*i*QIST: An open source continuous-time quantum Monte Carlo impurity solver toolkit[☆]

Li Huang [a,*], Yilin Wang [b], Zi Yang Meng [b,c], Liang Du [d], Philipp Werner [a], Xi Dai [b]

[a] *Department of Physics, University of Fribourg, 1700 Fribourg, Switzerland*
[b] *Beijing National Laboratory for Condensed Matter Physics, and Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China*
[c] *Department of Physics, University of Toronto, Toronto, Ontario M5S 1A7, Canada*
[d] *Department of Physics, The University of Texas at Austin, Austin, TX 78712, USA*

Quantum impurity solvers have a broad range of applications in theoretical studies of strongly correlated electron systems. Especially, they play a key role in dynamical mean-field theory calculations of correlated lattice models and realistic materials. Therefore, the development and implementation of efficient quantum impurity solvers is an important task. In this paper, we present an open source interacting quantum impurity solver toolkit (dubbed *i*QIST). This package contains several highly optimized quantum impurity solvers which are based on the hybridization expansion continuous-time quantum Monte Carlo algorithm, as well as some essential pre- and post-processing tools. We first introduce the basic principle of continuous-time quantum Monte Carlo algorithm and then discuss the implementation details and optimization strategies. The software framework, major features, and installation procedure for *i*QIST are also explained. Finally, several simple tutorials are presented in order to demonstrate the usage and power of *i*QIST.

Program summary

*Program title: i*QIST
Catalogue identifier: AEWQ_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEWQ_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: GNU General Public License, version 3
No. of lines in distributed program, including test data, etc.: 226270
No. of bytes in distributed program, including test data, etc.: 5263144
Distribution format: tar.gz
Programming language: Fortran 2008 and Python.
Computer: Desktop PC, laptop, high performance computing cluster.
Operating system: Unix, Linux, Mac OS X, Windows.
Has the code been vectorized or parallelized?: Yes, it is parallelized by MPI and OpenMP
RAM: Depends on the complexity of the problem
Classification: 7.3.
External routines: BLAS, LAPACK, Latex is required to build the user manual.
Nature of problem:
Quantum impurity models were originally proposed to describe magnetic impurities in metallic hosts. In these models, the Coulomb interaction acts between electrons occupying the orbitals of the impurity atom. Electrons can hop between the impurity and the host, and in an action formulation, this hopping is described by a time-dependent hybridization function. Nowadays quantum impurity models have a broad range of applications, from the description of heavy fermion systems, and Kondo insulators, to

---

quantum dots in nano-science. They also play an important role as auxiliary problems in dynamical mean-field theory and its diagrammatic extensions [1–3], where an interacting lattice model is mapped onto a quantum impurity model in a self-consistent manner. Thus, the accurate and efficient solution of quantum impurity models becomes an essential task.

Solution method:

The quantum impurity model can be solved by the numerically exact continuous-time quantum Monte Carlo method, which is the most efficient and powerful impurity solver for finite temperature simulations. In the *i*QIST software package, we implemented the hybridization expansion version of continuous-time quantum Monte Carlo algorithm. Both the segment representation and general matrix formalism are supported. The key idea of this algorithm is to expand the partition function diagrammatically in powers of the impurity-bath hybridization, and to stochastically sample these diagrams to all relevant orders using the Metropolis Monte Carlo algorithm. For a detailed review of the continuous-time quantum Monte Carlo algorithms, please refer to [4].

Running time:

Depends on the complexity of the problem. The sample run supplied in the distribution takes about 1.5 minutes.

References:

[1] A. Georges, G. Kotliar, W. Krauth, and M. J. Rozenberg, Rev. Mod. Phys. 68, 13 (1996)
[2] G. Kotliar, S. Y. Savrasov, K. Haule, V. S. Oudovenko, O. Parcollet, and C. A. Marianetti, Rev. Mod. Phys. 78, 865 (2006)
[3] T. Maier, M. Jarrell, T. Pruschke, and M. H. Hettler, Rev. Mod. Phys. 77, 1027 (2005)
[4] E. Gull, A. J. Millis, A. I. Lichtenstein, A. N. Rubtsov, M. Troyer, and Philipp Werner, Rev. Mod. Phys. 83, 349 (2011)

1. Introduction

In this paper we present *i*QIST (abbreviation for 'interacting quantum impurity solver toolkit'), an open source project for recently developed hybridization expansion continuous-time quantum Monte Carlo impurity solvers [1] and corresponding pre- and post-processing tools.

Dynamical mean-field theory (DMFT) [2,3] and its cluster extensions [4] play an important role in contemporary studies of correlated electron systems. The broad applications of this technique range from the study of Mott–Hubbard metal–insulator transitions [5], unconventional superconductivity in Cu- and Fe-based superconductors [6–9], and non-Fermi liquid behaviors in multi-orbital systems [10–13], to the investigation of anomalous transport properties of transition metal oxides [14]. For many of these applications, DMFT is the currently most powerful and reliable (sometimes the only) technique available and has in many cases produced new physical insights. Furthermore, the combination of *ab initio* calculation methods (such as density function theory) with DMFT [3] allows to capture the subtle electronic properties of realistic correlated materials, including those of partially filled 3d- and 4d-electron transition metal oxides, where lattice, spin and orbital degrees of freedom are coupled [14].

The key idea of DMFT is to map the original correlated lattice model onto a quantum impurity model whose mean-field bath is determined self-consistently [2–4]. Thus, the central task of a DMFT simulation becomes the numerical solution of a quantum impurity problem. During the past several decades, many methods have been developed and tested as impurity solvers, including the exact diagonalization (ED) [15], equation of motion (EOM) [16], Hubbard-I approximation (HIA) [17], iterative perturbation theory (IPT) [18], non-crossing approximation (NCA) [19], fluctuation-exchange approximation (FLEX) [20], and quantum Monte Carlo (QMC) [21,22], etc. Among the methods listed above, the QMC method has several very important advantages, which makes it so far the most flexible and widely used impurity solver. First, it is based on the imaginary time action, in which the infinite bath has been integrated out. Second, it can treat arbitrary couplings, and can thus be applied to all kinds of phases including the metallic phase, insulating state, and phases with spontaneous symmetry breaking. Third, the QMC method is numerically exact with a "controlled" numerical error. In other words, by increasing the computational effort the numerical error of the QMC simulation can be systematically reduced. For these reasons, the QMC algorithm is considered as the method of choice for many applications.

Several QMC impurity solvers have been developed in the past three decades. An important innovation was the Hirsch–Fye QMC (HF-QMC) impurity solver [21,22], in which the time axis is divided into small time steps and the interaction term in the Hamiltonian is decoupled on each time step by means of a discrete Hubbard–Stratonovich auxiliary field. HF-QMC has been widely used in the DMFT context [2–4], but is limited by the discretization on the time axis and also by the form of the electronic interactions (usually only density–density interactions can be efficiently treated). Recently, a new class of more powerful and versatile QMC impurity solvers, continuous-time quantum Monte Carlo (CT-QMC) algorithms, have been invented [1,23–27]. In the CT-QMC impurity solvers, the partition function of the quantum impurity problem is diagrammatically expanded, and then the diagrammatic expansion series is evaluated by stochastic Monte Carlo sampling. The continuous-time nature of the algorithm means that operators can be placed at any arbitrary position on the imaginary time interval, so that time discretization errors can be completely avoided. Depending on how the diagrammatic expansion is performed, the CT-QMC approach can be further divided into interaction expansion (or weak coupling) CT-QMC (CT-INT) [23], auxiliary field CT-QMC (CT-AUX) [24], and hybridization expansion (or strong coupling) CT-QMC (CT-HYB) [25–27].

At present, CT-HYB is the most popular and powerful impurity solver, since it can be used to solve multi-orbital impurity models with general interactions at low temperature [1]. In single-site DMFT calculations, the computational efficiency of CT-HYB is much higher than that of CT-INT, CT-AUX, and HF-QMC, especially when the interactions are intermediate or strong. However, in order to solve more complicated quantum impurity models (for example, five-band or seven-band impurity model with general interactions and spin–orbital

coupling) efficiently, further improvements of the CT-HYB impurity solvers are needed. In recent years many tricks and optimizations have been explored and implemented to increase the efficiency and accuracy of the original CT-HYB algorithm, such as the truncation approximation [27], Krylov subspace iteration [28], orthogonal polynomial representation [29–31], PS quantum number [32], lazy trace evaluation [33], skip-list technique [33], matrix product state implementation [34], and sliding window sampling scheme [34]. As the state-of-the-art CT-HYB impurity solvers become more and more sophisticated and specialized, it is not easy anymore to master all their facets and build one's implementations from scratch. Hence, we believe that it is a good time to provide a CT-HYB software package for the DMFT community such that researchers can focus more on the physics problems, instead of spending much time on (re-)implementing in-house codes. In fact, there are some valuable efforts in this direction, such as TRIQS [35], ALPS [36,37], W2DYNAMICS [32], and DMFT_W2K [27,38]. The present implementation of the CT-HYB impurity solvers is a useful complement to the existing codes. The open source *i*QIST software package contains several well-implemented and thoroughly tested modern CT-HYB impurity solvers, and the corresponding pre- and post-processing tools. We hope the release of *i*QIST can promote the quick development of this research field.

The rest of this paper is organized as follows: In Section 2, the basic theory of quantum impurity models, CT-QMC algorithms, and its hybridization expansion version are briefly introduced. The measurements of several important physical observables are presented. In Section 3, the implementation details of *i*QIST are discussed. Most of the optimization tricks and strategies implemented in *i*QIST, including dynamical truncation, lazy trace evaluation, sparse matrix technique, PS quantum number, and subspace algorithms, etc., are reviewed. These methods ensure the high efficiency of *i*QIST. In Section 4, we first present an overview on the software architecture and component framework. Then the main features of the *i*QIST software package, including the CT-HYB impurity solvers, the atomic eigenvalue solver, and the other auxiliary tools are presented. The compiling and installation procedures, and the basic usage of *i*QIST are introduced in Section 5. Section 6 shows several simple applications of *i*QIST, ranging from self-consistent single-site DMFT calculation to one-shot post-processing calculation. These examples serve as introductory tutorials. Finally, a short summary is given in Section 7 and the future development plans for the *i*QIST project are outlined as well.

2. Basic theory and methods

In this section, we will present the basic principles of CT-QMC impurity solvers, with an emphasis on the hybridization expansion technique. For detailed derivations and explanations, please refer to Ref. [1].

2.1. Quantum impurity model

The multi-orbital Anderson impurity model (AIM) can be written as $H_{\mathrm{imp}} = H_{\mathrm{loc}} + H_{\mathrm{bath}} + H_{\mathrm{hyb}}$, where

$$H_{\mathrm{loc}} = \sum_{\alpha\beta} E_{\alpha\beta} d_\alpha^\dagger d_\beta + \sum_{\alpha\beta\gamma\delta} U_{\alpha\beta\gamma\delta} d_\alpha^\dagger d_\beta^\dagger d_\gamma d_\delta, \tag{1a}$$

$$H_{\mathrm{hyb}} = \sum_{\mathbf{k}\alpha\beta} V_{\mathbf{k}}^{\alpha\beta} c_{\mathbf{k}\alpha}^\dagger d_\beta + h.c., \tag{1b}$$

$$H_{\mathrm{bath}} = \sum_{\mathbf{k}\alpha} \epsilon_{\mathbf{k}\alpha} c_{\mathbf{k}\alpha}^\dagger c_{\mathbf{k}\alpha}. \tag{1c}$$

In Eq. (1), Greek letters in the subscripts denote a combined spin–orbital index, the operator $d_\alpha^\dagger$ (d_α) is creating (annihilating) an electron with index α on the impurity site, while $c_{\mathbf{k}\alpha}^\dagger$ ($c_{\mathbf{k}\alpha}$) is the creation (annihilation) operator for conduction band (bath) electron with spin–orbital index α and momentum $\mathbf{k}$. The first term in H_{loc} is the general form of the impurity single particle term with impurity level splitting and inter-orbital hybridization. This term can be built by crystal field (CF) splitting and/or spin–orbit coupling (SOC), etc. The second term in H_{loc} is the Coulomb interaction term which can be parameterized by intra(inter)-band Coulomb interactions U (U') and Hund's rule coupling J or Slater integral parameters F^k. The hybridization term H_{hyb} describes the process of electrons hopping from the impurity site to the environment and back. H_{bath} describes the non-interacting bath. This Anderson impurity model is solved self-consistently in the DMFT calculations [2,3].

2.2. Principles of continuous-time quantum Monte Carlo algorithm

We first split the full Hamiltonian H_{imp} into two separate parts, $H_{\mathrm{imp}} = H_1 + H_2$, then treat H_2 as a perturbation term, and expand the partition function $\mathcal{Z} = \mathrm{Tr} e^{-\beta H}$ in powers of H_2,

$$\mathcal{Z} = \sum_{n=0}^{\infty} \int_0^\beta \cdots \int_{\tau_{n-1}}^\beta \omega(\mathcal{C}_n), \tag{2}$$

with

$$\omega(\mathcal{C}_n) = d\tau_1 \cdots d\tau_n \mathrm{Tr} \left\{ e^{-\beta H_1} [-H_2(\tau_n)] \cdots [-H_2(\tau_1)] \right\}, \tag{3}$$

where $H_2(\tau)$ is defined in the interaction picture with $H_2(\tau) = e^{\tau H_1} H_2 e^{-\tau H_1}$. Each term in Eq. (2) can be regarded as a diagram or configuration (labeled by $\mathcal{C}$), and $\omega(\mathcal{C}_n)$ is the diagrammatic weight of a specific order-n configuration. Next we use a stochastic Monte Carlo algorithm to sample the terms of this series. In the CT-INT and CT-AUX impurity solvers [23,24], the interaction term is the perturbation term, namely, $H_2 = H_{\mathrm{int}}$, while $H_2 = H_{\mathrm{hyb}}$ is chosen for the CT-HYB impurity solver [25]. In the intermediate and strong interaction region, CT-HYB is much more efficient than CT-INT and CT-AUX. This is also the main reason that we only implemented the CT-HYB impurity solvers in the *i*QIST software package.

2.3. Hybridization expansion

In the hybridization expansion algorithm, due to fact that H_1 does not mix the impurity and bath states, the trace in Eq. (3) can be written as $\mathrm{Tr} = \mathrm{Tr}_d \mathrm{Tr}_c$. As a result, we can split the weight of each configuration as

$$\omega(\mathcal{C}_n) = \omega_d(\mathcal{C}_n)\omega_c(\mathcal{C}_n) \prod_{i=1}^{n} d\tau_i. \tag{4}$$

$\omega_d(\mathcal{C}_n)$ is the trace over impurity operators (Tr_d), $\omega_c(\mathcal{C}_n)$ is the trace over bath operators (Tr_c). Further, since Wick's theorem is applicable for the $\omega_c(\mathcal{C}_n)$ part, we can represent it as a determinant of a matrix $\mathcal{Z}_{\mathrm{bath}}\mathcal{M}^{-1}$ with $\mathcal{Z}_{\mathrm{bath}} = \mathrm{Tr}_c e^{-\beta H_{\mathrm{bath}}}$ and $(\mathcal{M}^{-1})_{ij} = \Delta(\tau_i - \tau_j)$. The $\omega_d(\mathcal{C}_n)$ part can be expressed using the segment representation when $[n_\alpha, H_{\mathrm{loc}}] = 0$ [25]. However, if this condition is not fulfilled, we have to calculate the trace explicitly, which is called the general matrix algorithm [26,27]. The explicit calculation of the trace for a large multi-orbital AIM with general interactions is computationally expensive. Many tricks and strategies have been implemented in the *i*QIST software package to address this challenge. Please refer to Section 3 for more details.

In this package, we used importance sampling and the Metropolis algorithm to evaluate Eq. (2). The following four local update procedures, with which the ergodicity of Monte Carlo algorithm is guaranteed, are used to generate the Markov chain:

- Insert a pair of creation and annihilation operators in the time interval $[0, \beta)$.
- Remove a pair of creation and annihilation operators from the current configuration.
- Select a creation operator randomly and shift its position in the time interval $[0, \beta)$.
- Select an annihilation operator randomly and shift its position in the time interval $[0, \beta)$.

In the Monte Carlo simulations, sometimes the system can be trapped in some (for example symmetry-broken) state. In order to avoid unphysical trapping, we also consider the following two global updates:

- Swap the operators of randomly selected spin up and spin down flavors.
- Swap the creation and annihilation operators globally.

2.4. Physical observables

Many physical observables are measured in our CT-HYB impurity solvers. Here we provide a list of them.

Single-particle Green's function $G(\tau)$

The most important observable is the single-particle Green's function $G(\tau)$, which is measured using the elements of the matrix $\mathcal{M}$,

$$G(\tau) = \left\langle \frac{1}{\beta} \sum_{ij} \delta^-(\tau, \tau_i - \tau_j)\mathcal{M}_{ji} \right\rangle, \tag{5}$$

with

$$\delta^-(\tau, \tau') = \begin{cases} \delta(\tau - \tau'), & \tau' > 0, \\ -\delta(\tau - \tau' + \beta), & \tau' < 0. \end{cases} \tag{6}$$

Note that in the *i*QIST software package, the low-frequency Matsubara Green's function $G(i\omega_n)$ is also measured directly, instead of being calculated from $G(\tau)$ using Fourier transformation.

Two-particle correlation function $\chi_{\alpha\beta}(\tau_a, \tau_b, \tau_c, \tau_d)$

The two-particle correlation functions are often used to construct lattice susceptibilities within DMFT and diagrammatic extensions of DMFT. However, the measurements of two-particle correlation functions are a nontrivial task [39] as it is very time-consuming to obtain good quality data, and most of the previous publications in this field are restricted to measurements of two-particle correlation functions in one-band models. Thanks to the development of efficient CT-HYB algorithms, the calculation of two-particle correlation functions for multi-orbital impurity models now becomes affordable [29–31]. In the *i*QIST software package, we implemented the measurement for the two-particle correlation function $\chi_{\alpha\beta}(\tau_a, \tau_b, \tau_c, \tau_d)$, which is defined as follows:

$$\chi_{\alpha\beta}(\tau_a, \tau_b, \tau_c, \tau_d) = \langle c_\alpha(\tau_a)c_\alpha^\dagger(\tau_b)c_\beta(\tau_c)c_\beta^\dagger(\tau_d) \rangle. \tag{7}$$

Due to memory restrictions, the actual measurement is performed in frequency space, for which we use the following definition of the Fourier transform:

$$\chi_{\alpha\beta}(\omega, \omega', \nu) = \frac{1}{\beta} \int_0^\beta d\tau_a \int_0^\beta d\tau_b \int_0^\beta d\tau_c \int_0^\beta d\tau_d \chi_{\alpha\beta}(\tau_a, \tau_b, \tau_c, \tau_d)e^{i(\omega+\nu)\tau_a}e^{-i\omega\tau_b}e^{-i\omega'\tau_c}e^{-i(\omega'+\nu)\tau_d} \tag{8}$$

where ω and ω' [$\equiv (2n + 1)\pi\beta$] are fermionic frequencies, and ν is bosonic ($\equiv 2n\pi/\beta$).

Local irreducible vertex functions $\Gamma_{\alpha\beta}(\omega, \omega', \nu)$

From the two-particle correlation function $\chi_{\alpha\beta}(\omega, \omega', \nu)$, the local irreducible vertex function $\Gamma_{\alpha\beta}(\omega, \omega', \nu)$ can be calculated easily, via the Bethe–Salpeter equation [30,31,40]:

$$\Gamma_{\alpha\beta}(\omega, \omega', \nu) = \frac{\chi_{\alpha\beta}(\omega, \omega', \nu) - \beta[G_\alpha(\omega + \nu)G_\beta(\omega')\delta_{\nu,0} - G_\alpha(\omega + \nu)G_\beta(\omega')\delta_{\alpha\beta}\delta_{\omega\omega'}]}{G_\alpha(\omega + \nu)G_\alpha(\omega)G_\beta(\omega')G_\beta(\omega' + \nu)}. \tag{9}$$

4

The $G(i\omega_n)$ and $\Gamma_{\alpha\beta}(\omega, \omega', \nu)$ are essential inputs for the diagrammatic extensions of DMFT, such as the dual fermions (DF) [41] and dynamical vertex approximation (DΓA) [42] codes.

<u>Impurity self-energy function $\Sigma(i\omega_n)$</u>

The self-energy $\Sigma(i\omega_n)$ is calculated using Dyson's equation

$$\Sigma(i\omega_n) = G_0^{-1}(i\omega_n) - G^{-1}(i\omega_n), \tag{10}$$

or measured using the so-called improved estimator [30,31]. Note that in the current implementation the latter approach only works when the segment representation is used.

<u>Histogram of the perturbation expansion order</u>

We record the histogram of the perturbation expansion order k, which can be used to evaluate the kinetic energy via Eq. (15).

<u>Occupation number and double occupation number</u>

The orbital occupation number $\langle n_\alpha \rangle$ and double occupation number $\langle n_\alpha n_\beta \rangle$ are measured. From them we can calculate for example the charge fluctuation $\sqrt{\langle N^2 \rangle - \langle N \rangle^2}$, where N is the total occupation number:

$$N = \sum_\alpha n_\alpha. \tag{11}$$

<u>Spin–spin correlation function</u>

For a system with spin rotational symmetry, the expression for the spin–spin correlation function reads

$$\chi_{ss}(\tau) = \langle S_z(\tau) S_z(0) \rangle, \tag{12}$$

where $S_z = n_\uparrow - n_\downarrow$. From it we can calculate the effective magnetic moment:

$$\mu_{\text{eff}} = \int_0^\beta d\tau \, \chi_{ss}(\tau). \tag{13}$$

<u>Orbital–orbital correlation function</u>

The expression for the orbital–orbital correlation function reads

$$\chi_{\alpha\beta}^{nn}(\tau) = \langle n_\alpha(\tau) n_\beta(0) \rangle. \tag{14}$$

<u>Kinetic energy</u>

In DMFT, the expression for the kinetic energy of the lattice model reads

$$E_{\text{kin}} = -\frac{1}{\beta} \langle k \rangle, \tag{15}$$

where k is the perturbation expansion order.

<u>Atomic state probability</u>

The expression for the atomic state probability is

$$p_\Gamma = \langle |\Gamma\rangle\langle\Gamma| \rangle, \tag{16}$$

where Γ is the atomic state.

3. Implementations and optimizations

In this section, we will focus on the implementation details and discuss the optimization tricks adopted in the *i*QIST software package.

3.1. Development platform

The major part of the *i*QIST software package was developed with the modern Fortran 90 language. We extensively used advanced language features in the Fortran 2003/2008 standard such as an object oriented programming style (polymorphic, inheritance, and module, etc.) to improve the readability and re-usability of the source codes. The compiler is fixed to the Intel Fortran compiler. We cannot guarantee that the *i*QIST can be compiled successfully with other Fortran compilers. Some auxiliary scripts, pre- and post-processing tools are written using the Python language and Bash shell scripts. These scripts and tools act like a glue. They are very flexible and can be easily extended or adapted to deal with various problems. These Python codes can run properly under the Python 2.x or 3.x runtime environment.

Since *i*QIST is a big software development project, we use Git as the version control system, and the source codes are hosted in a remote server. The developers pull the source codes from the server into their local machines, and then try to improve them. Once the modification is completed, the source codes can be pushed back to the server and merged with the master branch. Then the other developers can access them and use them immediately to start further developments. The members of our developer team can access the code repository anywhere and anytime.

3.2. Orthogonal polynomial representation

Boehnke et al. [29] proposed to use Legendre polynomials to improve the measurements of single-particle and two-particle Green's functions. Thanks to the Legendre polynomial representation, the numerical noise and memory space needed to store Green's function are greatly reduced.

The imaginary time Green's function $G(\tau)$ is expressed using the Legendre polynomial $P_n(x)$ defined in $[-1, 1]$:

$$G(\tau) = \frac{1}{\beta} \sum_{n \geq 0} \sqrt{2n+1} P_n[x(\tau)] G_n, \tag{17}$$

where n is the order of Legendre polynomial, G_n is the expansion coefficient, $x(\tau)$ maps $\tau \in [0, \beta]$ to $x \in [-1, 1]$:

$$x(\tau) = \frac{2\tau}{\beta} - 1. \tag{18}$$

Using the orthogonality relations of Legendre polynomials, we obtain

$$G_n = \sqrt{2n+1} \int_0^\beta d\tau P_n[x(\tau)] G(\tau). \tag{19}$$

If we substitute Eq. (5) into Eq. (19), we get

$$G_n = -\frac{\sqrt{2n+1}}{\beta} \left\langle \sum_{i=1}^k \sum_{j=1}^k \mathcal{M}_{ji} \tilde{P}_n(\tau_i^e - \tau_j^s) \right\rangle, \tag{20}$$

where

$$\tilde{P}_n(\tau) = \begin{cases} P_n[x(\tau)], & \tau > 0, \\ -P_n[x(\tau + \beta)], & \tau < 0, \end{cases} \tag{21}$$

and τ^s and τ^e denote the positions of creation and annihilation operators on the imaginary time axis, respectively. We can also express the Matsubara Green's function $G(i\omega_n)$ using Legendre polynomials:

$$G(i\omega_m) = \sum_{n \geq 0} T_{mn} G_n. \tag{22}$$

The transformation matrix T_{mn} is defined as

$$T_{mn} = (-1)^m i^{n+1} \sqrt{2n+1} j_n \left[\frac{(2m+1)\pi}{2} \right], \tag{23}$$

where $j_n(z)$ is the spheric Bessel function. Actually, in the Monte Carlo simulation, only the expansion coefficients G_n are measured. When the calculation is finished, the final Green's function can be evaluated using Eqs. (17) and (22). It is worthwhile to note that the T_{mn} do not depend on the inverse temperature β, so that we can calculate and store the matrix elements beforehand to save computer time.

It is easy to extend this formalism to other orthogonal polynomials. For example, in the *iQIST* software package, we not only implemented the Legendre polynomial representation, but also the Chebyshev polynomial representation. In the Chebyshev polynomial representation, the imaginary time Green's function $G(\tau)$ is expanded as follows:

$$G(\tau) = \frac{2}{\beta} \sum_{n \geq 0} U_n[x(\tau)] G_n, \tag{24}$$

where the $U_n(x)$ denote the second kind of Chebyshev polynomials and $x \in [-1, 1]$. The equation for the expansion coefficients G_n is:

$$G_n = -\frac{2}{\pi\beta} \left\langle \sum_{i=1}^k \sum_{j=1}^k \mathcal{M}_{ji} \tilde{U}_n(\tau_i^e - \tau_j^s) \sqrt{1 - \tilde{x}(\tau_i^e - \tau_j^s)^2} \right\rangle, \tag{25}$$

where

$$\tilde{U}_n(x) = \begin{cases} U_n[x(\tau)], & \tau > 0, \\ -U_n[x(\tau + \beta)], & \tau < 0, \end{cases} \tag{26}$$

and

$$\tilde{x}(\tau) = \begin{cases} x(\tau), & \tau > 0, \\ x(\tau + \beta), & \tau < 0. \end{cases} \tag{27}$$

Unfortunately, there is no explicit expression for $G(i\omega_n)$ [like Eq. (22)] in the Chebyshev polynomial representation.

3.3. Improved estimator for the self-energy function and vertex function

Recently, Hafermann et al. proposed efficient measurement procedures for the self-energy and vertex functions within the CT-HYB algorithm [30,31]. In their method, some higher-order correlation functions (related to the quantities being sought through the equation of motion) are measured. For the case of density–density interactions, the segment algorithm is available [25]. Thus, the additional correlators can be obtained essentially without additional computational cost. When the calculations are completed, the required self-energy function and vertex function can be evaluated analytically.

The improved estimator for the self-energy function can be expressed in the following form:

$$\Sigma_{ab}(i\omega_n) = \frac{1}{2} \sum_{ij} G_{ai}^{-1}(i\omega_n)(U_{jb} + U_{bj})F_{ib}^{j}(i\omega_n), \tag{28}$$

where U_{ab} is the Coulomb interaction matrix element. The expression for the new two-particle correlator $F_{ab}^{j}(\tau - \tau')$ reads

$$F_{ab}^{j}(\tau - \tau') = -\langle \mathcal{T} d_a(\tau)d_b^{\dagger}(\tau')n_j(\tau')\rangle, \tag{29}$$

and $F_{ab}^{j}(i\omega_n)$ is its Fourier transform. The actual measurement formula is

$$F_{ab}^{j}(\tau - \tau') = -\frac{1}{\beta}\left\langle \sum_{\alpha\beta=1}^{k} \mathcal{M}_{\beta\alpha}\delta^{-}(\tau - \tau', \tau_\alpha^e - \tau_\beta^s)n_j(\tau_\beta^s)\delta_{a,\alpha}\delta_{b,\beta}\right\rangle. \tag{30}$$

The measurement formula for the vertex function can be found in the original paper [30,31]. Note that when the Coulomb interaction is frequency-dependent, Eqs. (28) and (30) should be modified slightly [31]. As one can see, this equation for $F_{ab}^{j}(\tau - \tau')$ looks quite similar to Eq. (5). Thus we use the same method to measure $F_{ab}^{j}(\tau - \tau')$ and finally get the self-energy function via Eq. (28). Here, the matrix element $n_j(\tau_\beta^s)$ (one or zero) denotes whether or not the flavor j is occupied (whether or not a segment is present) at time τ_β^s.

This method can be combined with the orthogonal polynomial representation [29] as introduced in the previous subsection to suppress fluctuations and filter out the Monte Carlo noise. Using this technique, we can obtain the self-energy and vertex functions with unprecedented accuracy, which leads to an enhanced stability in the analytical continuations of those quantities [30].

3.4. Subspaces and symmetry

As mentioned before, for a Hamiltonian H_{loc} with general interactions the evaluation of local trace is heavily time-consuming,

$$\omega_d(\mathcal{C}) = \text{Tr}_{\text{loc}}(T_{2k+1}F_{2k}T_{2k}\cdots F_1 T_1), \tag{31}$$

where $T = e^{-\tau H_{\text{loc}}}$ is time evolution operator, F is fermionic creation or annihilation operator, and k is expansion order for the current diagrammatic configuration $\mathcal{C}$. The straightforward method to evaluate this trace is to insert the complete eigenstates $\{\Gamma\}$ of H_{loc} into the RHS of Eq. (31), then

$$\omega_d(\mathcal{C}) = \sum_{\{\Gamma_1\cdots\Gamma_{2k}\}} \langle\Gamma_1|T_{2k+1}|\Gamma_1\rangle\langle\Gamma_1|F_{2k}|\Gamma_{2k}\rangle\langle\Gamma_{2k}|T_{2k}|\Gamma_{2k}\rangle\cdots\langle\Gamma_2|F_1|\Gamma_1\rangle\langle\Gamma_1|T_1|\Gamma_1\rangle. \tag{32}$$

Thus, we must do $4k + 1$ matrix–matrix multiplications with the dimension of the Hilbert space of H_{loc}. This method is robust but very slow for large multi-orbital impurity model as the dimension of the matrix is impractically large for 5- and 7-band systems, and the expansion order k is large as well.

Actually, the matrices of the fermion operators (F-matrix) are very sparse due to the symmetry of H_{loc}. We can take advantage of this to speed up the matrix–matrix multiplications. We exploit the symmetry of H_{loc} to find some good quantum numbers (GQNs) and divide the full Hilbert space of H_{loc} with very large dimension into much smaller subspaces labeled by these GQNs [1]. We call such a subspace $|\alpha\rangle$ a superstate [27] which consists of all the n_α eigenstates of this subspace, $|\alpha\rangle = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{n_\alpha}\}$. The F-matrix element can only be nonzero between pairs of superstates with different values of GQNs. One fermion operator may bring one initial superstate $|\alpha\rangle$ to some other final superstates $|\beta\rangle$,

$$F|\alpha\rangle = |\beta\rangle, \tag{33}$$

or outside of the full Hilbert space. We have to carefully choose the GQNs to make sure that for a fixed initial superstate $|\alpha\rangle$ and a fixed fermion operator, there is one and only one final superstate $|\beta\rangle$ if it does not go outside of the full Hilbert space. Given an arbitrary diagrammatic configuration, starting with a superstate $|\alpha_1\rangle$, there will be only one possible evolution path. That is,

$$|\alpha_1\rangle \xrightarrow{F_1} |\alpha_2\rangle \xrightarrow{F_2} |\alpha_3\rangle \xrightarrow{F_3} |\alpha_4\rangle \cdots |\alpha_{2k-1}\rangle \xrightarrow{F_{2k-1}} |\alpha_{2k}\rangle \xrightarrow{F_{2k}} |\alpha_1\rangle. \tag{34}$$

The path may break at some point because it goes outside of the full Hilbert space or violates the Pauli principle. For a successful path starting with $|\alpha_1\rangle$, its contribution to the local trace is

$$\text{Tr}_{\alpha_1} = \sum_{\{\Gamma_{\alpha_1}\cdots\Gamma_{\alpha_{2k}}\}} \langle\Gamma_{\alpha_1}|T_{2k+1}|\Gamma_{\alpha_1}\rangle\langle\Gamma_{\alpha_1}|F_{2k}|\Gamma_{\alpha_{2k}}\rangle\langle\Gamma_{\alpha_{2k}}|T_{2k}|\Gamma_{\alpha_{2k}}\rangle\cdots\langle\Gamma_{\alpha_2}|F_1|\Gamma_{\alpha_1}\rangle\langle\Gamma_{\alpha_1}|T_1|\Gamma_{\alpha_1}\rangle, \tag{35}$$

where $\{\Gamma_{\alpha_i}\}$ are the eigenstates of subspace α_i. Thus, the final local trace should be

$$\omega_d(\mathcal{C}) = \sum_i \text{Tr}_{\alpha_i}. \tag{36}$$

Table 1
The GQNs supports for various types of local Hamiltonians H_{loc}.

GQNs	Kanamori-U	Slater-U	SOC
N, S_z	Yes	Yes	No
N, S_z, PS	Yes	No	No
N, J_z	Yes	Yes	Yes
N	Yes	Yes	Yes

Table 2
The total number of subspaces N, maximum and mean dimensions of subspaces for different GQNs schemes and multi-orbital models.

GQNs	2-band N/max/mean	3-band N/max/mean	5-band N/max/mean	7-band N/max/mean
N, S_z	9/4/1.78	16/9/4.00	36/100/28.44	64/1225/256.00
N, S_z, PS	14/2/1.14	44/3/1.45	352/10/2.91	2368/35/6.92
N, J_z	–	26/5/2.46	96/37/10.67	246/327/66.60
N	5/6/3.20	7/20/9.14	11/252/93.09	15/3432/1092.27

As a result, the original $4k + 1$ matrix–matrix multiplications with large dimension reduces to several times $4k + 1$ matrix–matrix multiplications with much smaller dimensions, resulting in a huge speedup.

In our codes, we implemented several GQNs schemes for different types of local Hamiltonians H_{loc}, as summarized in Table 1. For H_{loc} without SOC, we have two choices: (1) with Slater parameterized Coulomb interaction matrix, we use the total occupation number N, the z component of total spin S_z as GQNs; (2) with Kanamori parameterized Coulomb interaction matrix, besides N and S_z, we can use another powerful GQN, the so-called PS number [32]. It is defined as,

$$\text{PS} = \sum_{\alpha=1}^{N_{\text{orb}}} (n_{\alpha\uparrow} - n_{\alpha\downarrow})^2 \times 2^{\alpha}, \tag{37}$$

where α is the orbital index, $\{\uparrow, \downarrow\}$ is spin index, $n_{\alpha\uparrow}$ and $n_{\alpha\downarrow}$ are the orbital occupancy numbers. The PS number labels the occupation number basis with the same singly occupied orbitals. With its help, the dimensions of the subspaces become very small, such that we can treat 5-band Kanamori parameterized interaction systems efficiently without any approximations. For H_{loc} with SOC, we can use the total occupancy number N and the z component of total angular momentum J_z as GQNs. We summarize the total number of subspaces, maximum and mean dimensions of subspaces for different GQNs schemes and multi-orbital impurity models in Table 2. Obviously, using these GQNs can largely reduce the dimension of the F-matrix, and make accurate DMFT calculations for complex electronic systems (such as the d- and f-electron materials) possible.

3.5. Truncation approximation

As discussed in Section 3.4, although we have used GQNs to split the full Hilbert space with very large dimension into blocks with smaller dimensions [for cases such as 7-band systems with GQNs (N, J_z) and 5-band systems with GQN (N)], the dimensions of some blocks are still too large and the number of blocks is too high, so that it is still very expensive to evaluate the local trace. K. Haule proposed in Ref. [27] to discard some high-energy states because they are rarely visited. For example, for 7-band system with only 1 electron (like Ce metal), only states with occupancy $N = 0, 1, 2$ will be frequently visited, and states with occupancy $N > 2$ can be truncated completely to reduce the large Hilbert space to a very small one. Of course, this truncation approximation may cause some bias because a frequently visited state may be accessed via an infrequently visited state. Therefore, one should be cautious when adopting the truncation approximation, and for example run some convergence tests.

Currently, we adopted two truncation schemes in our codes. The first scheme relies on the cut-off of the occupation number. We just keep those states whose occupation numbers are close to the nominal valence and skip the other states, as shown in the above Ce metal example. This scheme is quite robust if the charge fluctuations are small enough, such as in the case of a Mott insulating phase. Another scheme is to dynamically truncate the states with very low probability based on statistics which is recorded during the Monte Carlo sampling. This scheme is not very stable, so one needs to use it with caution.

3.6. Lazy trace evaluation

The diagrammatic Monte Carlo sampling algorithm consists of the following steps: (1) Propose an update for the current diagrammatic configuration. (2) Calculate the acceptance probability p according to the Metropolis–Hasting algorithm,

$$p = \min\left(1, \frac{A'}{A} \left|\frac{\omega_c}{\omega_c'}\right| \left|\frac{\omega_d}{\omega_d'}\right|\right), \tag{38}$$

where, A is the proposal probability for the current update and A' for the inverse update, ω_c and ω_c' are the determinants for the new and old configurations, respectively, and ω_d and ω_d' are the local traces for the new and old configurations, respectively. (3) Generate a random number r. If $p > r$, the proposed update is accepted, otherwise it is rejected. (4) Update the current diagrammatic configuration if the proposed update is accepted. It turns out that for CT-HYB, p is usually low (1%–20%), especially in the low temperature region. On the other hand, the calculation of p involves a costly local trace evaluation. To avoid wasting computation time when the acceptance probability is very low, in the subspace algorithm, we implemented the so-called lazy trace evaluation proposed in Ref. [33].
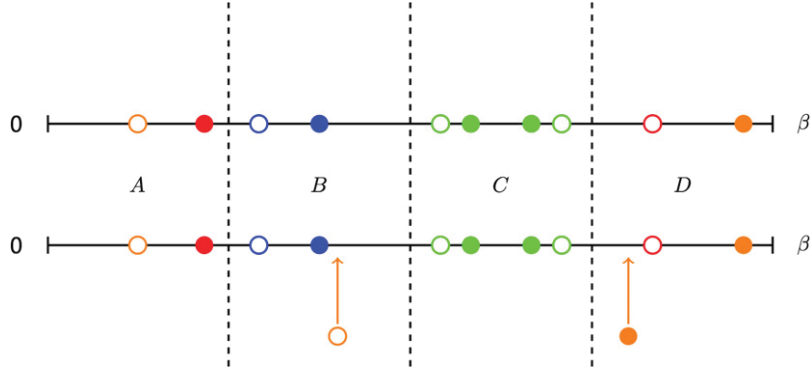
Fig. 1. Illustration of the divide-and-conquer algorithm. The imaginary time interval $[0, \beta)$ is split into four parts with equal length by vertical dashed lines. The open (filled) circles mean creation (annihilation) operators. The color is used to distinguish different flavors. It shows that a creation operator is inserted into the B part, while a annihilation operator is inserted into the D part. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The basic idea of the lazy trace evaluation is simple. For the proposed Monte Carlo move, we first generate a random number r. Then, instead of calculating the local trace from scratch to determine p, we calculate bounds for $|\text{Tr}_{\text{loc}}|$,

$$|\omega_d| = |\text{Tr}_{\text{loc}}| \leq \sum_i |\text{Tr}_i| \leq \sum_i B_i, \tag{39}$$

where $B_i \geq |\text{Tr}_i|$. B_i is a product of some chosen matrix norms of T and F matrices:

$$B_i = C \, \|T_{2k+1}\| \, \|F_{2k}\| \, \|T_{2k}\| \cdots \|F_1\| \, \|T_1\| \geq |\text{Tr}(T_{2k+1}F_{2k}T_{2k}\cdots F_1T_1)| \,, \tag{40}$$

where C is a parameter depending on the specific type of matrix norm, and $\|\cdot\|$ denotes a matrix norm. If $\text{Tr}_{i'}$ denotes the exact traces of some subspaces, then we have

$$\left| |\text{Tr}_{\text{loc}}| - \sum_{i'} |\text{Tr}_{i'}| \right| \leq \sum_{i \neq i'} B_i. \tag{41}$$

Thus, we can determine the upper $p_{\max}$ and lower $p_{\min}$ bounds of p as

$$p_{\max} = R \left(\sum_{i'} |\text{Tr}_{i'}| + \sum_{i \neq i'} B_i \right),$$
$$p_{\min} = R \left(\sum_{i'} |\text{Tr}_{i'}| - \sum_{i \neq i'} B_i \right), \tag{42}$$

where $R = \frac{A'}{A} \left| \frac{\omega_c}{\omega'_c} \right| \left| \frac{1}{\omega'_d} \right|$. If $r > p_{\max}$, we reject this move immediately. If $r < p_{\min}$, we accept the move and calculate the determinant and local trace from scratch. If $p_{\min} < r < p_{\max}$, we refine the bounds by calculating the local trace of one more subspace Tr_i until we can reject or accept the move. The calculation of these bounds involves only simple linear algebra calculations of matrix norms which cost little computation time, and one refining operation involves only one subspace trace evaluation. On average, it saves a lot of computation time, as confirmed by our benchmarks.

3.7. Divide-and-conquer and sparse matrix tricks

The Monte Carlo updates, such as inserting (removing) a pair of creation and annihilation operators, usually modify the diagrammatic configuration locally. Based on this fact, we implemented a divide-and-conquer algorithm to speed up the trace evaluation. As illustrated in Fig. 1, we divide the imaginary time interval $[0, \beta)$ into a few parts with equal length. For each part, there will be zero or nonzero fermion operators, and we save their matrix products when evaluating the local trace in the beginning. In the next Monte Carlo sampling, we first determine which parts may be modified or influenced, and then for these parts we recalculate the matrix products from scratch and save them again. For the unchanged parts, we will leave them alone. Finally, we will multiply the contributions of all parts to obtain the final local trace. By using this divide-and-conquer trick, we can avoid redundant computations and speed up the calculation of the acceptance probability p. This trick can be combined with the GQNs algorithm and lazy trace evaluation to achieve a further speedup.

If direct matrix–matrix multiplications are used when evaluating the local trace, the F-matrix must be very sparse. Thus, we can convert them into sparse matrices in compressed sparse row (CSR) format, and then the sparse matrix multiplication can be applied to obtain a significant speedup.

3.8. Random number generators

Fast, reliable, and long period pseudo-random number generators are a key factor for Monte Carlo simulations. Currently, the most popular random number generator is the Mersenne Twister which was developed by Matsumoto and Nishimura [43]. Its name derives
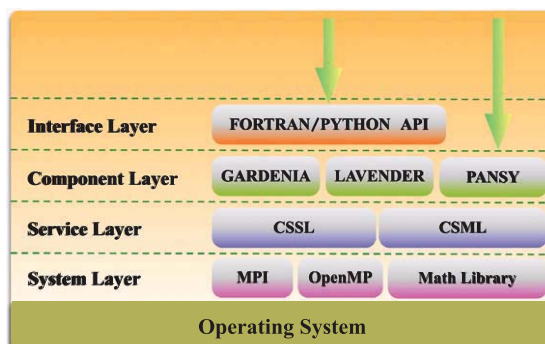
Fig. 2. The hierarchical structure of the *i*QIST software package. Note that in the component layer, not all of the components are listed due to space limitations. See the main text for detailed explanations.

from the fact that its period length is chosen to be a Mersenne prime. In the *i*QIST software package, we implemented the commonly used version of Mersenne Twister, MT19937. It has a very long period of $2^{19937} - 1$.

The Mersenne Twister is a bit slow by today's standards. So in 2006, a variant of Mersenne Twister, the SIMD-oriented Fast Mersenne Twister (SFMT) was introduced [44]. It was designed to be fast when it runs on 128-bit SIMD. It is almost twice as fast as the original Mersenne Twister and has better statistics properties. We also implemented it in the *i*QIST software package, and use it as the default random number generator.

3.9. Parallelization

All of the CT-HYB impurity solvers in the *i*QIST software package are parallelized by MPI. The strategy is very simple. In the beginning, we launch n processes simultaneously. The master process is responsible for reading input data and configuration parameters, and broadcasts them among the child processes. Then each child process will perform Monte Carlo samplings and measure physical observables independently. After all the processes finish their jobs, the master process will collect the measured quantities from all the processes and average them to obtain the final results. Apart from that, no additional inter-process communication is needed. Thus, we can anticipate that the parallel efficiency will be very good, and near linear speedups are possible, as long as the number of thermalization steps is small compared to the total number of Monte Carlo steps. In practical calculations, we usually fix the number of Monte Carlo steps N_{sweep} done by each process, and launch as many processes as possible. Given that the number of processes is N_{proc}, then the total number of Monte Carlo samplings should be $N_{\text{proc}}N_{\text{sweep}}$. Naturally, the more processes we use, the more accurate data we can obtain.

For some specific tasks, such as the measurement of two-particle quantities, fine-grained parallelism is necessary. Thus, we further parallelized them with the OpenMP multi-thread technology. So, in order to attain ideal speedup, we have to carefully choose suitable numbers of MPI processes and OpenMP threads.

4. Features

In this section, we will introduce the software architecture and component framework of *i*QIST. The major features of its components are presented in detail.

4.1. Software architecture

To solve a quantum impurity model is not a straightforward job. Besides the necessary quantum impurity solvers, we need some auxiliary programs or tools. The *i*QIST is an all-in-one software package, which can be used to solve a broad range of quantum impurity problems. It is a collection of various codes and scripts whose core components contain about 120 000 lines of code.

The software architecture of *i*QIST is slightly involved. In Fig. 2, we use a layer model to illustrate it. The bottom layer is the operating system (OS). In principle, the *i*QIST is OS-independent. It can run properly on top of Unix/Linux, Mac OS X, FreeBSD, and Windows. The second layer is the system layer, which contains highly optimized linear algebra math libraries (such as BLAS and LAPACK) and parallelism supports (such as MPI and OpenMP). The third layer is the service layer. In this layer, we implemented some commonly used modules and subroutines. They are called common service subroutine library (CSSL) and common service module library (CSML), respectively. They provide a useful interface between the system layer and the component layer and facilitate the development of core components. The features of CSSL and CSML include basic data structures (stack and linked list), random number generators, sparse matrix manipulations, linear algebra operations, string processing, linear interpolation, numerical integration, fast Fourier transformation (FFT), etc.

The core part of *i*QIST is in the fourth layer – the component layer – which contains various impurity solvers and auxiliary tools as shown in Fig. 3. At present, *i*QIST contains ten different components. They are AZALEA, GARDENIA, NARCISSUS, BEGONIA, LAVENDER, PANSY, MANJUSHAKA, DAISY, JASMINE, and HIBISCUS. Here, AZALEA, GARDENIA, NARCISSUS, BEGONIA, LAVENDER, PANSY, and MANJUSHAKA are all CT-HYB impurity solver components (as shown on the LHS of Fig. 3), and DAISY is a HF-QMC impurity solver component. JASMINE is an atomic eigenvalue solver. HIBISCUS is a collection of several pre- and post-processing tools, including the maximum entropy method, stochastic analytical continuation, Padé approximation, and Kramers–Kronig transformation, etc. For more details about these components, please consult the following sections.

The top layer is the interface layer or user layer. On the one hand, we can execute *i*QIST's components directly as usual. On the other hand, we can also invoke *i*QIST's components from other languages. The role of *i*QIST's components becomes a library or subroutine. To
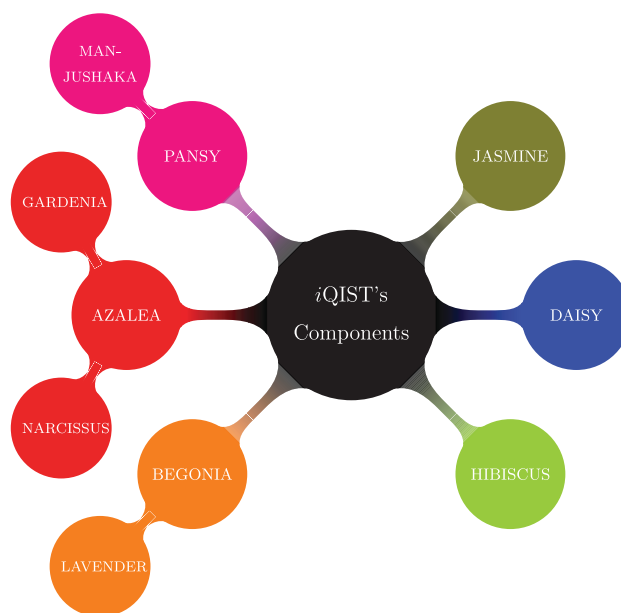
Fig. 3. Schematic picture for the *i*QIST's components. Components on the LHS are the CT-HYB solvers, `JASMINE` is the atomic eigenvalue solver, `DAISY` is a HF-QMC solver, and `HIBISCUS` contains the other pre- and post-processing tools.

Table 3
The models supported by various CT-HYB impurity solvers in the *i*QIST software package. In this and the following tables, the CT-HYB impurity solvers are abbreviated using the first capital letter of their names. For example, `A` denotes the `AZALEA` component.

Models	CT-HYB
Density–density interaction	A, G, N, B, L, P, M
General Coulomb interaction (Slater or Kanamori schemes)	B, L, P, M
Spin–orbit coupling interaction	B, L, P, M
Crystal field splitting	A, G, N, B, L, P, M
Hubbard–Holstein model	N
Frequency-dependent (retarded) interaction	N

Table 4
The measurement tricks used by various CT-HYB impurity solvers in the *i*QIST software package.

Measurement tricks	CT-HYB
Orthogonal polynomial representation (Legendre and Chebyshev types)	G, N, L, M
Improved estimator for self-energy and vertex functions	G, N

Table 5
The trace evaluation algorithms supported by various CT-HYB impurity solvers in the *i*QIST software package.

Trace algorithms	CT-HYB
Segment representation algorithm	A, G, N
Divide-and-conquer algorithm	B, L, P, M
Sparse matrix multiplication	B, L
Good quantum numbers	P, M
Skip-lists trick	M
Lazy trace evaluation	M
Dynamical truncation approximation	M

achieve this goal, in the interface layer, we offer the Fortran/Python language bindings for most of the *i*QIST components, so that we can develop our own codes on top of *i*QIST and consider it as a computational engine in black box.

4.2. CT-HYB impurity solvers

As mentioned before, the *i*QIST software package contains seven CT-HYB impurity solvers (as schematically shown in Fig. 3). In this subsection, in order to help the users to choose a suitable CT-HYB impurity solver, we briefly discuss their main features, pros, and cons. The main results are also summarized in Tables 3–6 for a quick query.

When the Coulomb interaction term in the local Hamiltonian H_{loc} is of density–density type, H_{loc} becomes a diagonal matrix in the occupation number basis. In this case, the CT-HYB impurity solver is extremely efficient if the so-called segment picture (or segment representation)[1,25] is adopted. Thus, we implemented the segment algorithm in the `AZALEA`, `GARDENIA`, and `NARCISSUS` components.

Table 6
The observables measured by various CT-HYB impurity solvers in the *iQIST* software package.

Physical observables	CT-HYB
Single-particle Green's function $G(\tau)$	A, G, N, B, L, P, M
Single-particle Green's function $G(i\omega_n)$	A, G, N, B, L, P, M
Two-particle correlation function $\chi(\omega, \omega', \nu)$	G, N, L, M
Local irreducible vertex function $\Gamma(\omega, \omega', \nu)$	G, N, L, M
Pair susceptibility $\Gamma_{pp}(\omega, \omega', \nu)$	G, N, L, M
Self-energy function $\Sigma(i\omega_n)$	A, G, N, B, L, P, M
Histogram of perturbation expansion order	A, G, N, B, L, P, M
Kinetic and potential energies	A, G, N, B, L, P, M
(Double) occupation numbers, magnetic moment	A, G, N, B, L, P, M
Atomic state probability	A, G, N, B, L, P, M
Spin–spin correlation function	G, N
Orbital–orbital correlation function	G, N
Autocorrelation function and autocorrelation time	G, N, L, M

In the `AZALEA` component, we only implemented the basic segment algorithm and very limited physical observables are measured. It is the simplest and the most efficient code. In fact, it is the development prototype of the other CT-HYB components, and usually used to test some experimental features. In the `GARDENIA` component, we add more features on the basis of the `AZALEA` component. For example, we can use the orthogonal polynomial technique to improve the numerical accuracy and suppress stochastic noise in Green's function [29]. The self-energy function can be measured with the improved estimator method [30,31]. More single-particle and two-particle correlation functions are measured. Though `GARDENIA` is much more powerful than `AZALEA`, it is a bit less efficient. The features of the `NARCISSUS` component are almost the same as those of the `GARDENIA` component. In addition, it can be used to deal with dynamically screened interactions [9,45]. In other words, the Coulomb interaction U need not to be a static value any more, but can be frequency-dependent. Thus, it is used for example in extended-DMFT calculations [46]. Note that since the Hubbard–Holstein model can be mapped in DMFT onto a dynamical Anderson impurity model [47], it can be solved using the `NARCISSUS` component as well.

When the local Hamiltonian H_{loc} contains general Coulomb interaction terms, there is no simple expression for the $\omega_d(\mathcal{C}_n)$ and the segment representation is not applicable any more. At that time, the general matrix formulation [26,27], which is implemented in the `BEGONIA`, `LAVENDER`, `PANSY`, and `MANJUSHAKA` components, should be used. Each of these components has its own features and targets specific systems.

In the `BEGONIA` component, we implemented the direct matrix–matrix multiplications algorithm. We adopted the divide-and-conquer scheme and sparse matrix technique to speed up the calculation. This component can be used to deal with impurity models with up to 3 bands with fairly good efficiency. However, it is not suitable for 5- and 7-band systems. In the `LAVENDER` component, we implemented all the same algorithms as in the `BEGONIA` component. Besides, we implemented the orthogonal polynomial representation to improve the measurement quality of physical quantities. Some two-particle quantities are also measured. This component should also only be used to conduct calculations for 1–3 bands systems. But it can produce measurements of very high quality with small additional cost. In the `PANSY` component, we exploited the symmetries of H_{loc} and applied the GQNs trick to accelerate the evaluation of local trace. This algorithm is general and does not depend on any details of the GQNs, so it can support all the GQNs schemes which fulfill the conditions discussed in Section 3.4. We also adopted the divide-and-conquer algorithm to speed it up further. This component can be used to study various impurity models ranging from 1-band to 5-band with fairly good efficiency. However, it is still not suitable for 7-band models. In the `MANJUSHAKA` component, we implemented all the same algorithms as the `PANSY` component. Besides, we implemented the lazy trace evaluation [33] to speed up the Monte Carlo sampling process. It can gain quite high efficiency, and is extremely useful in the low temperature region. We also implemented a smart algorithm to truncate some high-energy states dynamically in the Hilbert space of H_{loc} to speed up the trace evaluation further. This algorithm is very important and efficient (in many situations it is necessary) for dealing with 7-band systems. We implemented the orthogonal polynomial representation to improve the measurements of key observables as well. By using all of these tricks, the computational efficiency of the `MANJUSHAKA` component for multi-orbital impurity models with general Coulomb interaction is very high. We believe that it can be used to study most quantum impurity systems ranging from 1-band to 7-band.

4.3. Atomic eigenvalue solver

When the Coulomb interaction is general in the local Hamiltonian H_{loc}, as discussed above, we have to diagonalize H_{loc} in advance to obtain its eigenvalues, eigenvectors, and the F-matrix. In general, the local Hamiltonian is defined as

$$H_{loc} = H_{int} + H_{cf} + H_{soc}, \tag{43}$$

where H_{int} means the Coulomb interaction term, H_{cf} the CF splitting term, and H_{soc} the SOC interaction. The `JASMINE` component is used to solve this Hamiltonian and generate necessary inputs for some CT-HYB impurity solvers (i.e., `BEGONIA`, `LAVENDER`, `PANSY`, and `MANJUSHAKA` components).

The `JASMINE` component will build H_{loc} in the Fock representation at first. For the Coulomb interaction term H_{int}, both Kanamori parameterized and Slater parameterized forms are supported. In other words, we can use U and J, or Slater integrals F^k to define the Coulomb interaction matrix as we wish. For the CF splitting term H_{cf}, both diagonal and non-diagonal elements are accepted. The SOC term H_{soc} is defined as follows,

$$H_{soc} = \lambda \sum_i \vec{\mathbf{l}}_i \cdot \vec{\mathbf{s}}_i, \tag{44}$$

where λ is the strength of the SOC. Note that the SOC term can only be activated for the 3-, 5-, and 7-band systems.

Next, the `JASMINE` component will diagonalize H_{loc} to get all eigenvalues and eigenvectors. There are two running modes for `JASMINE`. (1) It diagonalizes H_{loc} in the full Hilbert space directly to obtain the eigenvalues E_α and eigenvectors Γ_α, then the F-matrix is built from the eigenvectors,

$$(F_i)_{\alpha,\beta} = \langle \Gamma_\alpha | F_i | \Gamma_\beta \rangle, \tag{45}$$

where i is the flavor index. The eigenvalues and F-matrix will be fed into the `BEGONIA` and `LAVENDER` components as necessary input data. (2) It diagonalizes each subspace of H_{loc} according to the selected GQNs. Currently, four GQNs schemes for various types of H_{loc} are supported, which are summarized in Table 1. `JASMINE` also builds indices to record the evolution sequence depicted in Eq. (33). According to the indices, it builds the F-matrix between two different subspaces. The eigenvalues, the indices, and the F-matrix will be collected and written into an external file (atom.cix), which will be read by the `PANSY` and `MANJUSHAKA` components.

Apart from this, the `JASMINE` component will also generate the matrix elements of some physical operators, such as $\vec{L}^2$, L_z, $\vec{S}^2$, S_z, $\vec{J}^2$, and J_z. They can be used by the other post-processing codes to analyze the averaged expectation value of these operators.

4.4. Auxiliary tools

In the `HIBISCUS` component, many auxiliary tools are provided to deal with the output data of the CT-HYB impurity solvers. Here we briefly describe some of these tools:

Maximum entropy method

In the Monte Carlo community, the maximum entropy method [48] is often used to extract the spectral function $A(\omega)$ from the imaginary time Green's function $G(\tau)$. Thus, in the `HIBISCUS` component, we implemented the standard maximum entropy algorithm. In the Extended-DMFT calculations, sometimes we have to perform an analytical continuation for the retarded interaction function $\mathcal{U}(i\nu)$ to obtain $\mathcal{U}(\nu)$ [49]. So we developed a modified version of the maximum entropy method to enable this calculation.

Stochastic analytical continuation

An alternative way to extract $A(\omega)$ from $G(\tau)$ is the stochastic analytical continuation [50]. Unlike the maximum entropy method, the stochastic analytical continuation does not depend on any *a priori* parameters. It has been argued that the stochastic analytical continuation can produce more accurate spectral functions with more subtle structures. In the `HIBISCUS` component, we also implemented the stochastic analytical continuation which can be viewed as a useful complementary procedure to the maximum entropy method. Since the stochastic analytical continuation is computationally much heavier than the maximum entropy method, we parallelized it with MPI and OpenMP.

Kramers–Kronig transformation

Once the analytical continuation is finished, we can obtain the spectral function $A(\omega)$ and the imaginary part of the real-frequency Green's function $\Im G(\omega)$,

$$A(\omega) = -\frac{\Im G(\omega)}{\pi}. \tag{46}$$

From the well-known Kramers–Kronig transformation, the real part of $G(\omega)$ can be determined as well:

$$\Re G(\omega) = -\frac{1}{\pi} \int_{-\infty}^{\infty} d\omega' \frac{\Im G(\omega)}{\omega - \omega'}. \tag{47}$$

In the `HIBISCUS` component, we offer a utility program to do this job.

Analytical continuation for the self-energy function: Padé approximation

To calculate real physical quantities, such as the optical conductivity, Seebeck coefficient, electrical resistivity, etc., the self-energy function on the real axis is an essential input. With the Padé approximation [51], we can convert the self-energy function from the Matsubara frequency to the real frequency axis. We implemented the Padé approximation for $\Sigma(i\omega_n)$ in the `HIBISCUS` component.

Analytical continuation for the self-energy function: Gaussian polynomial fitting

The calculated results for the self-energy function on the real axis using the Padé approximation strongly depend on the numerical accuracy of the input self-energy data. However, the CT-HYB/DMFT calculations usually yield a Matsubara self-energy function with significant noise [52]. In this case, the Padé approximation does not work so well. To overcome this problem, K. Haule et al. [38] suggested to split the Matsubara self-energy function into a low-frequency part and a high-frequency tail. The low-frequency part is fitted by some sort of model functions which depends on whether the system is metallic or insulating, and the high-frequency part is fitted by modified Gaussian polynomials. It was shown that their trick works quite well even when the original self-energy function is noisy, and is superior to the Padé approximation in most cases. Thus, in the `HIBISCUS` component, we also implemented this algorithm. It has broad applications in the context of LDA + DMFT calculations [3].

4.5. Application programming interface

We can not only execute the components of the *i*QIST software package directly, but also invoke them from external programs. To achieve this, we provide simple application programming interfaces (APIs) for most of the components in the *i*QIST software package for the Fortran and Python languages. With these well-defined and easy-to-use APIs, one can easily set up, start, and stop the CT-HYB impurity solvers. For example, one can use the following Python script fragment to start the CT-HYB impurity solver:

```
from mpi4py import MPI        # import mpi support
from pyiqist import api as ctqmc # import python api for iQIST
...
comm = MPI.COMM_WORLD          # get the mpi communicator
ctqmc.init_ctqmc(comm.rank, comm.size) # init. ctqmc impurity solver
ctqmc.exec_ctqmc(1)           # exec. ctqmc impurity solver
ctqmc.stop_ctqmc()            # stop  ctqmc impurity solver
```

When the computations are finished, one can also collect and analyze the calculated results with Python scripts. Using these APIs, we have more freedom to design and implement very complex computational procedures. Please see Section 6.4 for more details.

5. Installation and usage

In this section, we will explain how to install and use the *i*QIST software package.

5.1. Get *i*QIST

The *i*QIST is an open source free software package. We release it under the GNU General Public Licence 3.0 (GPL). The readers who are interested in it can write a letter to the authors to request an electronic copy of the newest version of *i*QIST, or they can download it directly from the public code repository:

http://bitbucket.org/huangli712/iqist.

5.2. Build *i*QIST

In order to build and install *i*QIST successfully, a Fortran 90 compiler (MPI-enabled), BLAS, and LAPACK linear algebra libraries are necessary. The components in *i*QIST can be successfully compiled using a recent Intel Fortran compiler. Most of the MPI implementations, such as MPICH, MVAPICH, OpenMPI and Intel MPI are compatible with *i*QIST. As for the BLAS implementation, we strongly recommend OpenBLAS. For the LAPACK, the Intel Math Kernel Library is undoubtedly a good candidate. Of course, it is also possible to use the linear algebra library provided by the operating system, for example, the vecLib Framework in the Mac OS X system. Some post-processing scripts contained in the HIBISCUS component are developed using Python. In order to execute these scripts or use the Python binding for *i*QIST, one should ensure that Python 2.x or 3.x is installed. Furthermore, the latest numpy, scipy, and f2py packages are also necessary.

The downloaded *i*QIST software package is likely a compressed file with zip or tar.gz suffix. One should uncompress it at first:

```
$ tar xvfz iqist.tar.gz
```

where $ is the command line prompt. Then go to the iqist/src/build directory (in the following we just assume the top directory for *i*QIST software package is iqist) and edit the make.sys file to configure the compiling environment. One must set up the Fortran compiler, BLAS and LAPACK libraries manually:

```
$ cd iqist/src/build
$ editor make.sys
```

Once the compiling environment is configured, please type the following command in the current directory (iqist/src/build) to compile *i*QIST:

```
$ make all
```

After a few minutes (depending on the performance of compiling platform), if there are no error messages, all of the *i*QIST components are successfully compiled.

Note that what you obtain are a few standalone applications. You can execute them in the terminal directly. If you want to compile them to a library, please edit the make.sys file again to activate the API and MPY flags, and then re-compile the *i*QIST:

```
$ editor make.sys
$ make clean  (this step is optional)
$ make lib
```

At this time the libctqmc.a is generated. Then you can link it with your own Fortran programs. If you want to generate the Python binding for *i*QIST, please change the current directory to iqist/src/api:

```
$ cd ../api
```

and then use the following command to build pyiqist.so which is a valid Python module:

```
$ make pyiqist
```

5.3. Setup iQIST

Here we assume that the *i*QIST is built properly. Next we have to do one more step to finalize the installation. Please go to the iqist/bin directory and run the setup.sh:

```
$ cd iqist/bin
$ ./setup.sh
```

If everything is OK, all of the executable programs, libraries, scripts, and Python modules will be collected and copied into the iqist/bin directory. Please add this directory into the system environment variables PATH and PYTHONPATH. Now the *i*QIST is ready for use.

5.4. Use iQIST

(i) At first, since there are several CT-HYB impurity solvers in the package and their features and efficiencies are somewhat different, it is the user's responsibility to choose suitable CT-HYB components to deal with the impurity problem at hand. (ii) Second, the *i*QIST is in essence a computational engine, so the users have to prepare scripts or programs to execute the selected CT-HYB impurity solver directly or to call it using the APIs. For example, if the users want to conduct CT-HYB/DMFT calculations, they must implement the DMFT self-consistent equation by themselves (the *i*QIST software package also provide a mini DMFT self-consistent engine for the Hubbard model on the Bethe lattice). (iii) Third, an important task is to prepare proper input data for the selected CT-HYB impurity solver. The optional input for the CT-HYB impurity solver is the hybridization function $[\Delta(i\omega_n)]$, impurity level ($E_{\alpha\beta}$), interaction parameters (U, J, and μ), etc. If the users do not feed these data to the impurity solver, it will use the default settings automatically. Specifically, if the Coulomb interaction matrix is general, one should use the JASMINE component to diagonalize the local atomic problem at first to generate the necessary eigenvalues and eigenvectors. (iv) Fourth, execute the CT-HYB impurity solver. (v) Finally, when the calculations are finished, one can use the tools contained in the HIBISCUS component to post-process the output data, such as the imaginary-time Green's function $G(\tau)$, Matsubara self-energy function $\Sigma(i\omega_n)$, and other physical observables. For more details, please refer to the user manual of *i*QIST.

6. Examples

In the last few years, the *i*QIST software package has been successfully used in many projects, such as the study of the pressure-driven orbital-selective Mott metal–insulator transition in cubic CoO [53], the metal–insulator transition in a three-band Hubbard model with or without SOC [54,55], the non-Fermi-liquid behavior in cubic phase BaRuO$_3$ [56], dynamical screening effects in the electronic structure of the strongly correlated metal SrVO$_3$ and local two-particle vertex functions [57], the electronic excitation spectra of the five-orbital Anderson impurity model [58], an extended dynamical mean-field study of the 2D/3D Hubbard model with long range interactions [49], electronic structures of the topological crystalline Kondo insulators YbB$_6$ and YbB$_{12}$ [59], and superconducting instabilities of a multi-orbital system with strong SOC (doped Sr$_2$IrO$_4$) [60,61], etc. In order to illustrate the basic usage of the *i*QIST software package, we describe here several easily repeatable and simple applications of it. The testing platform is a Macbook laptop (CPU: Intel Core i7 2.3 GHz, Memory: 8 GB DDR3). We compile the *i*QIST software package using Intel Fortran Compiler 13.0.0 and the linear algebra library is Intel MKL.

6.1. Single-band Hubbard model

Here we consider the simplest case—the single-band half-filled Hubbard model on the Bethe lattice. The model parameters are: Coulomb interaction $U = 6.0$, chemical potential $\mu = 3.0$, system temperature $T = 0.1$, hopping parameter $t = 0.5$. As mentioned before, we have implemented the DMFT self-consistency condition for the Bethe lattice ($\Delta = t^2G$) [2], so we use *i*QIST to solve this model directly. The input file is as follows:

```
# file name: solver.ctqmc.in
isscf = 2    ! control the running mode, self-consistent calculation
isbin = 1    ! control the running mode, no data binning
Uc    = 6.0  ! Coulomb interaction
mune  = 3.0  ! chemical potential
beta  = 10.0 ! inversion of temperature
```

Note that the filename for the input file must be solver.ctqmc.in. Anything after the # or ! character will be considered as comments and be skipped completely. Blank lines or even a blank solver.ctqmc.in file is valid. We choose the 'key = value' or 'key : value' format to set up the computational parameters. We do not need to set up all of the computational parameters in the solver.ctqmc.in file. They all have default values. As for the detailed explanations for the file format of solver.ctqmc.in and accurate definitions of all input parameters, please refer to the corresponding user manual encapsulated in the *i*QIST software package.

Now we choose the AZALEA component to solve this model. In order to reduce the numerical noise, 4 MPI processes are used:

```
$ mpiexec -n 4 iqist/bin/azalea.x
```

it takes about 2 min to complete this task. The calculated impurity Green's function (stored in the solver.grn.dat file), which exhibits clear insulating behavior, is shown in Fig. 4. Finally, we should emphasize that the GARDENIA and NARCISSUS components are also applicable. The only thing we have to do is use gardenia.x or narcissus.x to replace azalea.x in the above command.

15
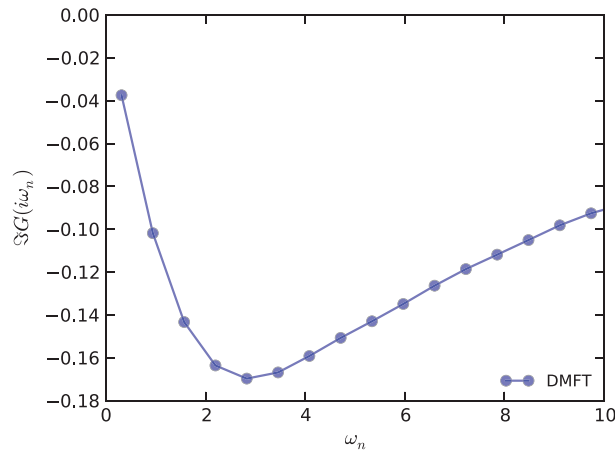
Fig. 4. Imaginary part of the impurity Green's function $\Im G(i\omega_n)$ of the single-band Hubbard model solved by DMFT. The model parameters are $U = 6.0$, $\mu = 3.0$, $\beta = 10.0$, $t = 0.5$.

6.2. Multiband Hubbard model with general Coulomb interaction

Next we consider a two-band Hubbard model with rotationally invariant interaction on the Bethe lattice. The model parameters are: Coulomb interaction $U = 6.0$, Hund's exchange $J = 1.0$, chemical potential $\mu = 6.5$, system temperature $T = 0.1$, hopping parameter $t = 0.5$.

Since the interaction term is not of density–density type anymore, we have to use the general matrix version of the CT-HYB impurity solver, i.e., the BEGONIA, LAVANDER, PANSY, or MANJUSHAKA component to solve it. The atom.cix file which contains the eigenvalues and eigenvectors of local atomic problem, are necessary for these impurity solvers. So we have to generate the atom.cix file using the JASMINE component at first. The input file for the JASMINE must be atom.config.in. The required atom.config.in file is as follows:

```
# file name: atom.config.in
nband  : 2    # number of bands
norbs  : 4    # number of orbitals (include spin index)
ncfgs  : 16   # number of atomic configurations (= 2**norbs)
nmini  : 0    # minmum occupancy
nmaxi  : 4    # maximum occupancy
Uc     : 6.00 # intraorbital Coulomb interaction
Uv     : 4.00 # interorbital Coulomb interaction
Jz     : 1.00 # z component of Hund's exchange interaction
Js     : 1.00 # spin-flip
Jp     : 1.00 # pair-hopping
```

We execute the JASMINE code in the command line:

```
$ iqist/bin/jasmine.x    (the jasmine code is not parallelized)
```

The key output files is atom.cix. Please do not modify it manually.

Here we select the BEGOINA component to solve this model. The corresponding input file looks as follows:

```
# file name: solver.ctqmc.in
isscf  : 2    ! control the running mode, self-consistent calculation
isbin  : 1    ! control the running mode, no data binning
nband  : 2    ! number of bands
norbs  : 4    ! number of orbitals (include spin index)
ncfgs  : 16   ! number of atomic configurations (= 2**norbs)
mune   : 6.50 ! chemical potential for half-filling case
beta   : 10.0 ! inversion of temperature
```

You can see that in the solver.ctqmc.in file, the parameters for the Coulomb interaction and Hund's exchange interaction are absent. This is because the information about the local interaction has been included in the atom.cix file already.

Next let us conduct the calculation using MPI:

```
$ mpiexec -n 4 iqist/bin/begonia.x
```

The running time is about 16 min. In Fig. 5, the obtained impurity Green's function is shown as a reference. In this example, we can use the LAVENDER component as well. With it we can adopt the orthogonal polynomial algorithm to improve the numerical accuracy and reduce the data noise.
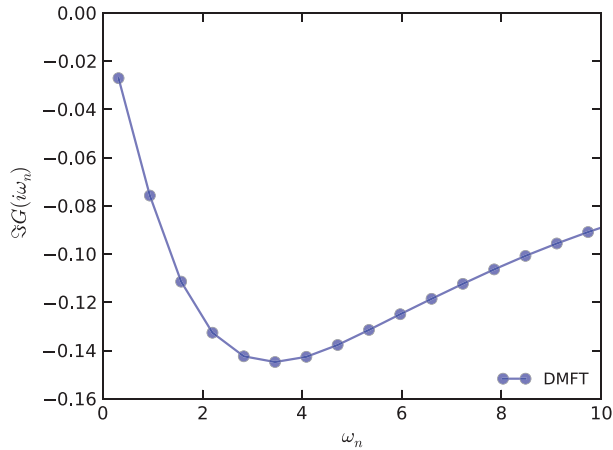
Fig. 5. Imaginary part of the impurity Green's function $\Im G(i\omega_n)$ of the two-band Hubbard model solved by DMFT. The model parameters are $U = 6.0, J_z = J_s = J_p = 1.0$, $\mu = 6.5, \beta = 10.0, t = 0.5$.

6.3. Two-particle Green's function and vertex function

In the previous two examples, DMFT self-consistent calculations are performed. Here we will show how to use *i*QIST to perform one-shot calculation to measure the two-particle Green's function and vertex function for a given impurity model.

For simplicity, we consider the same model as Section 6.1 which was solved using the AZALEA component already. The converged hybridization function $\Delta(i\omega_n)$ is stored in the solver.hyb.dat file. Please copy it to the current directory and rename it to solver.hyb.in. Next, we prepare the solver.ctqmc.in file for the CT-HYB impurity solver:

```
# file name: solver.ctqmc.in
isscf = 1    # control the running mode, one-shot calculation
isbin = 1    # control the running mode, no data binning
isvrt = 8    # calculate two-particle quantities
Uc    = 6.0  # Coulomb interaction
mune  = 3.0  # chemical potential
beta  = 10.0 # inversion of temperature
nbfrq = 1    # number of bosonic frequencies
nffrq = 128  # number of fermionic frequencies
```

Since we are going to get the two-particle Green's function $\chi(\omega, \omega', \nu)$ and vertex function $\Gamma(\omega, \omega', \nu)$, the GARDENIA component is the best (the NARCISSUS component is OK, but it is less efficient than GARDENIA). We then use the following command to invoke it:

```
$ mpiexec -n 4 iqist/bin/gardenia.x
```

After about 10 min, the calculation is finished. The calculated two-particle quantities (stored in solver.twop.dat file) are shown in Fig. 6 in which only the real part of the spin-up-up component is displayed.

6.4. Python API

In the previous examples, we always execute the CT-HYB impurity solver components directly. However, *i*QIST provides flexible and powerful APIs for the Fortran and Python languages. We can use these APIs to develop complex computational programs easily. In this subsection, we try to use the Python binding of *i*QIST to build a somewhat complicated DMFT program, and use it to study the classic Mott–Hubbard metal–insulator transition in the single-band Hubbard model. The model parameters are $U = 1.0-4.0, \mu = U/2, \beta = 50.0$, $t = 0.5$.

Here is the full source code of the Python script:

```
#!/usr/bin/env python
import numpy            # import array support
import shutil           # import high-level file operation support
from mpi4py import MPI  # import mpi support

from u_ctqmc import *   # import the writer for solver.ctqmc.in file
from pyiqist import api as ctqmc # import python module for iqist

# get mpi communicator
comm = MPI.COMM_WORLD

# set up the basic parameters and allocate memory
mfreq = 8193 # number of matsubara frequency points
norbs = 2    # number of orbitals
```
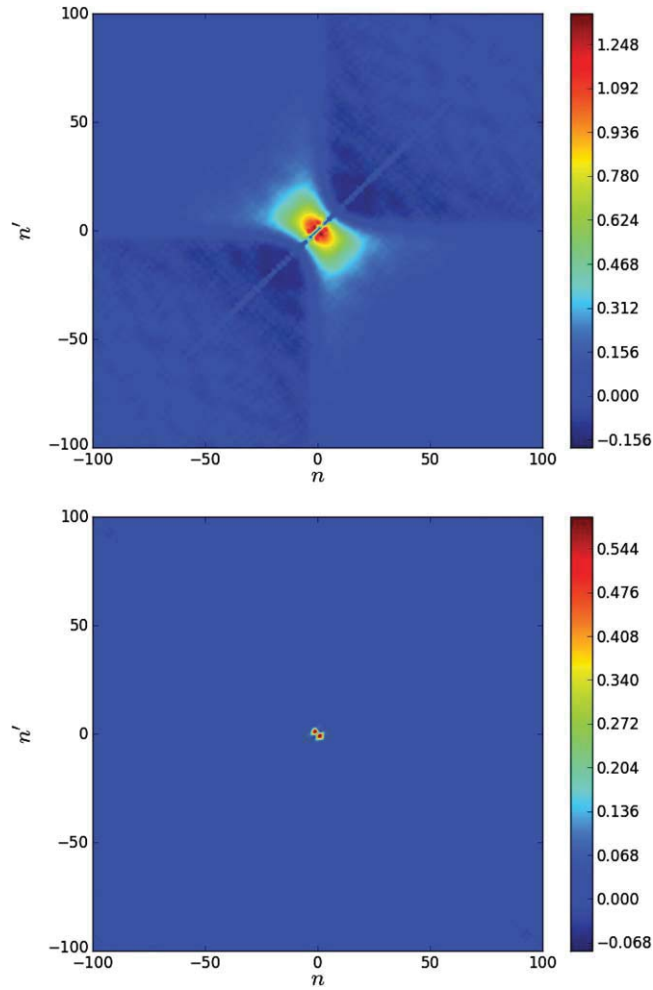
17

Fig. 6. Two-particle quantities of the single-band Hubbard model solved by DMFT. (Top) Two-particle Green's function $\Re\chi_{\uparrow\uparrow}(\omega_n, \omega_{n'}, \nu = 0)$. (Bottom) Two-particle vertex function $\Re\Gamma_{\uparrow\uparrow}(\omega_n, \omega_{n'}, \nu = 0)$. The model parameters are $U = 6.0$, $\mu = 3.0$, $\beta = 10.0$, $t = 0.5$.

```
size_t = mfreq * norbs * norbs
hybf_s = numpy.zeros(size_t, dtype=numpy.complex)

# loop over Coulomb interaction strength: from 1.0 to 4.0
for u in range(1,5):
    # build ctqmc input file: solver.ctqmc.in
    if comm.rank == 0: # only the master process can do it
        p = p_ctqmc_solver('azalea') # select impurity solver
        p.setp(isscf = 1, isbin = 1) # set up parameters
        p.setp(beta = 50.0)          # set up parameters
        p.setp(Uc = u, mune = u/2.0) # set up parameters
        p.write()                    # write solver.ctqmc.in
        del p
    comm.Barrier() # mpi barrier

    # DMFT self-consistent loop
    ctqmc.init_ctqmc(comm.rank, comm.size) # init ctqmc impurity solver
    for i in range(20): # number of iterations = 20
        ctqmc.exec_ctqmc(i+1)          # execute ctqmc impurity solver
        grnf = ctqmc.get_grnf(size_t) # get impurity Green's function
        hybf = (0.25*grnf+hybf_s)/2.0 # DMFT self-consistent condition
        hybf_s = hybf                 # update old hybridization function
        ctqmc.set_hybf(size_t, hybf)  # set up hybridization function
    ctqmc.stop_ctqmc() # stop ctqmc impurity solver
    comm.Barrier() # mpi barrier

    # save calculated results
```
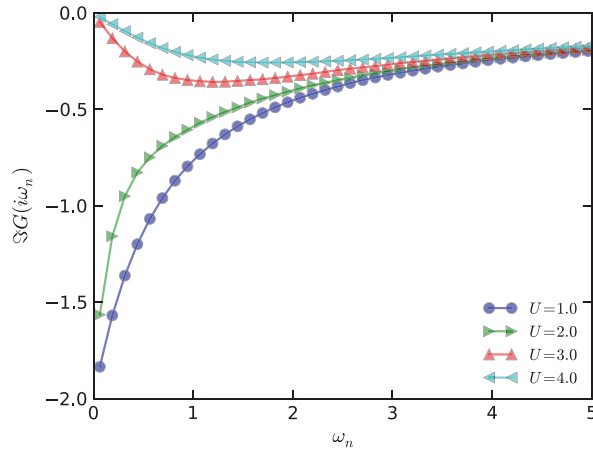
Fig. 7. Imaginary part of the impurity Green's function $\Im G(i\omega_n)$ of the single-band Hubbard model solved by DMFT. The model parameters are $\mu = U/2$, $\beta = 50.0$, $t = 0.5$.
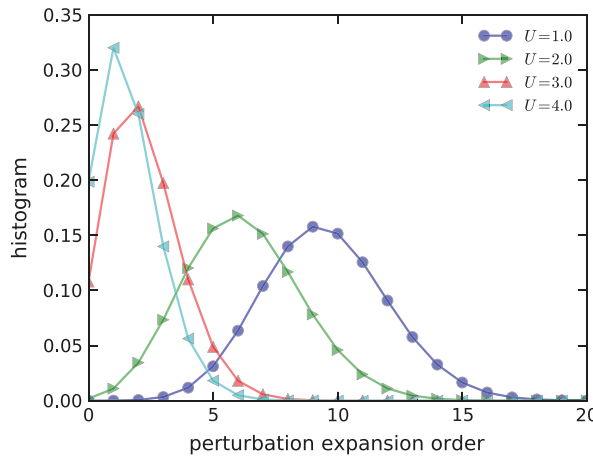

Fig. 8. Histogram of the perturbation expansion order for the single-band Hubbard model solved by DMFT. The model parameters are $\mu = U/2$, $\beta = 50.0$, $t = 0.5$.

```
if comm.rank == 0: # only the master process can do it
    shutil.move('solver.grn.dat','solver.grn.dat.'+str(u))
    shutil.move('solver.hist.dat','solver.hist.dat.'+str(u))
```

In this Python script (dmft.py), the pyiqist module contains the Python binding for *i*QIST which is introduced in Section 4.5. The u_ctqmc module which implements the p_ctqmc_solver class is included in the `HIBISCUS` component and is often used to generate solver.ctqmc.in file dynamically. The MPI parallelism is fully supported in this script via the mpi4py module. To run it, please use the following command:

```
$ mpiexec -n 4 ./dmft.py
```

It takes about half an hour to finish this job. The calculated results (the solver.grn.dat file contains the impurity Green's function, and the solver.hist.dat file contains the histogram data) are shown in Fig. 7 and Fig. 8, respectively. Clearly, between $U = 2.0$ and $U = 3.0$, a Mott metal–insulator transition induced by electronic interaction occurs. The perturbation expansion order of CT-HYB impurity solver decreases with the increment of interaction strength.

7. Future developments

In this paper, we explained and demonstrated the *i*QIST software package. *i*QIST aims to provide a complete toolkit for solving various quantum impurity systems. At first, we introduced the basic theory about quantum impurity models and the CT-QMC/CT-HYB algorithm briefly. Then various optimization tricks and algorithms implemented in *i*QIST have been discussed in detail. Following that we reviewed the software architecture and major features of *i*QIST. The compiling, setup, and workflow of *i*QIST were also illustrated. Finally, several simple examples have been shown to help the readers master the basic usage of *i*QIST step by step.

Although proven to be very versatile in applications and efficient in performance, the *i*QIST project is still a work in progress and the development will continue. The future developments of the *i*QIST project are likely to be along the following directions.

As the study of interacting electronic systems is moving towards treating their correlated multi-band nature in a more realistic fashion (5- or 7-bands, SOC included, competing multi-orbital interactions, etc.), it is important to develop even more efficient and optimized CT-HYB impurity solvers. An effective way to reduce the average size of the matrices used during the calculation is to fully consider the point group symmetry of the impurity model, which provides more GQNs to the problem. The corresponding coding work has already been started by some of the authors.

Recent developments in condensed matter theories need to be added into the features of the *i*QIST software package. For example, the measurement of entanglement entropy in realistic correlated fermion systems [62–64] will be considered, with which one will be able to explore and discovery more symmetry protected topological states and even interaction-driven topological orders that might exist in nature [65,66].

The two-particle correlation functions (susceptibilities) contain more information than the single-particle quantities, but the DMFT formalism is only self-consistent at the single-particle level. To conduct a calculation which is self-consistent both at the single- and two-particle levels is the next step in the CT-HYB/DMFT simulations. The DMFT + Parquet scheme present in Refs. [60] and [61] is the first step to incorporate correlation effects at the two-particle level beyond single-site DMFT, but it is only self-consistent at the two-particle level, and in many occasions only one-shot simulations at the two-particle level are considered due to numerical difficulties. To be fully self-consistent among single- and two-particle quantities, one still needs to employ the Schwinger–Dyson equation to feed the two-particle information back to the single-particle quantities [67,68]. This will also be a further development of the *i*QIST software package.

Instead of using single- and two-particle diagrammatic relations to capture the spatial correlation effects, one can also develop cluster CT-QMC impurity solvers, such that the spatial correlations within the cluster can be captured exactly. While in one-band models and a few two-band models cluster CT-QMC impurity solvers are available [3,4,8,69,70], generic cluster CT-QMC impurity solvers which take care of both the multi-orbital interactions within each cluster site and the spatial correlations between the cluster sites are still missing. This is also an arena for future developments.

In the end, we would like to emphasize that *i*QIST is an open initiative and the feedback and contributions from the community are very welcome.

Acknowledgments

References

[1] E. Gull, A.J. Millis, A.I. Lichtenstein, A.N. Rubtsov, M. Troyer, P. Werner, Rev. Modern Phys. 83 (2011) 349.
[2] A. Georges, G. Kotliar, W. Krauth, M.J. Rozenberg, Rev. Modern Phys. 68 (1996) 13.
[3] G. Kotliar, S.Y. Savrasov, K. Haule, V.S. Oudovenko, O. Parcollet, C.A. Marianetti, Rev. Modern Phys. 78 (2006) 865.
[4] T. Maier, M. Jarrell, T. Pruschke, M.H. Hettler, Rev. Modern Phys. 77 (2005) 1027.
[5] M.J. Rozenberg, G. Kotliar, H. Kajueter, Phys. Rev. B 54 (1996) 8452.
[6] Z.P. Yin, K. Haule, G. Kotliar, Nat. Phys. 7 (2011) 294.
[7] E. Gull, O. Parcollet, A.J. Millis, Phys. Rev. Lett. 110 (2013) 216405.
[8] K.-S. Chen, Z.Y. Meng, S.-X. Yang, T. Pruschke, J. Moreno, M. Jarrell, Phys. Rev. B 88 (2013) 245110.
[9] P. Werner, M. Casula, T. Miyake, F. Aryasetiawan, A.J. Millis, S. Biermann, Nat. Phys. 8 (2012) 331.
[10] N.S. Vidhyadhiraja, A. Macridin, C. Şen, M. Jarrell, M. Ma, Phys. Rev. Lett. 102 (2009) 206407.
[11] G. Sordi, P. Sémon, K. Haule, A.-M.S. Tremblay, Phys. Rev. Lett. 108 (2012) 216401.
[12] X. Deng, J. Mravlje, R. Žitko, M. Ferrero, G. Kotliar, A. Georges, Phys. Rev. Lett. 110 (2013) 086401.
[13] P. Werner, E. Gull, M. Troyer, A.J. Millis, Phys. Rev. Lett. 101 (2008) 166405.
[14] M. Imada, A. Fujimori, Y. Tokura, Rev. Modern Phys. 70 (1998) 1039.
[15] M. Caffarel, W. Krauth, Phys. Rev. Lett. 72 (1994) 1545.
[16] C. Gros, Phys. Rev. B 50 (1994) 7295.
[17] W. Nolting, W. Borgiel, Phys. Rev. B 39 (1989) 6962.
[18] A. Georges, G. Kotliar, Phys. Rev. B 45 (1992) 6479.
[19] N.E. Bickers, Rev. Modern Phys. 59 (1987) 845.
[20] N.E. Bickers, S.R. White, Phys. Rev. B 43 (1991) 8044.
[21] J.E. Hirsch, R.M. Fye, Phys. Rev. Lett. 56 (1986) 2521.
[22] M. Jarrell, Phys. Rev. Lett. 69 (1992) 168.
[23] A.N. Rubtsov, V.V. Savkin, A.I. Lichtenstein, Phys. Rev. B 72 (2005) 035122.
[24] E. Gull, O. Parcollet, M. Troyer, Europhys. Lett. 82 (2008) 57003.
[25] P. Werner, A. Comanac, L. de' Medici, M. Troyer, A.J. Millis, Phys. Rev. Lett. 97 (2006) 076405.
[26] P. Werner, A.J. Millis, Phys. Rev. B 74 (2006) 155107.
[27] K. Haule, Phys. Rev. B 75 (2007) 155113.
[28] A.M. Läuchli, P. Werner, Phys. Rev. B 80 (2009) 235117.
[29] L. Boehnke, H. Hafermann, M. Ferrero, F. Lechermann, O. Parcollet, Phys. Rev. B 84 (2011) 075145.
[30] H. Hafermann, K.R. Patton, P. Werner, Phys. Rev. B 85 (2012) 205106.
[31] H. Hafermann, Phys. Rev. B 89 (2014) 235128.
[32] N. Parragh, A. Toschi, K. Held, G. Sangiovanni, Phys. Rev. B 86 (2012) 155158.
[33] P. Sémon, C.-H. Yee, K. Haule, A.-M.S. Tremblay, Phys. Rev. B 90 (2014) 075149.
[34] M. Shinaoka, M. Dolfi, M. Troyer, P. Werner, J. Stat. Mech. Theory Exp. 2014 (2014) P06012.
[35] M. Ferrero, O. Parcollet, TRIQS: a Toolbox for Research in Interacting Quantum Systems, http://ipht.cea.fr/triqs.
[36] H. Hafermann, P. Werner, E. Gull, Comput. Phys. Comm. 184 (2013) 1280.
[37] B. Bauer, L.D. Carr, H.G. Evertz, A. Feiguin, J. Freire, S. Fuchs, L. Gamper, J. Gukelberger, E. Gull, S. Guertler, A. Hehn, R. Igarashi, S.V. Isakov, D. Koop, P.N. Ma, P. Mates, H. Matsuo, O. Parcollet, G. Pawowski, J.D. Picon, L. Pollet, E. Santos, V.W. Scarola, U. Schollwöck, C. Silva, B. Surer, S. Todo, S. Trebst, M. Troyer, M.L. Wall, P. Werner, S. Wessel, J. Stat. Mech. Theory Exp. 2011 (2011) P05001.
[38] K. Haule, C.-H. Yee, K. Kim, Phys. Rev. B 81 (2010) 195107.
[39] J. Kuneš, Phys. Rev. B 83 (2011) 085102.
[40] G. Rohringer, A. Valli, A. Toschi, Phys. Rev. B 86 (2012) 125114.
[41] A.N. Rubtsov, M.I. Katsnelson, A.I. Lichtenstein, Phys. Rev. B 77 (2008) 033101.
[42] A. Toschi, A.A. Katanin, K. Held, Phys. Rev. B 75 (2007) 045118.
[43] M. Matsumoto, T. Nishimura, ACM Trans. Model. Comput. Simul. 8 (1998) 3.
[44] M. Saito, M. Matsumoto, in: A. Keller, S. Heinrich, H. Niederreiter (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer Berlin Heidelberg, 2008, pp. 607–622.

[45] P. Werner, A.J. Millis, Phys. Rev. Lett. 104 (2010) 146401.
[46] T. Ayral, S. Biermann, P. Werner, Phys. Rev. B 87 (2013) 125149.
[47] P. Werner, A.J. Millis, Phys. Rev. Lett. 99 (2007) 146404.
[48] M. Jarrell, J. Gubernatis, Phys. Rep. 269 (1996) 133.
[49] L. Huang, T. Ayral, S. Biermann, P. Werner, Phys. Rev. B 90 (2014) 195114.
[50] K.S.D. Beach, arXiv:0403055 [cond-mat].
[51] H. Vidberg, J. Serene, J. Low Temp. Phys. 29 (1977) 179.
[52] N. Blümer, Phys. Rev. B 76 (2007) 205120.
[53] L. Huang, Y. Wang, X. Dai, Phys. Rev. B 85 (2012) 245110.
[54] L. Huang, L. Du, X. Dai, Phys. Rev. B 86 (2012) 035150.
[55] L. Du, L. Huang, X. Dai, Eur. Phys. J. B 86 (2013) 94. http://dx.doi.org/10.1140/epjb/e2013-31024-6.
[56] L. Huang, B. Ao, Phys. Rev. B 87 (2013) 165139.
[57] L. Huang, Y. Wang, Europhys. Lett. 99 (2012) 67003.
[58] L. Huang, T.O. Wehling, P. Werner, Phys. Rev. B 89 (2014) 245104.
[59] H. Weng, J. Zhao, Z. Wang, Z. Fang, X. Dai, Phys. Rev. Lett. 112 (2014) 016403.
[60] Z.Y. Meng, K.-S. Chen, F. Yang, H. Yao, H.-Y. Kee, 2014. arXiv:1408.1407 [cond-mat].
[61] Z.Y. Meng, Y.B. Kim, H.-Y. Kee, Phys. Rev. Lett. 113 (2014) 177003.
[62] T. Grover, Phys. Rev. Lett. 111 (2013) 130402.
[63] F.F. Assaad, T.C. Lang, F. Parisen Toldin, Phys. Rev. B 89 (2014) 125121.
[64] L. Wang, M. Troyer, Phys. Rev. Lett. 113 (2014) 110401.
[65] X. Chen, Z.-C. Gu, Z.-X. Liu, X.-G. Wen, Science 338 (2012) 1604.
[66] C. Wang, A.C. Potter, T. Senthil, Science 343 (2014) 629.
[67] S.X. Yang, H. Fotso, J. Liu, T.A. Maier, K. Tomko, E.F. D'Azevedo, R.T. Scalettar, T. Pruschke, M. Jarrell, Phys. Rev. E 80 (2009) 046706.
[68] K.-M. Tam, H. Fotso, S.-X. Yang, T.-W. Lee, J. Moreno, J. Ramanujam, M. Jarrell, Phys. Rev. E 87 (2013) 013311.
[69] K.S. Chen, Z.Y. Meng, U. Yu, S. Yang, M. Jarrell, J. Moreno, Phys. Rev. B 88 (2013) 041103.
[70] Y. Nomura, S. Sakai, R. Arita, Phys. Rev. B 89 (2014) 195146.