40 repetitions of identical tensor rotations. At the end of each run, the elapsed time is also averaged for the 40 rotations. For all runs, we choose rotation 1 and the minimum buffer sizes shown in Fig. 15 are assumed. The comparison results are shown in Fig. 16. Except for $N_t = 768$ and 2048, latency hiding outperforms the case without latency hiding by a significant amount. The performance differences are even higher for $N_t \geqslant 2304$. For the MPI_ALLTOALL case, there is a sudden speedup at $N_t = 2048$. We are exploring this behavior further. We believe that it is caused by changes in the data traffic controlled by the hardware.

Overall, latency hiding provides a higher speedup for larger tensor sizes and core count. From the general trends, it can be expected that twofold or more efficiency improvement for N_t greater than 2816 can be obtained by implementing latency hiding with our nonblocking adaptation of the all-to-all routine for tensor rotations.

[1] L. D. Landau, A. A. Abrikosov, and I. M. Khalatnikov, Dokl. Akad. Nauk. **95**, 497 (1954); **95**, 773 (1954); **95**, 1177 (1954).
[2] I. Ya. Pomeranchuk, V. V. Sudakov, and K. A. Ter-Martirosyan, Phys. Rev. **103**, 784 (1956).
[3] K. Ter-Martirosyan, Phys. Rev. **111**, 948 (1958).
[4] C. de Dominicis and P. C. Martin, J. Math. Phys. **5**, 14 (1964).
[5] C. de Dominicis and P. C. Martin, J. Math. Phys. **5**, 31 (1964).

[6] B. Roulet, J. Gavoret, and P. Nozières, Phys. Rev. **178**, 1072 (1969).
[7] V. M. Yakovenko, Phys. Rev. B **47**, 8851 (1993).
[8] S. A. Brazovskii, Sov. Phys. JETP **34**, 1286 (1972).
[9] S. A. Brazovskii, Sov. Phys. JETP **35**, 433 (1972).
[10] P. Kleinert and H. Schlegel, Physica A **218**, 507 (1995).
[11] C.-X. Chen and N. E. Bickers, Solid State Commun. **82**, 311 (1992).
[12] V. Janiš and P. Augustinský, Phys. Rev. B **75**, 165108 (2007).
[13] V. Janiš and P. Augustinský, Phys. Rev. B **77**, 085106 (2008).
[14] P. Augustinský and V. Janiš, Phys. Rev. B **83**, 035114 (2011).
[15] V. Janiš and J. Kolorenč, Phys. Rev. B **71**, 033103 (2005).
[16] V. Janiš, Phys. Rev. B **64**, 115115 (2001).
[17] V. Janiš, J. Phys.: Condens. Matter **21**, 485501 (2009).
[18] I. E. Dzyaloshinskii and V. M. Yakovenko, Int. J. Mod. Phys. B **2**, 667 (1988).
[19] N. E. Bickers, in *Numerical Methods for Lattice Quantum Many-Body Problems*, edited by D. J. Scalapino (Addison-Wesley, New York, 1998).
[20] N. E. Bickers and S. R. White, Phys. Rev. B **43**, 8044 (1991).
[21] D. W. Hess, J. J. Deisz, and J. W. Serene, Philos. Mag. **74**, 457 (1996).
[22] J. Luo and N. E. Bickers, Phys. Rev. B **48**, 15983 (1993).
[23] V. Janiš, Phys. Rev. B **60**, 11345 (1999).
[24] H. Kusunose, J. Phys. Soc. Jpn. **79**, 094707 (2010).
[25] A. D. Jackson, A. Lande, R. W. Guitink, and R. A. Smith, Phys. Rev. B **31**, 403 (1985).
[26] A. D. Jackson and R. A. Smith, Phys. Rev. A **36**, 2517 (1987).
[27] M. Pfitzner and P. Wölfle, Phys. Rev. B **35**, 4699 (1987).
[28] R. A. Weiner, Phys. Rev. Lett. **24**, 1071 (1970).
[29] R. A. Weiner, Phys. Rev. B **4**, 3165 (1971).
[30] J. Yeo and M. A. Moore, Phys. Rev. Lett. **76**, 1142 (1996).
[31] J. Yeo and M. A. Moore, Phys. Rev. B **54**, 4218 (1996).
[32] J. Yeo and M. A. Moore, Phys. Rev. B **64**, 024514 (2001).
[33] J. Yeo, H. Park, and S. Yi, J. Phys.: Condens. Matter **18**, 3607 (2006).
[34] A. Shishanin and I. Ziyatdinov, J. High Energy Phys. **07**, 32 (2003).
[35] I. Ya. Aref'eva and A. P. ZubarevPhys. Lett. B **386**, 258 (1996).
[36] E. Bergli and M. Hjorth-Jensen, Ann. Phys. (NY) **326**, 1125 (2011).
[37] S. X. Yang, H. Fotso, J. Liu, T. A. Maier, K. Tomko, E. F. D'Azevedo, R. T. Scalettar, T. Pruschke, and M. Jarrell, Phys. Rev. E **80**, 046706 (2009).
[38] N. E. Bickers (unpublished).
[39] D. Bohm and D. Pines, Phys. Rev. **92**, 609 (1953).
[40] D. Pines, Phys. Rev. **92**, 626 (1953).
[41] T. Moriya, *Spin Fluctuations in Itinerant Electron Magnetism* (Springer, Berlin, 1985).
[42] N. E. Bickers and D. J. Scalapino, Ann. Phys. (NY) **193**, 106 (1989).
[43] S. Weinberg, *The Quantum Theory of Fields* (Cambridge University Press, Cambridge, 2005), Vol. 1.
[44] P. Kleinert, Prog. Theor. Phys. **123**, 327 (2009).
[45] N. E. Bickers and D. J. Scalapino, Phys. Rev. B **46**, 8050 (1992).
[46] H. A. Bethe and E. E. Salpeter, Phys. Rev. **82**, 309 (1951).
[47] H. A. Bethe and E. E. Salpeter, Phys. Rev. **84**, 1232 (1951).
[48] J. W. Negele and H. Orland, *Quantum Many-Particle Systems* (Perseus Books, New York, 1988).
[49] A. L. Fetter and J. D. Walecka, *Quantum Theory of Many-Particle Systems* (McGraw-Hill, New York, 1971).
[50] L. D. Landau, Sov. Phys. JETP **3**, 920 (1957).
[51] L. D. Landau, Sov. Phys. JETP **5**, 101 (1957).
[52] L. D. Landau, Sov. Phys. JETP **8**, 70 (1958).
[53] A. A. Abrikosov, L. P. Gor'kov, and I. Ye. Dzyaloshinskii, *Quantum Field Theoretical Methods in Statistical Physics* (Pergamon Press, Oxford, 1965).
[54] N. D. Mermin, Phys. Rev. **159**, 161 (1967).
[55] F. J. Dyson, Phys. Rev. **75**, 1736 (1949).
[56] J. Schwinger, Proc. Natl, Acad. Sci. USA **37**, 452 (1951).
[57] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. (Society for Industrial and Applied Mathematics, Philadelphia, 1999).
[58] R. E. Bank and D. J. Rose, Numer. Math. **37**, 279 (1981).
[59] M. Jarrell, K. Tomko, Th. Maier, E. D'Azevedo, R. T. Scalettar, Z. Bai, and S. Savrasov, J. Phys.: Conf. Ser. **78**, 012031 (2007).
[60] C. Slezak, M. Jarrell, Th. Maier, and J. Deisz, J. Phys.: Condens. Matter **21**, 435604 (2009).
[61] M. Snir, J. Dongarra, J. Kowalik, S. Hauss-lederman, S. Otto, and D. Walker, *MPI: The Complete Reference* (MIT Press, Cambridge, MA, 1998).
[62] *OpenMP Application Program Interface*, Version 3.0, May 2008.