Documentation for the General QMC code

Authors: Martin Bercx and Fakher F. Assaad

November 6, 2016

Contents

1 Definition of the model Hamiltonian

The class of solvable models includes Hamiltonians $\hat{\mathcal{H}}$ that have the following general form: $\hat{\mathcal{H}} = \hat{\mathcal{H}}_T + \hat{\mathcal{H}}_V + \hat{\mathcal{H}}_I + \hat{\mathcal{H}}_{0,I}$, where

$$\hat{\mathcal{H}}_T = \sum_{k=1}^{M_T} \sum_{s=1}^{N_{\mathrm{fl}}} \sum_{\sigma=1}^{N_{\mathrm{col}}} \sum_{x,y} \hat{c}^{\dagger}_{x\sigma s} T^{(ks)}_{xy} \hat{c}_{y\sigma s} \equiv \sum_{k=1}^{M_T} \hat{T}^{(k)} \tag{1}$$

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

$$\hat{\mathcal{H}}_I = \sum_{k=1}^{M_I} \hat{Z}_k \left\{ \sum_{s=1}^{N_{\mathrm{fl}}} \sum_{\sigma=1}^{N_{\mathrm{col}}} \left[\sum_{x,y} \hat{c}^{\dagger}_{x\sigma s} I^{(ks)}_{xy} \hat{c}_{y\sigma s} \right] \right\} \equiv \sum_{k=1}^{M_I} \hat{Z}_k \hat{I}^{(k)} . \tag{3}$$

The indices have the following meaning:

- The number of fermion *flavors* is set by N_{fl}. After the Hubbard Stratonovitch transformation, the action will be block diagonal in the flavor index.
- The number of fermion *colors* is set by N_{col}. The Hamiltonian is invariant under SU(N_{col}) rotations. Note that in the code $N_{\mathrm{col}} \equiv N_{sun}$.
- The indices x, y label lattice sites where $x, y = 1, \cdots, N_{\mathrm{dim}}$. N_{dim} is the total number of spacial vertices: $N_{\mathrm{dim}} = N_{unit\ cell} N_{orbital}$, 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 $\boldsymbol{T}^{(ks)}$, $\boldsymbol{V}^{(ks)}$ and $\boldsymbol{I}^{(ks)}$ are of dimension $N_{\mathrm{dim}} \times N_{\mathrm{dim}}$
- The number of interaction terms is labelled by M_V and M_I. $M_T > 1$ would allow for a checkerboard decomposition.
- $\hat{Z}_k$ is an Ising variable which couples to a general one-body term.
- $\mathcal{H}_{0,I}$ gives the dynamics of the Ising variable. This term has to be specified by the user and is only relevant when the Monte Carlo update probability is computed in the code (see Sec. **??**).

Note that the matrices $\boldsymbol{T}^{(ks)}$, $\boldsymbol{V}^{(ks)}$ and $\boldsymbol{I}^{(ks)}$ explicitly depend on the flavor index s but not on the color index σ. The color index σ only appears in the second quantized operators such that the Hamiltonian is manifestly SU(N_{col}) symmetric. We also require the matrices $\boldsymbol{T}^{(ks)}$, $\boldsymbol{V}^{(ks)}$ and $\boldsymbol{I}^{(ks)}$ to be hermitian.

1.1 Formulation of the QMC

The formulation of the Monte Carlo simulation is based on the following.

- We will discretize the imaginary time propagation: $\beta = \Delta\tau L_{\mathrm{Trotter}}$
- We will use the discrete Hubbard Stratonovitch transformation:

$$e^{\Delta\tau\lambda\hat{A}^2} = \sum_{l=\pm1,\pm2} \gamma(l) e^{\sqrt{\Delta\tau\lambda}\eta(l)\hat{A}} + \mathcal{O}(\Delta\tau^4) , \tag{4}$$

where the fields η and γ take the values:

$$\gamma(\pm1) = 1 + \sqrt{6}/3, \quad \eta(\pm1) = \pm\sqrt{2\left(3-\sqrt{6}\right)} ,$$
$$\gamma(\pm2) = 1 - \sqrt{6}/3, \quad \eta(\pm2) = \pm\sqrt{2\left(3+\sqrt{6}\right)} . \tag{5}$$

- We will work in a basis where $\hat{Z}_k$ is diagonal: $\hat{Z}_k|s_j\rangle = s_k\delta_{kj}|s_k\rangle$, where $s_k = \pm 1$.
- From the above it follows that the Monte Carlo configuration space is given by:

$$C = \{s_{i,\tau}, l_{j,\tau} \text{ with } i = 1\cdots M_I,\ j = 1\cdots M_V,\ \tau = 1\cdots L_{\text{Trotter}}\} \tag{6}$$

  Here, $s_{i,\tau} = \pm 1$ and $l_{j,\tau} = \pm 2, \pm 1$.

With the above, the partition function of the model can be written as follows.

$$\begin{aligned}
Z &= \operatorname{Tr} e^{-\beta\hat{\mathcal{H}}} \\
&= \operatorname{Tr}\left[e^{-\Delta\tau\hat{\mathcal{H}}_{0,I}} \prod_{k=1}^{M_T} e^{-\Delta\tau\hat{T}^{(k)}} \prod_{k=1}^{M_V} e^{\Delta\tau U_k \left(\hat{V}^{(k)}\right)^2} \prod_{k=1}^{M_I} e^{-\Delta\tau\hat{\sigma}_k\hat{I}^{(k)}}\right]^{L_{\text{Trotter}}} \\
&= \sum_C \left(\prod_{j=1}^{M_V}\prod_{\tau=1}^{L_{\text{Trotter}}} \gamma_{j,\tau}\right) e^{-S_{0,I}(\{s_{i,\tau}\})} \times \\
&\quad \operatorname{Tr}_{\text{F}} \prod_{\tau=1}^{L_{\text{Trotter}}}\left[\prod_{k=1}^{M_T} e^{-\Delta\tau\hat{T}^{(k)}} \prod_{k=1}^{M_V} e^{\sqrt{\Delta\tau U_k}\eta_{k,\tau}\hat{V}^{(k)}} \prod_{k=1}^{M_I} e^{-\Delta\tau s_{k,\tau}\hat{I}^{(k)}}\right]
\end{aligned} \tag{7}$$

In the above, the trace Tr runs over the Ising spins as well as over the fermionic degrees of freedom, and $\operatorname{Tr}_{\text{F}}$ only over the fermionc Fock space. $S_{0,I}\left(\{s_{i,\tau}\}\right)$ is the action corresponding to the Ising Hamiltonian, and is only dependent on the Ising spins so that it can be pulled out of the fermionic trace. At this point, and since for a given configuration C we are dealing with a free propagation, we can integrate out the fermions to obtain a determinant:

$$\begin{aligned}
&\operatorname{Tr}_{\text{F}} \prod_{\tau=1}^{L_{\text{Trotter}}}\left[\prod_{k=1}^{M_T} e^{-\Delta\tau\hat{T}^{(k)}} \prod_{k=1}^{M_V} e^{\sqrt{\Delta\tau U_k}\eta_{k,\tau}\hat{V}^{(k)}} \prod_{k=1}^{M_I} e^{-\Delta\tau s_{k,\tau}\hat{I}^{(k)}}\right] = \\
&\prod_{s=1}^{N_{\text{fl}}}\left[e^{-\sum_{k=1}^{M_V}\sum_{\tau=1}^{L_{\text{Trotter}}}\sqrt{\Delta\tau U_k}\alpha_{k,s}\eta_{k,\tau}}\right]^{N_{\text{col}}} \times \\
&\prod_{s=1}^{N_{\text{fl}}}\left[\det\left(1 + \prod_{\tau=1}^{L_{\text{Trotter}}}\prod_{k=1}^{M_T} e^{-\Delta\tau\mathbf{T}^{(\mathbf{ks})}} \prod_{k=1}^{M_V} e^{\sqrt{\Delta\tau U_k}\eta_{k,\tau}\mathbf{V}^{(\mathbf{ks})}} \prod_{k=1}^{M_I} e^{-\Delta\tau s_{k,\tau}\mathbf{I}^{(\mathbf{ks})}}\right)\right]^{N_{\text{col}}} .
\end{aligned} \tag{8}$$

This all in all, the partition function is given by:

$$\begin{aligned}
Z &= \operatorname{Tr} e^{-\beta\hat{\mathcal{H}}} \\
&= \sum_C e^{-S_{0,I}(\{s_{i,\tau}\})}\left[\prod_{k=1}^{M_V}\prod_{\tau=1}^{L_{\text{Trotter}}}\gamma_{k,\tau}\right] e^{-N_{\text{col}}\sum_{s=1}^{N_{\text{fl}}}\sum_{k=1}^{M_V}\sum_{\tau=1}^{L_{\text{Trotter}}}\sqrt{\Delta\tau U_k}\alpha_{k,s}\eta_{k,\tau}} \times \\
&\quad \prod_{s=1}^{N_{\text{fl}}}\left[\det\left(1 + \prod_{\tau=1}^{L_{\text{Trotter}}}\prod_{k=1}^{M_T} e^{-\Delta\tau\mathbf{T}^{(\mathbf{ks})}} \prod_{k=1}^{M_V} e^{\sqrt{\Delta\tau U_k}\eta_{k,\tau}\mathbf{V}^{(\mathbf{ks})}} \prod_{k=1}^{M_I} e^{-\Delta\tau s_{k,\tau}\mathbf{I}^{(\mathbf{ks})}}\right)\right]^{N_{\text{col}}} .
\end{aligned} \tag{9}$$

In the above, one notices that the weight factorizes in the flavor index. The color index raises the determinant to the power N_{col}. This corresponds to an explicit $SU(N_{\text{col}})$ symmetry for each configuration. This symmetry is manifest in the fact that the single particle Green functions, again for a given configuration C are color independent.

1.2 The `Operator` type and specification of the model.

The fundamental data structure in the code is the derived data type `Operator` which one uses to define the Hamiltonian.

Name of variable in the code	Description
`Op_V%N`	effective dimension N
`Op_V%O`	matrix $\mathbf{O}$ of dimension $N \times N$
`Op_V%P`	projection matrix $\mathbf{P}$ encoded as a vector of dimension N.
`Op_V%g`	coupling strength g
`Op_V%alpha`	constant α
`Op_V%type`	integer parameter to set the type of HS transformation (1 = Ising, 2 = Discrete HS, for perfect square)
`Op_V%U`	matrix containing the eigenvectors of $\mathbf{O}$
`Op_V%E`	eigenvalues of $\mathbf{O}$
`Op_V%N_non_zero`	number of non-vanishing eigenvalues of $\mathbf{O}$

Table 1: Components of the `Operator` type, using the example of the variable `Op_V` which describes the interaction.

In general, the matrices $\mathbf{T}^{(ks)}$, $\mathbf{V}^{(ks)}$ and $\mathbf{I}^{(ks)}$are sparse Hermitian matrices. Consider the matrix $\boldsymbol{M}$ of dimension $N_{\text{dim}} \times N_{\text{dim}}$. Let us denote with $\{z_1, \cdots, z_N\}$ a subset of N indices, for which

$$M_{x,y} = \begin{cases} M_{x,y} & \text{if } x, y \in \{z_1, \cdots z_N\} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

We define the $N \times N_{\text{dim}}$ matrices $\mathbf{P}$ as

$$P_{i,x} = \delta_{z_i,x} \; , \tag{11}$$

where $i \in [1, \cdots, N]$ and $x \in [1, \cdots, N_{\text{dim}}]$. The matrix $\boldsymbol{P}$ picks out the non-vanishing entries of $\boldsymbol{M}$, which are contained in the rank-N matrix $\boldsymbol{O}$. Thereby:

$$\boldsymbol{M} = \boldsymbol{P}^T \boldsymbol{O} \boldsymbol{P} \; , \tag{12}$$

such that:

$$M_{x,y} = \sum_{i,j}^{N} P_{i,x} O_{i,j} P_{j,y} = \sum_{i,j}^{N} \delta_{z_i,x} O_{ij} \delta_{z_j,y} \; . \tag{13}$$

Since the $\boldsymbol{P}$ matrices have only one non-vanishing entry per column, they can be stored as a vector $\vec{P}$:

$$P_i = z_i. \tag{14}$$

There are many useful identities which emerge from this structure. For example:

$$e^{\boldsymbol{M}} = e^{\boldsymbol{P}^T \boldsymbol{O} \boldsymbol{P}} = \sum_{n=0}^{\infty} \frac{\left(\boldsymbol{P}^T \boldsymbol{O} \boldsymbol{P}\right)^n}{n!} = \boldsymbol{P}^T e^{\boldsymbol{O}} \boldsymbol{P} \tag{15}$$

since

$$\boldsymbol{P} \boldsymbol{P}^T = 1_{N \times N}. \tag{16}$$

In the code, we define a structure called `Operator` to capture the above. This type `Operator` bundles several components that are needed to define and use an operator matrix in the program. In general, we will not only have one structure variable `Operator`, but a whole array of these structures, which defines the very Hamiltonian. The implementation is as follows:

- Interaction term: If the interaction is of perfect-square type, we set $\boldsymbol{M} = \boldsymbol{V}^{(k,s)}$ and define the corresponding structure variables `Op_V` using the array `Op_V(M_V,N_fl)`. Precisely, a single variable `Op_V` describes the operator matrix:

$$\left[\left(\sum_{x,y} \hat{c}_x^{\dagger} V_{x,y} \hat{c}_y \right) - \alpha \right] . \tag{17}$$

  Its components are listed in table 1. For the perfect-square interaction, $\alpha = \alpha_{k,s}$ and $g = \sqrt{\Delta\tau U_k}$. The discrete Hubbard-Stratonovich decomposition is selected by setting the type variable to `Op_V%type` $= 2$.

- Hopping term: In this case $\boldsymbol{M} = \boldsymbol{T}^{(k,s)}$. The corresponding array of structure variables `Op_T` is `Op_T(`M_T`,`N_{fl}`)` . We have $g = -\Delta\tau$, $\alpha = 0$, and the type variable `Op_T%type` is irrelevant.

- Ising interaction term: In this case, $\boldsymbol{M} = \boldsymbol{I}^{(k,s)}$ and we define the array `Op_V(M_I,N_fl)`. The Ising interaction is specified by setting the type variable `Op_V%type=1`, $\alpha = 0$ and $g = -\Delta\tau$.

- In case of a full interaction (perfect-square term and Ising term), we define the array `Op_V(`M_V`+`M_I`,`N_{fl}`)`) and set the individual variables according to the above.

1.3 The Lattice

We have a lattice module which generate a two dimensional Bravais lattice. The user has to specify unit vectors $\vec{a}_1$ and $\vec{a}_2$ as well as the size of the lattice. The size is characterized by two vectors $\vec{L}_1$ and $\vec{L}_2$ and the lattice is placed on a torus:

$$\hat{c}_{\vec{i}+\vec{L}_1} = \hat{c}_{\vec{i}+\vec{L}_2} = \hat{c}_{\vec{i}} \tag{18}$$

The call to `Call Make_Lattice( L1, L2, a1, a2, Latt )` will generate the lattice `Latt` of type `Lattice`. Note that the structure of the unit cell has to be provided by the user.

We need to add a table for the lattice type.

1.4 The Observables

We have three types of observables.

- Scalar observables such as the energy

- Equal time correlation functions. Let $\hat{O}_{\vec{i},\alpha}$ be a local observable, with $\vec{i}$ labelling the unit cell and α labelling the orbital or bone emanating from the unit cell. The program will compute:

$$S_{\alpha,\beta}(\vec{k}) = \frac{1}{N_{unit\ cells}} \sum_{\vec{i},\vec{j}} e^{i\vec{k}\cdot(\vec{i}-\vec{j})} \left(\langle \hat{O}_{\vec{i},\alpha} \hat{O}_{\vec{j},\alpha} \rangle - \langle \hat{O}_{\vec{i},\beta} \rangle \langle \hat{O}_{\vec{i},\beta} \rangle \right) \tag{19}$$

- Time displaced correlation functions. This has a very similar structure than above but now with an additional time index.

$$S_{\alpha,\beta}(\vec{k},\tau) = \frac{1}{N_{unit\ cells}} \sum_{\vec{i},\vec{j}} e^{i\vec{k}\cdot(\vec{i}-\vec{j})} \left(\langle \hat{O}_{\vec{i},\alpha}(\tau) \hat{O}_{\vec{j},\alpha} \rangle - \langle \hat{O}_{\vec{i},\beta} \rangle \langle \hat{O}_{\vec{i},\beta} \rangle \right) \tag{20}$$

We have to add some more details.

2 Input and output files, compilation etc.

To Do

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

To implement a Hamiltonian, the user has to provide a module which specifies the lattice, the model, as well as the observables he/she wishes to compute. In this section, we describe the module `Hamiltonian_Hub.f90` which is an implementation of the Hubbard model on the square lattice. The Hamiltonian reads

$$\mathcal{H} = \sum_{\sigma=1}^{2} \sum_{x,y=1}^{N_{unit\ cells}} c^{\dagger}_{x\sigma} T_{x,y} c_{y\sigma} + \frac{U}{2} \sum_{x} \left[\sum_{\sigma=1}^{2} \left(c^{\dagger}_{x\sigma} c_{x\sigma} - 1/2 \right) \right]^2 . \tag{21}$$

We can make contact with the general form of the Hamiltonian by setting: $N_{fl} = 1$, $N_{col} \equiv N_{SUn} = 2$, $M_T = 1$, $T^{(ks)}_{xy} = T_{x,y}$, $M_V = N_{Unit\ cells}$, $U_k = -\frac{U}{2}$, $V^{(ks)}_{xy} = \delta_{x,y}\delta_{x,k}$, $\alpha_{ks} = \frac{1}{2}$ and $M_I = 0$.

3.1 Setting the Hamiltonian. `Ham_set`

The main program will call the subroutine `Ham_set` in the module `Hamiltonian_Hub.f90`. This subroutine defines the public variables

```
Type (Operator), dimension(:,:), allocatable  :: Op_V
Type (Operator), dimension(:,:), allocatable  :: Op_T
Integer, allocatable :: nsigma(:,:)
Integer              :: Ndim,  N_FL,  N_SUN,  Ltrot
```

which specify the model. This routine will first read the parameter file, then set the lattice, `Call Ham_latt`, set the hopping `Call Ham_hop` and set the interaction `call Ham_V`. The parameters are read in from the file `parameters`:

```
===============================================================================
!  Variables for the Hubb program
!-------------------------------------------------------------------------------
&VAR_lattice
L1=4                     ! Length in direction a_1
L2=4                     ! Length in direction a_2
Lattice_type = "Square"  ! a_1 = (1,0) and a_2=(0,1)
Model = "Hubbard_SU2"    ! Sets  Nf = 1, N_sun = 2
/
&VAR_Hubbard             ! Variables for the Hubbard model
ham_T   =1.D0
ham_chem=0.D0
ham_U   =4.D0
Beta    =5.D0
dtau    =0.1D0           ! Thereby Ltrot=Beta/dtau
/
&VAR_QMC      ! Variables for the QMC run
Nwrap   = 10  ! Stabilization. Green functions will be computed from scratch
              ! after each time interval  Nwrap*Dtau
NSweep  = 500 ! Number of sweeps
NBin    = 2   ! Number of bins
Ltau    = 1   ! 1 for calculation of time desplaced. 0 otherwise
LOBS_ST = 1   ! Start measurments at time slice LOBS_ST
LOBS_EN =50   ! End   measurments at time slice LOBS_EN
```

```
CPU_MAX= 0.1  ! Code will stop after CPU_MAX hours.
              ! If not specified, code will stop after Nbin bins.
/
```

Here we have three name lists relevant for defining the lattice, model parameters as well as the Monte Carlo run. Thereby, `Ltrot=Beta/dtau`.

3.1.1 The lattice. `Call Ham_latt`

The choice `Lattice_type = "Square"` sets $\vec{a}_1 = (1,0)$ and $\vec{a}_2 = (0,1)$ and for an $L_1 \times L_2$ lattice $\vec{L}_1 = L_1\vec{a}_1$ and $\vec{L}_2 = L_2\vec{a}_2$. The call to `Call Make_Lattice( L1, L2, a1, a2, Latt)` will generate the lattice `Latt` of type `Lattice` such that $N_{dim} = N_{unit\ cell} \equiv Latt\%N$.

3.1.2 Hopping term. `Call Ham_hop`

The hopping matrix is implemented as follows. We allocate an array of dimension 1×1 of type operator called `Op_T` and set the dimension for the hopping matrix to $N = N_{dim}$. One allocates and initializes this type by a single call to the subroutine `Op_make`:

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

Since the hopping does not break down into small blocks $\boldsymbol{P} = \mathbb{1}$ and

```
Do i= 1,Latt%N
  Op_T(1,1)%P(i) = i
Enddo
```

We set the hopping matrix with

```
DO I = 1, Latt%N
   Ix = Latt%nnlist(I,1,0)
   Iy = Latt%nnlist(I,0,1)
   Op_T(1,1)%O(I  ,Ix) = cmplx(-Ham_T,   0.d0,kind(0.D0))
   Op_T(1,1)%O(Ix,I  ) = cmplx(-Ham_T,   0.d0,kind(0.D0))
   Op_T(1,1)%O(I  ,Iy) = cmplx(-Ham_T,   0.d0,kind(0.D0))
   Op_T(1,1)%O(Iy, I ) = cmplx(-Ham_T,   0.d0,kind(0.D0))
   Op_T(1,1)%O(I  ,I ) = cmplx(-Ham_chem,0.d0,kind(0.D0))
ENDDO
```

Here, the integer function `j= Latt%nnlist(I,n,m)` is defined in the lattice module and returns the index of the lattice site $\vec{I} + n\vec{a}_1 + m\vec{a}_2$. Note that periodic boundary conditions are already taken into account. The hopping parameter, `Ham_T` as well as the chemical potential `Ham_chem` are read from the parameter file.

Note that although a checkerboard decomposition is not used here, it can be implemented by considering a larger number of sparse Hopping matrices.

3.1.3 Interaction term. `Call Ham_V`

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

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

For each lattice site i, the matrices $\boldsymbol{P}$ are of dimension $1 \times N_{dim}$ and have only one non-vanishing entry. Thereby we can set:

```
Do i = 1,Latt%N
   Op_V(i,1)%P(1)    = i
   Op_V(i,1)%O(1,1)  = cmplx(1.d0,0.d0, kind(0.D0))
   Op_V(i,1)%g       = sqrt(cmplx(-dtau*ham_U/(dble(N_SUN)),0.D0,kind(0.D0)))
   Op_V(i,1)%alpha   = cmplx(-0.5d0,0.d0, kind(0.D0))
   Op_V(i,1)%type    = 2
Enddo
```

so as to completely define the interaction term.

3.2 Observables

At this point, all the information for the simulation to start has been provided. The code will sequentially go through the operator list `Op_V` and update the fields. Between time slices `LOBS_ST` and `LOBS_EN` the main program will call the routine `Obser(GR,Phase,Ntau)` which is also provided by the user. For each configuration of the fields Wicks theorem holds so that it suffices to know the single particle Green function so as to compute any observable. The main program provides the equal time Green function: `GR(Ndim,Ndim,N_FL)`, the phase `PHASE` and the time slice on which the measurement is being carried out `Ntau`. The Green function is defined as:

$$GR(x,y,\sigma) = \langle c_{x,\sigma} c^{\dagger}_{y,\sigma} \rangle \tag{22}$$

Space for observables in allocated in the subroutine `Call Alloc_obs`. At the beginning of each bin, the observables are set to zero `Call Init_obs` and at the end of each bin the observables are written out on disc `Call Pr_obs`.

4 Walkthrough: the $SU(2)$-Hubbard model on a square lattice coupled to a transverse Ising field

The model we consider here is very similar to the above, but has an additional coupling to a transverse field.

$$\mathcal{H} = \sum_{\sigma=1}^{2} \sum_{x,y} c^{\dagger}_{x\sigma} T_{x,y} c_{y\sigma} + \frac{U}{2} \sum_{x} \left[\sum_{\sigma=1}^{2} \left(c^{\dagger}_{x\sigma} c_{x\sigma} - 1/2 \right) \right]^2 + \xi \sum_{\sigma,\langle x,y \rangle} \hat{Z}_{\langle x,y \rangle} \left(c^{\dagger}_{x\sigma} c_{y\sigma} + h.c. \right) + h \sum_{\langle x,y \rangle} \hat{X}_{\langle x,y \rangle} \tag{23}$$

We can make contact with the general form of the Hamiltonian by setting: $N_{fl} = 1$, $N_{col} \equiv N_{SUn} = 2$, $M_T = 1$, $T^{(ks)}_{xy} = T_{x,y}$, $M_V = N_{Unit\ cells} \equiv N_{dim}$, $U_k = -\frac{U}{2}$, $V^{(ks)}_{xy} = \delta_{x,y}\delta_{x,k}$, $\alpha_{ks} = \frac{1}{2}$ and $M_I = 2N_{Unit\ cells}$. The modifications required to generalize the Hubbard model code to the above model are two-fold. Firstly, one has to specify the function `Real (Kind=8) function S0(n,nt)` and secondly modify the interaction `Call Ham_V`.

4.1 Interaction term. `Call Ham_V`

The dimension of `Op_V` is now $(M_V + M_I) \times N_{fl} = (3 * N_{dim}) \times 1$. We set the effective dimension for the Hubbard term to $N = 1$ and to $N = 2$ for the Ising term. The allocation of this array of operators reads:

```
do i  = 1,Ndim
```

```
  call Op_make(Op_V(i,1),1)
enddo
do i  =  Ndim+1, 3*Ndim
  call Op_make(Op_V(i,1),2)
enddo
```

As for the Hubbard case, the first `Ndim` operators read:

```
Do i = 1,Ndim
    Op_V(i,1)%P(1)   = i
    Op_V(i,1)%O(1,1) = cmplx(1.d0  ,0.d0, kind(0.D0))
    Op_V(i,1)%g      = sqrt(cmplx(-dtau*ham_U/(DBLE(N_SUN)), 0.D0, kind(0.D0)))
    Op_V(i,1)%alpha  = cmplx(-0.5d0,0.d0, kind(0.D0))
    Op_V(i,1)%type   = 2
Enddo
```

The next `2*Ndim` operators run through the 2N bonds of the square lattice and are given by:

```
Do nc = 1,N_coord   ! Coordination number = 2
   Do i = 1,Ndim
      j = i + nc*Ndim
      Op_V(j,1)%P(1)    =  i
      If (nc == 1) Op_V(j,1)%P(2)   = Latt%nnlist(i,1,0)
      If (nc == 2) Op_V(j,1)%P(2)   = Latt%nnlist(i,0,1)
      Op_V(j,1)%O(1,2) = cmplx(1.d0  ,0.d0, kind(0.D0))
      Op_V(j,1)%O(2,1) = cmplx(1.d0  ,0.d0, kind(0.D0))
      Op_V(j,1)%g      = cmplx(-dtau*ham_xi, 0.D0, kind(0.D0)))
      Op_V(j,1)%alpha  = cmplx(0d0,0.d0, kind(0.D0))
      Op_V(j,1)%type   = 1
    Enddo
Enddo
```

Here, `ham_xi` defines the coupling strength between the Ising and fermion degree of freedom.

4.2 The function `Real (Kind=8) function S0(n,nt)`

As mentioned above, a configuration is given by

$$C = \{s_{i,\tau}, l_{j,\tau} \text{ with } i = 1 \cdots M_I, j = 1 \cdots M_V, \tau = 1, L_{Trotter}\} \tag{24}$$

and is stored in the integer array `nsigma(M_V + M_I, Ltrot)`. With the above ordering of Hubbard and Ising interaction terms, and a for a given imaginary time, the first Ndim fields corresponds to the Hubbard interaction and the next 2*Ndim ones to the Ising interaction. The first argument of the function `S0`, n, corresponds to the index of the operator string `Op_V(n,1)`. If `Op_V(n,1)%type = 2`, `S0(n,nt)` returns 1. If `Op_V(n,1)%type = 1` then function `S0` returns

$$\frac{e^{-S_{0,I}\left(s_{1,\tau},\cdots,-s_{m,\tau},\cdots s_{M_I,\tau}\right)}}{e^{-S_{0,I}\left(s_{1,\tau},\cdots,s_{m,\tau},\cdots s_{M_I,\tau}\right)}} \tag{25}$$

That is, `S0(n,nt)` returns the ratio of the new to old weight of the Ising Hamiltonian upon flipping a single Ising spin $s_{m,\tau}$. Note that in this specific case `m = n - Ndim`