Reference Manual for Interacting Quantum Impurity Systems Simulating Toolkit

Core Developers:

Li Huang[†] and **Yilin Wang**[‡]

Key Contributors:

Zi Yang Meng[‡,♭] and **Liang Du**[♯]

Directors and Supervisors:

Philipp Werner[†] and **Xi Dai**[‡]

[†]Department of Physics, University of Fribourg, 1700 Fribourg, Switzerland

[‡]Beijing National Laboratory for Condensed Matter Physics, and
Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China

[♭]Department of Physics, University of Toronto, Toronto, Ontario M5S 1A7, Canada

[♯]Department of Physics, The University of Texas at Austin, Austin, Texas 78712, USA

Draft Version October 7, 2014

To my lovely wife X. Zhao

L. H

To my lovely girlfriend X.Y. Mao

Y.L. Wang

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 What's *i*QIST ?

The Interacting Quantum Impurity Solver Toolkit (dubbed iQIST) is an open source software package aiming to provide a full, reliable, flexible, and powerful tool chain for various quantum impurity models. It contains a few continuous-time quantum Monte Carlo impurity solvers (hybridization expansion version), a Hirsch-Fye quantum Monte Carlo impurity solver, and numerous prep-processed and post-processed tools. The iQIST is an all-in-one package. With it you can solve quantum impurity models and anaylyze the calculated results easily and efficiently.

1.2 Motivation

1.3 Software architecture

1.4 Main features

The iQIST is a powerful software package. It consists of many components (We just call the executable program as component in iQIST). The main components of iQIST is the continuous-time quantum Monte Carlo impurity solvers. So far these impurity solvers support the following features:

* Density-density interaction * General interaction * SOC interaction * Hubbard model and Hubbard-Holstein model * Frequency-dependent interaction * Orthogonal polynomial representation * Kernel polynomial representation * Improved estimator for self-energy * Single-particle Greens function $G(\tau)$ * Single-particle Greens function $G(i\omega_n)$ * Two-particle correlation function $\chi(\omega, \omega', \nu)$ * Local irreducible

vertex function $\Gamma(\omega, \omega', \nu)$ * Self-energy function $\Sigma(i\omega_n)$ * Histogram of perturbation expansion order * Kinetic and potential energies * (Double) occupation numbers, magnetic moment * Atomic state probability * Spin-spin correlation function * Orbital-orbital correlation function * Autocorrelation function and autocorrelation time * Divide-and-conquer algorithm * Sparse matrix multiplication * Good quantum numbers * Skip listing trick * Lazy trace evaluation * Dynamical truncation approximation

1.5 Development history

v0.2.0 // Aug ————-

* add bin/clean.sh * add doc/guide support in src/build/Makefile

v0.1.9 // Aug 18, 2014 ——————-

* make new building/compiling system (src/build). * make setup shell script (bin/). * update CSSL code (src/common/s_vector.f90). * refine azalea code (src/ctqmc/azalea). * refine ctqmc api (src/ctqmc/api).

v0.1.8 // Aug 4, 2014 ——————

* add pansy code (experimental). * add manjushaka code (experimental). * add jasmine code (experimental). * implement CSSL and CSML codes (experimental).

v0.1.7 // Jul 2, 2014 ——————

* add entropy code. * add stochastic code. * add swing code. * add toolbox code.

v0.1.6 // Jul 2, 2014 ——————

* add iris code.

v0.1.5 // Jul 2, 2014 ——————

* add daisy code.

v0.1.4 // Jul 2, 2014 ——————

* add lavender code.

v0.1.3 // Jul 2, 2014 ——————

* add begonia code.

v0.1.2 // Jul 2, 2014 ——————

* change the file mode for gardenia code.

v0.1.1 // Jul 2, 2014 ——————

* change the file mode for azalea code.

v0.1.0 // Jul 2, 2014 ——————

* init the whole directory structure.

v0.0.0 // Jul 2, 2014 ——————

* init the project.

1.6 Policy and licences

The iQIST software package is released under the General Public Licence 3.0 (GPL) or later version.

We are sorry. We DO NOT provide any technical support now. If you meet some problems when you are using iQIST. You can write a letter to us. But we can not guarantee we will reply you.

Chapter 2

INSTALLATION

2.1 Obtain

2.2 Uncompress

2.3 Direcrory structures

2.4 Compiling environment

2.5 Compiling system

In order to compile and install iQIST correctly, you should ensure the following softwares are correctly installed and configured in your OS.

* Intel Fortran compiler * MPICH2 or OpenMPI * BLAS * LAPACK * Python 2.X * scipy, numpy, and f2py

The downloaded iQIST software package is likely a compressed file with zip or tar.gz suffix. The users should uncompress it at first. And then go to the iqist/src/build directory, edit the make.sys file to configure the compiling environment. Once the compiling environment is configured, please run the make command in the top-level directory of iQIST. After a few minutes (depending on the performance of compiling platform), the iQIST is ready for you. Note that all of the executable programs will be copied into the iqist/bin directory automatically. Please add this directory into the system environment variable PATH.

2.6 Build impurity solvers

2.7 Build auxiliary tools

2.8 Build documents

2.9 Build application programming interfaces

Chapter 3

RUNNING

3.1 Configure your system

3.2 Create input files

3.3 Execute codes

3.4 Monitor and Profile

Chapter 4

STANDARD INPUT FILES

Chapter 5

STANDARD OUTPUT FILES

5.1 Terminal output

5.1.1 out.dat

5.2 File output

5.2.1 solver.green.dat

5.2.2 solver.green.bin

5.2.3 solver.weiss.dat

5.2.4 solver.hybrid.dat

5.2.5 solver.grn.dat

5.2.6 solver.wss.dat

5.2.7 solver.hyb.dat

5.2.8 solver.sgm.dat

5.2.9 solver.hub.dat

5.2.10 solver.nmat.dat

5.2.11 solver.schi.dat

5.2.12 solver.ochi.dat

5.2.13 solver.twop.dat

5.2.14 solver.vrtx.dat

Chapter 6

PARAMETERS

6.1 isscf

6.2 issun

6.3 isspn

6.4 isbin

6.5 isort

6.6 isvrt

6.7 isscr

6.8 nband

6.9 nspin

6.10 norbs

6.11 ncfgs

6.12 nzero

6.13 nvect

6.14 nhmat

Chapter 7

AUXILIARY TOOLS

7.1 $\mathcal{J}$asmine component

7.2 $\mathcal{H}$ibiscus component

7.2.1 Maximum entropy method: entropy1

7.2.2 Maximum entropy method: entropy2

7.2.3 Stochastic analytical continuation: sac

7.2.4 Analytical continuation for self-energy: swing

7.2.5 toolbox/makechi

7.2.6 toolbox/makedos

7.2.7 toolbox/makekra

7.2.8 toolbox/makescr

7.2.9 toolbox/makesig

7.2.10 toolbox/makestd

7.2.11 toolbox/maketau

7.2.12 toolbox/makeups

7.2.13 script/pysci.py

7.2.14 script/check.py

7.3 Parquet component

Chapter 8

APPLICATION PROGRAMMING INTERFACES

8.1 Fortran binding

8.2 Python binding

8.3 iqist.py

Chapter 9

iQIST IN ACTION

9.1 Basic applications

9.1.1 Hello *iQIST* !

9.1.2 Mott metal-insulator transition

9.2 Advanced applications I: Complex systems

9.2.1 General Coulomb interaction

9.2.2 Spin-orbital coupling

9.2.3 Crystal field splitting

9.2.4 Retarded interaction and dynamical screening effect

9.3 Advanced applications II: Accurate measurement of physical observables

9.3.1 One-shot and self-consistent calculations

9.3.2 Data binning mode

9.3.3 Imaginary-time Green's function

9.3.4 Matsubara Green's function and self-energy function

9.3.5 Spin-spin correlation function and orbital-orbital correlation function

9.3.6 Two-particle Green's function and vertex function

9.4 Advanced applications III: post-processing procedures

9.4.1 Analytical continuation for imaginary-time Green's function

9.4.2 Analytical continuation for Matsubara self-energy function

9.5 Practical exercises

9.5.1 Orbital-selective Mott transition in two-band Hubbard model

9.5.2 Orbital Kondo and spin Kondo effects in three-band Anderson impurity

Chapter 10

INSIDE iQIST

10.1 Basic theory and methods

10.1.1 Quantum impurity model

10.1.2 Principles of continuous-time quantum Monte Carlo algorithm

10.1.3 Hybridization expansion

10.1.4 Physical observables

10.1.5 Two-particle measurements and DMFT + Parquet formalism

10.2 Implementations and optimizations

10.2.1 Development platform

10.2.2 Orthogonal polynomial representation

10.2.3 Improved estimator for the self-energy function

10.2.4 Random number generators

10.2.5 Subspaces and symmetry

10.2.6 Truncation approximation

10.2.7 Lazy trace evaluation

10.2.8 Divide-and-conquer and sparse matrix tricks

10.2.9 Parallelization

Appendix

A.1 TODO