# Documentation for the General QMC code

Martin Bercx

July 19, 2016

## 1 Using the code

*Example simulation, tutorial: where to find and how to start*

### 1.1 Parameter files

*describe the input parameters, give sample values for the stabilization parameters*

### 1.2 Analysis files

*how the analysis of Monte Carlo data is done*

## 2 List of files

*all files that constitute the code, with a brief description*

### 2.1 cgr1.f90 & cgr2.f90

Stable computation of the physical single-particle equal time Green function $G(\tau)$.

### 2.2 control_mod.f90

Includes a set of auxilliary routines, regarding the flow of the simulation. Examples are initialization of performance variables, precision tests and controlled termination of the code.

### 2.3 gperp.f90

### 2.4 Hamiltonian_Hub.f90

Here, the physical simulation parameters (the model parameters) and the lattice parameters are read in. The lattice, the non-interacting and the interacting part of the Hubbard Hamiltonian are set according to the parameters.

### 2.5 Hop_mod.f90

### 2.6 inconfc.f90

The auxiliary-field QMC method is based on a Hubbard-Stratonovich decomposition of the interaction term. This decomposition introduces a space-time array of (discrete) configurations of auxiliary fields, i.e. Ising spins. In this routine, an existing configuration is read in, check on the dimensionality are made and, in case no prior configuration exist, a random configuration of Ising spins is set up.

### 2.7 main.f90

### 2.8 nranf.f90

### 2.9 Operator.f90

### 2.10 outconfc.f90

### 2.11 print_bin_mod.f90

### 2.12 tau_m.f90

### 2.13 truncation.f90

### 2.14 UDV_WRAP.f90

### 2.15 upgrade.f90

### 2.16 wrapgrdo.f90 & wrapgrup.f90

Single-particle equal-time Green functions are the central object of the code. The physical single-particle equal-time Green function $G(\tau)$ is updated in wrapgrup.f90 (up propagation, from $\tau = 0$ to $\tau = LTROT$) and in wrapgrdo.f90 (down propagation, from $\tau = LTROT$ to $\tau = 0$). The update is sequentially, over all (interacting) lattice sites or lattice bonds.

### 2.17 wrapul.f90 & wrapur.f90

To stabilize the simulation at the time slice $\tau_2 = in_{stab}$, the Green function has to be recomputed regularly, based on the stable matrices at $\tau_1 = (i-1)n_{stab}$. These stable matrices result from a singular-value-decomposition of the propagation matrix. They are computed in wrapur.f90 (up propagation, from $\tau = 0$ to $\tau = LTROT$) and wrapul.f90 (down propagation).

## 3 Module Hamiltonian

*Detailed description of the module Hamiltonian since it will be modified by the users*

The module contains the following subroutines:

### 3.1 ham_set

It calls the subroutines

- ham_latt
- ham_hop
- ham_v

It reads in the file

- parameters

It sets the variables `ltrot,n_fl,n_sun`. If compiled as a MPI-program, it broadcasts all variables that define the lattice, the model and the simulation process.

### 3.2 ham_latt

It sets the lattice, by calling the subroutine

- make_lattice(l1_p, l2_p, a1_p, a2_p, latt)

### 3.3 ham_hop

Setup of the hopping amplitudes between the vertices of the graph (lattice sites and unit cell orbitals). It calls the subroutines

- op_make(op_t(nc,n),ndim
- op_set(op_t(nc,n))

### 3.4 ham_v

It calls the subroutines

- op_make(op_v(i,nf),1)
- op_set( op_v(nc,nf) )

### 3.5 s0(n,nt)

It is defined as $s0(n, nt) = 1.d0$. Why? It is superfluous.

### 3.6 alloc_obs(ltau)

Allocation of equal time and time-resolved quantities.

### 3.7 init_obs(ltau)

Initializes equal time and time-resolved quantities with zero.

### 3.8 obser(gr,phase,ntau)

Includes the definition of all equal-time observables (scalars and correlation functions) that are built from the single-particle Green function based on Wick's theorem.

### 3.9 pr_obs(ltau)

Output (print) of the observables.

### 3.10 obsert(nt,gt0,g0t,g00,gtt,phase)

Includes the definition of time-resolved observables that are built from the time-resolved single-particle Green function based on Wick's theorem.

## 4 Necessary background information

### 4.1 Definition of the physical Hamiltonian and its implementation

The physical Hamiltonians that we can simulate have the general form:

$$\mathcal{H} = \sum_{s=1}^{N_{fl}} \sum_{\boldsymbol{x},\boldsymbol{y}} c^{\dagger}_{\boldsymbol{x}s} M_{\boldsymbol{x}\boldsymbol{y}} c_{\boldsymbol{y}s} - \sum_{k=1}^{M} U_k \left[ \sum_{s=1}^{N_{fl}} \sum_{\boldsymbol{x},\boldsymbol{y}} \left( c^{\dagger}_{\boldsymbol{x}s} T^{(k)}_{\boldsymbol{x}\boldsymbol{y}} c_{\boldsymbol{y}s} - \alpha_k \right) \right]^2 . \tag{1}$$

The indices $\boldsymbol{x}, \boldsymbol{y}$ are multi-indices that label sites and spin states: $\boldsymbol{x} = (i, \sigma)$, where $i = 1, \cdots N_{sites}$ and $\sigma = 1, \cdots N_{sun}$, so

$$\sum_{\boldsymbol{x},\boldsymbol{y}} \equiv \sum_{i=1,j=1}^{N_{sites}} \sum_{\sigma=1,\sigma'=1}^{N_{sun}} . \tag{2}$$

Note, that we introduced *two* different labels for the number of spin states (flavours): $N_{fl}$ and $N_{sun}$. The number of correlated sites which is a subset of all sites, is labelled by $M$ ($M \leq N_{sites}$).

I suggest to use a more intuitive notation and to label the hopping matrix by $T$ and the interaction matrix by $V$:

$$\mathcal{H} = \sum_{s=1}^{N_{fl}} \sum_{\boldsymbol{x},\boldsymbol{y}} c^\dagger_{\boldsymbol{x}s} T_{\boldsymbol{x}\boldsymbol{y}} c_{\boldsymbol{y}s} - \sum_{k=1}^{M} U_k \left[ \sum_{s=1}^{N_{fl}} \sum_{\boldsymbol{x},\boldsymbol{y}} \left( c^\dagger_{\boldsymbol{x}s} V^{(k)}_{\boldsymbol{x}\boldsymbol{y}} c_{\boldsymbol{y}s} - \alpha_k \right) \right]^2 , \tag{3}$$

### 4.2 Which information does the type *operator* contain?

## 5 Tutorial to set up the Hubbard model

The $SU(2)$ symmetric Hubbard model is given by

$$\mathcal{H} = -t \sum_{\langle i,j \rangle,\sigma} \left( c^\dagger_{i,\sigma} c_{j,\sigma} + \text{H.c.} \right) + \frac{U}{2} \sum_i \left[ \sum_\sigma (c^\dagger_{i\sigma} c_{i\sigma}) - 1 \right]^2 . \tag{4}$$

To bring Eq. (4) in the general form (3), we define:

$$\begin{aligned} N_{fl} &= 1 \\ N_{sun} &= 2 \\ T_{\boldsymbol{x}\boldsymbol{y}} &= -t \delta_{\langle i,j \rangle} \delta_{\sigma,\sigma'} \\ M &= N_{sites} \\ U_k &= -U/2 \\ V^{(k)}_{\boldsymbol{x}\boldsymbol{y}} &= \delta_{i,j} \delta_{i,k} \delta_{\sigma,\sigma'} \\ \alpha_k &= 1/(N_{sites} N_{sun})^2 . \end{aligned} \tag{5}$$

## 6 Tutorial to set up a lattice

## 7 Installation

### 7.1 Dependencies

*which software and libraries are needed and which version*

- libraries: LAPACK, BLAS, EISPACK, NAG *They are included in the package, but NAG is not public-domain (?)*
- tools: cmake
- compiler: gfortran or ifort

### 7.2 Build the GQMC program from source code

*configuration, compile and Installation* In the top level directory, where the README file resides, do:

```
mkdir build
cd build
cmake ..
make
```

# 8 Reference manual

# 9 License

Use of the GQMC code requires citation of the paper ... The GQMC code is available for academic and non-commercial use under the terms of the license ... For commercial licenses, please contact the GQMC development team.

# 10 ideas

FAQ, walkthroughs,