# Documentation for the General QMC code

Author: Martin Bercx

November 4, 2016

## Contents

## 1 Definition of the model Hamiltonian

The class of solvable models includes Hamiltonians $\mathcal{H}$ having the following general form: $\mathcal{H} = \mathcal{H}_T + \mathcal{H}_V$, where

$$\mathcal{H}_T = \sum_{s=1}^{N_{fl}} \sum_{\sigma=1}^{N_{col}} \sum_{x,y} c^{\dagger}_{x\sigma s} T^{(s)}_{xy} c_{y\sigma s} \tag{1}$$

$$\mathcal{H}_V = -\sum_{k=1}^{M} U_k \left\{ \sum_{s=1}^{N_{fl}} \sum_{\sigma=1}^{N_{col}} \left[ \sum_{x,y} \left( c^{\dagger}_{x\sigma s} V^{(ks)}_{xy} c_{y\sigma s} \right) - \alpha_{ks} \right] \right\}^2 . \tag{2}$$

The indices have the following meaning:

- The number of fermion *flavors* is set by $N_{fl}$.
- The number of fermion *colors* is set by $N_{col}$. Does it set the symmetry group of the fermions, namely the dimension of the special unitary group $SU(N_{sun})$?
- The indices $x, y$ label lattice sites where $x, y = 1, \cdots N_{dim}$. $N_{dim}$ is the total number of spacial vertices: $N_{dim} = N_{unit\ cell} N_{orbital}$, where $N_{unit\ cell}$ is the number of unit cells of the underlying Bravais lattice and $N_{orbital}$ is the number of (spacial) orbitals per unit cell Check the definition of $N_{orbital}$ in the code.
- Therefore, the matrices $\boldsymbol{T}^{(s)}$ and $\boldsymbol{V}^{(ks)}$ are of dimension $N_{dim} \times N_{dim}$
- The number of correlated sites which is a subset of all sites, is labelled by $M$ ($M \leq N_{dim}$). Be more general here and speak of correlated blocks?

Note that the matrices $\boldsymbol{T}^{(s)}$ and $\boldsymbol{V}^{(ks)}$ explicitly depend on the flavor index $s$ but not on the color index $\sigma$. Using this symmetry property is essential for an efficient code implementation. The color index $\sigma$ only appears

- in the coupling $g$ in the `Operator` structure (see Sec. **??**).
- as normalization constant in the definition of observables (see Sec. **??**)
- as exponent in the calculation of the phase factor and the Monte Carlo update ratio.

### 1.1 Structure of the hopping matrix T and the interaction matrices $\mathbf{V}^{(k)}$

In general, the matrices $\mathbf{T}^{(s)}$ and $\mathbf{V}^{(ks)}$ are sparse matrices. This property is used to minimize computational cost and storage. In the following, we discuss the implementation of the single-particle matrix representation $\mathbf{V}^{(ks)}$ of the interaction operator. The same logic applies for the hopping matrix $\mathbf{T}^{(s)}$.

We denote a subset of $N_{eff}$ (in the code, $N_{eff}$ is called just $N$) degrees of freedom here: sites by the set $[z_1, \cdots z_{N_{eff}}]$ and define it to contain only vertices for which an interaction term is defined:

$$V^{(ks)}_{xy} \neq 0 \quad \text{only if} \quad x, y \in [z^{(ks)}_1, \cdots z^{(ks)}_{N^{(ks)}_{eff}}] . \tag{3}$$

We define the projection matrices $\mathbf{P}^{(ks)}_V$ of dimension $N^{(ks)}_{eff} \times N_{dim}$:

$$(P^{(ks)}_V)_{i,z} = \delta_{z^{(ks)}_i, z} , \tag{4}$$

where $i \in [1, \cdots N^{(ks)}_{eff}]$ and $z \in [1, \cdots N_{dim}]$. The matrix operator $\boldsymbol{P}^{(ks)}_V$ picks out the non-vanishing entries of $\boldsymbol{V}^{(ks)}$, which are contained in the rank-$N^{(ks)}_{eff}$ matrix $\boldsymbol{O}^{(ks)}_V$:

$$\boldsymbol{V}^{(ks)} = \boldsymbol{P}^{(ks)T}_V \boldsymbol{O}^{(ks)}_V \boldsymbol{P}^{(ks)}_V , \tag{5}$$

and

$$V^{(ks)}_{xy} = (P^{(ks)}_V)_{ix} \left[O^{(ks)}_V\right]_{ij} (P^{(ks)}_V)_{jy} = \sum_{i,j}^{N^{(ks)}_{eff}} \delta_{z^{(ks)}_i, x} \left[O^{(ks)}_V\right]_{ij} \delta_{z^{(ks)}_j, y} \; . \tag{6}$$

Comment that the P matrices have only one non-vanishing entry per column. To set the interaction part, we therefore have to specify the following:

- the matrix elements $\left[O^{(k)}_V\right]_{ij}$
- the set $[z^{(k)}_1, \cdots z^{(k)}_{N^{(k)}_{eff}}]$
- the interaction strenghts $U_k$
- the numbers $\alpha_k$.

Be more specific here what really has to specified in the actual code. The same logic also applies to the implementation of the hopping interaction be more specific.

### 1.2 The Hubbard-Stratonovich decomposition

Consider a single-particle (in other words bilinear) operator $O_i$. One obtains an approximation to the evolution operator by the following series expansion [?]

$$e^{-\Delta\tau O_i^2} = \sum_{s=\pm1,\pm2} \gamma(s) e^{i\sqrt{\Delta\tau}\eta(s) O_i} + \mathcal{O}(\Delta\tau^4) \; , \tag{7}$$

with

$$\begin{aligned} \gamma(\pm1) = (1+\sqrt{6}/3)/4 \; , \; \gamma(\pm2) = (1-\sqrt{6}/3)/4 \; , \\ \eta(\pm1) = \pm\sqrt{2(3-\sqrt{6})} \; , \; \eta(\pm2) = \pm\sqrt{2(3+\sqrt{6})} \; . \end{aligned} \tag{8}$$

Eq. (7) can be easily proven by expanding its right hand side to eighth order in $O_i$. The transformation introduces therefore two Ising fields $s$ per lattice site $i$, taking the values $\pm1$ and $\pm2$. same label as the flavor index

## 2 Implementation of a model Hamiltonian

To implement a Hamiltonian which belongs to the class of Hamiltonians defined by Eq. (1), the user only has to write/modify a single subroutine. A template is given by `Hamiltonian_template.f90`. Existing model subroutines are `Hamiltonian_Hubb.f90`.

### 2.1 The `Operator` variable

In the code implementation, we define a structure called `Operator`. This structure variable `Operator` bundles several components that are needed to define and use an operator matrix in the program. In Fortran a structure variable like this is called a derived type. The components it contains are:

- the projector $\boldsymbol{P}_V$,
- the matrix $\boldsymbol{O}_V$,
- the effective dimension $N_{eff}$,
- and a couple of auxiliary matrices and scalars.

In general, we will not only have one structure variable `Operator`, but a whole array of these structures.

| Name of variable in the code | Description |
|---|---|
| `Op_V%N` | effective dimension $N_{eff}$ |
| `Op_V%O` | matrix $\mathbf{O}_V$ |
| `Op_V%U` | matrix containing the eigenvectors of $\mathbf{O}_V$ |
| `Op_V%E` | eigenvalues of $\mathbf{O}_V$ |
| `Op_V%P` | projection matrix $\mathbf{P}_V$ |
| `Op_V%N_non_zero` | number of non-vanishing eigenvalues of $\mathbf{O}_V$ |
| `Op_V%g` | coupling strength in Hubbard-Stratonovich transformation |
| `Op_V%alpha` | constant |
| `Op_V%type` | integer parameter to set the type of Hubbard-Stratonovich transformation Possible Issue: type is also a Fortran keyword |

Table 1: Components of the `Operator` structure variable `Op_V`.

## 2.2 The observables

## 2.3 The lattice

## 3 Input and output files

## 4 Walkthrough: the $SU(2)$-Hubbard model on a square lattice

In this section, we describe the subroutine `Hamiltonian_Hub.f90` which is an implementation of the Hubbard model on the square lattice. The $SU(2)$-symmetric Hubbard model is given by

$$\mathcal{H} = -t \sum_{\sigma=1}^{2} \sum_{\langle x,y \rangle} \left( c^{\dagger}_{x\sigma} c_{y\sigma} + \text{H.c.} \right) + \frac{U}{2} \sum_{x} \left[ \sum_{\sigma=1}^{2} \left( c^{\dagger}_{x\sigma} c_{x\sigma} - 1/2 \right) \right]^2 . \tag{9}$$

In order to bring the general Hamiltonian (1) to this form, we set

$$\begin{aligned} N_{fl} &= 1 \\ N_{col} &= 2 \\ T_{xy} &= -t\delta_{\langle x,y \rangle} \\ M &= N_{dim} \\ U_k &= -\frac{U}{2} \\ V^{(ks)}_{xy} &= \delta_{x,y}\delta_{x,k} \\ \alpha_{ks} &= \frac{1}{2} . \end{aligned} \tag{10}$$

Note that in this example $N_{dim} = N_{unit\ cell} = Latt\%N$. And since $N_{fl} = 1$ for $SU(N)$-symmetric Hubbard models, we will drop the flavor index $\sigma$ in the following.

### 4.1 Hopping term

The hopping matrix is implemented as follows. We allocate an array of dimension $1 \times 1$, called `Op_T`. It therefore contains only a single `Operator` structure. We set the effective dimension for the hopping term: $N_{eff} = N_{dim}$. And we allocate and initialize this structure by a single call to the subroutine `Op_make`:

```
call Op_make(Op_T(1,1),Ndim)
```

Since the effective dimension is identical to the total dimension, it follows trivially, that $\boldsymbol{P}_T = \mathbb{1}$ and $\boldsymbol{O}_T = \boldsymbol{T}$. Note that although a checkerboard decomposition is not yet used for the Hubbard model, in principle it can be implemented.

## 4.2 Interaction term

To implement this interaction, we allocate an array of `Operator` structures. The array is called `Op_V` and has dimensions $N_{dim} \times N_{fl} = N_{dim} \times 1$. We set the effective dimension for the interaction term: $N_{eff} = 1$. And we allocate and initialize this array of structures by repeatedly calling the subroutine `Op_make`:

```
N_dim = Latt%N
N_fl = 1
N_eff = 1

do nf = 1, N_FL
do i  = 1, Latt%N
call Op_make(Op_V(i,nf),N_eff)
enddo
enddo
```

For each lattice site $i$, the projection matrices $\boldsymbol{P}_V^{(i)}$ are of dimension $1 \times N_{dim}$ and have one non-vanishing entry: $(P_V^{(i)})_{1j} = \delta_{ij}$. The effective matrices are scalars in this example: $\boldsymbol{O}_V^{(i)} = 1$.

| Name of variable in the code | Description |
|---|---|
| `Ndim` | Spacial dimension of the lattice (total number of sites) |
| `Latt%N` | Number of unit cells of the underlying Bravais lattice |
| `Op_T` | Array of structure variables that bundles all variables needed to define the hopping operator. |
| `Op_V` | Array of structure variables that bundles all variables needed to define the two-particle interaction operator. |
| `N_sun` | Number of fermion colors spin states of the $SU(N_{sun})$-symmetric fermions |
| `N_fl` | Number of fermion flavors |

Table 2: Common variables that are set in the Hamiltonian, operator and lattice modules of the code. !!! We have a missmatch in the labelling: $N_{col}$ = `N_sun` !!!

## 4.3 Definition of the square lattice

This is set in the subroutine `Ham_latt`. The square lattice is already implemented. In principle, one can specify other lattice geometries and use them by specifying the keyword `Lattice_type` in the parameter file.

## 4.4 Observables for the Hubbard model

To do next:

- dicuss the measurements: what observables exit and how do I add a new one?
- discuss the implementation of the lattice.
- discuss the Hubbard-Stratonovich decompositions (this is related to the coupling in the operator structure), discuss also the spin-symmetry-breaking HS-decomposition for the Hubbard model.

## 5 Tutorial: set up a model Hamiltonian

based on the (not yet existing) template `Hamiltonian_template.f90`.