cttg Documentation

Release 3.0.1

Charles-David Hebert, Maxime Charlebois, Patrick Semon

Jan 17, 2018

CONTENTS

CHAPTER

ONE

INTRODUCTION

cttg stands for *Continuous time Tremblay group* and regroups two main Continuous-time quantum monte-carlo algorithms, namely CT-INT and CT-AUX.

1.1 Warning: Bad Docs

The documentation has only been started very recently. It is thus full of language errors, probably wrong at certain places and of poor quality. However, the quality will increase overtime.

1.2 Conventions

1.2.1 Comments

We use the convention "$" for the start of a shell command and "#" for commenting shell commands, etc.

Ex: $ cd path/to/thing # change to the path of thing.

1.2.2 Executable names

When not specified, the algorithm is CT-INT. if there is "aux" in the name, then CT-AUX. If "sub" in name, then submatrix algorithm, else normal fast-update scheme.

1.2.3 Hamiltonian

$$H = +\sum_{ij} t_{ij} + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$

Thus, for the cuprates, the convention of the program is

- t = -1.0
- t' = 0.3
- t" = -0.2

CHAPTER

TWO

INSTALLATION

2.1 Dependencies

1. Armadillo
2. boost (mpi, serialization, filesystem, system)

2.2 Pre-Steps

1. Make sure you have a “bin” directory in your home folder
2. Append the bin folder to your path. Add the following line to your ~/.bashrc: export PATH=”$PATH:~/bin”
3. $ source ~/.bashrc

2.3 Linux (Ubuntu 16.04)

This installation procedure should work for many recent Linux flavors. For the following we present the instructions specific for Ubuntu or derivatives.

1. **Install the Dependencies** $ sudo apt-get install libarmadillo-dev libboost-all-dev
2. $ mkdir build && cd build && cmake -DTEST=OFF .. && make -j NUMBER_OF_CORES install
   # replace NUMBER_OF_CORE by say = 4

2.4 Mac

This has not been tested yet, works?

1. **Install the Dependencies** $ homebrew install armadillo boost
2. $ mkdir build && cd build && cmake -DHOME=OFF -DMAC=ON .. && make -j NUMBER_OF_CORES install
   # replace NUMBER_OF_CORE by say = 4

2.5 Mp2

1. $ module reset
2. $ module load cmake/3.6.1 gcc/6.1.0 intel64/17.4 boost64/1.65.1_intel17 openmpi/1.8.4_intel17 armadillo/8.300.0
3. $ mkdir build && cd build && cmake -DHOME=OFF -DMP2=ON -DMPI_BUILD=ON .. && make install

2.6 Graham and Ceder

1. $ module reset
2. $ module load nixpkgs/16.09 gcc/5.4.0 armadillo boost-mpi
3. $ mkdir build && cd build && \
   cmake -DHOME=OFF -DGRAHAM=ON -DMPI_BUILD=ON .. && make install

CHAPTER

THREE

TUTORIAL

3.1 Graham: Tutorial 1

1. Connect to Graham:
   $ ssh -X "user"@graham.computecanada.ca # where "user" is your compute canada/Mp2 Username
2. Ensure you have done the Pre-Steps described in *Installation*.
3. **Start an interactive job:** $ salloc –time=01:00:00 –ntasks=1 –mem-per-cpu=4000
4. Follow th Graham Procedure in *Installation*.
5. $ cd examples/Graham
6. $ module reset
7. $ module load nixpkgs/16.09 gcc/5.4.0 armadillo boost-mpi
8. $ cd examples/CDMFT
9. $ cdmft_square4x4 params 1

3.2 Home

3.2.1 Home: Tutorial 1 = dmft

To Come. For now, go into the "examples folder", select your installation and run the bash script. If there are some problems, then install "dos2unix" and run it on the bash files.

> $dos2unix runCDMFT

Ex: If you installed on your computer 1. $ cd examples/Home 2. $ bash runCDMFT.sh # in fact runs dmft

3.2.2 Home: tutorial 2 = cdmft_square4x4

1. $ cd examples/CDMFT
2. **copy the script file, for example if on Home:** $ cp ../Home/runCDMFT ./
3. **in "runCDMFT", replace the line:** myExe=dmft -> myExe=cdmft_square4x4
4. $ bash runCDMFT

3.2.3 Launching a simulation

To launch a simulation, you need three files:

1. a script file, dependant on the platform (home, mp2, graham-cedar)
2. a "params" file (Ex: params1.json)
3. a "hyb" file (Ex: hyb1.arma)

3.3 When to use which algorithm

CT-Aux and CT-INT behave in a similar manner, and from my experience, one is not much faster than the other.

Cocerning Submatrix Updates, the codes "..._sub" should be used when the expansion order is high, say k>1000 For k~<500, the algorithm may be slower.

Ex: cdmft_square4x4 for k < 1000 cdmft_square4x4_int_sub for k > 1000

CHAPTER

FOUR

THE PARAMS FILE

4.1 Fast-Update Scheme

4.1.1 Main Paremeters

The following parameters are the ones that the user has to change and need to understand. There is a bit of unconsistency, to be corrected (some are lower case, while other are upper case.)

SEED the seed for the random number generator.

PROBFLIP the ratio of updates that are spin flips for the Auxiliary spins. Should be bewteen 0.0 and 0.2. 0.1 is reasonable. A non zero number will help the convergence speed.

beta inverse temperature

NMAT the cutoff of matsubara frequencies in energy, 100 is fine

NTAU the time discretization for G(tau) and for binning measurements. Normally a value of 1000 is sufficient, but, for low temperatures and big NMAT, a higher value is neccessary to get unbiaised results.

UPDATESMEAS The numbre of Updates proposed bewteen each measurement. This value should be approximately equal to the average expansion order. Updates proposed bewteen measurements = UPDATESMEAS

THERMALIZATION_TIME the time for which each processor will thermalize. It is difficult to give a good optimal value. I would say, ~10% of the measurement time.

MEASUREMENT_TIME The time each processor measures.

PRECISE If set to true, then the measure of the green function is more precise, but slower. Should be generally set to false, unless analytic continuation is neccessary. Even then, the solution should be converged before setting to 'true'

4.1.2 Implementation detailed Parameters

These parameters can be left at their current value, i.e they are implementation details. Make sure you understand what you are doing before changing them.

CLEANUPDATE Specifies when to perform a clean update. Ex, if =100, than at each 100 measures, a cleanupdate will be performed. 100 is a good number. does not substantially influence the simulation, except if this number is to low or to high. "K": The value of the K parameter of CT-Aux. Influences the acceptance rate and the expansion order 1 seems a reasanable value

THERM_FROM_CONFIG if true, there will be no thermalization, the last saved configuartion will be loaded and the measurements will start. Not really tested yet. So the default is false. André-Marie argues that it is better to thermalize each time.

n If this parameter is in the params file, than the program will change the chemical potential to attain the given value.

4.2 Submatrix Update Scheme

For the *Submatrix* algorithm, only one parameter has a differnet meaning than in the "Normal" fast-update algorithm, it is the UPDATESMEAS parameter.

UPDATESMEAS For the Submatrix update scheme, the number of updates proposed bewteen each measurement is given by this parameter multipled by KMAX_UPD. I.E: submatrix scheme: Updates proposed bewteen measurements = UPDATESMEAS * KMAX_UPD. For exemple, if the average expansion order is k = 1000, than you should set something like: KMAX_UPD = 100, UPDATESMEAS = 10.

KMAX_UPD the maximum number of updates proposed for each internal iteration (implementation details). This parameter should be ~100 on MP2, ~125 on Graham, and has a different optimal value for different architectures.