Documentation for the General QMC code

Author: Martin Bercx

November 6, 2016

Contents

1 Definition of the model Hamiltonian

Notation: Hats for second quantized operators, bold for matrices.
Structure:
1) We first want to define the model.
2) implementation of the QMC.
3) Data structure
4) Practical implementation and some simple test cases.

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

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

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

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

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 $N_{col} \equiv N_{SUn}$.
- 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 $\boldsymbol{T}^{(ks)}$, $\boldsymbol{V}^{(ks)}$ and $\boldsymbol{I}^{(ks)}$ are of dimension $N_{dim} \times N_{dim}$
- 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.

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 only 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 work in a basis where $\hat{Z}_k$ is diagonal.
- We will discretize the imaginary time propagation: $\beta = \Delta\tau L_{\text{Trotter}}$

- We will the discrete Hubbard Stratonovitch transformation:

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

  where the fields η and γ take the values:

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

- From the above it follows that the Monte Carlo configuration space is given by:

$$C = \{s_{i,\tau}, l_{j,\tau} \text{ with } i = 1\cdots M_I, j = 1\cdots M_V, \tau = 1, L_{Trotter}\} \tag{5}$$

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

$$Z = \text{Tr}e^{-\beta\mathcal{H}} = \text{Tr}\left[e^{-\Delta\tau\mathcal{H}_{0,I}} \prod_{k=1}^{M_T} e^{-\Delta\tau\hat{T}^{(k)}} \prod_{k=1}^{M_V} e^{\Delta\tau U_k\left(\hat{V}^{(k)}\right)^2} \prod_{k=1}^{M_I} e^{-\Delta\tau\hat{\sigma}_k\hat{I}^{(k)}}\right]^{L_{\text{Trotter}}}$$

$$\sum_C \left(\prod_{j=1}^{M_V}\prod_{\tau=1}^{L_{Trotter}} \gamma_{j,\tau}\right) e^{-S_{0,I}(\{s_{i,\tau}\})}\text{Tr}_F \prod_{\tau=1}^{L_{Trotter}} \left[\prod_{k=1}^{M_T} e^{-\Delta\tau\hat{T}^{(k)}} \prod_{k=1}^{M_V} e^{\sqrt{\Delta\tau U_k}\eta_{k,\tau}\hat{V}^{(k)}} \prod_{k=1}^{M_I} e^{-\Delta\tau s_{k,\tau}\hat{I}^{(k)}}\right]$$

In the above, Tr runs over the Ising spins as well as over the fermionic degrees of freedom, and Tr_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:

$$\text{Tr}_F \prod_{\tau=1}^{L_{Trotter}} \left[\prod_{k=1}^{M_T} e^{-\Delta\tau\hat{T}^{(k)}} \prod_{k=1}^{M_V} e^{\sqrt{\Delta\tau U_k}\eta_{k,\tau}\hat{V}^{(k)}} \prod_{k=1}^{M_I} e^{-\Delta\tau s_{k,\tau}\hat{I}^{(k)}}\right] = \tag{6}$$
$$\prod_{s=1}^{N_{fl}} \left[e^{-\sum_{k=1}^{M_V}\sum_{\tau=1}^{L_{Trotter}} \sqrt{\Delta\tau U_k}\alpha_{k,s}\eta_{k,\tau}} \det\left(1 + \prod_{\tau=1}^{L_{Trotter}} \prod_{k=1}^{M_T} e^{-\Delta\tau\mathbf{T}^{(\mathbf{ks})}} \prod_{k=1}^{M_V} e^{\sqrt{\Delta\tau U_k}\eta_{k,\tau}\mathbf{V}^{(\mathbf{ks})}} \prod_{k=1}^{M_I} e^{-\Delta\tau s_{k,\tau}\mathbf{I}^{(\mathbf{ks})}}\right)\right]^{N_{col}}$$

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

$$Z = \text{Tr}e^{-\beta\mathcal{H}} = \sum_C e^{-S_{0,I}(\{s_{i,\tau}\})} \left[\prod_{k=1}^{M_V}\prod_{\tau=1}^{L_{Trotter}} \gamma_{k,\tau}\right] e^{-N_{col}\sum_{s=1}^{N_{fl}}\sum_{k=1}^{M_V}\sum_{\tau=1}^{L_{Trotter}} \sqrt{\Delta\tau U_k}\alpha_{k,s}\eta_{k,\tau}}$$
$$\left[\prod_{s=1}^{N_{fl}} \det\left(1 + \prod_{\tau=1}^{L_{Trotter}} \prod_{k=1}^{M_T} e^{-\Delta\tau\mathbf{T}^{(\mathbf{ks})}} \prod_{k=1}^{M_V} e^{\sqrt{\Delta\tau U_k}\eta_{k,\tau}\mathbf{V}^{(\mathbf{ks})}} \prod_{k=1}^{M_I} e^{-\Delta\tau s_{k,\tau}\mathbf{I}^{(\mathbf{ks})}}\right)\right]^{N_{col}} \tag{7}$$

In the above, one notices that the weight factorizes in the flavor index, and that the underlying color symmetry shows up in the fact that the the weight does not depend upon the flavor index.

The fundamental data structure in the code is the Operator data structure which one uses to define all quantities in the Hamiltonian. This data structure will be defined below.

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

In general, the matrices $\mathbf{T}^{(ks)}$ 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 a sparse

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 $\mathbf{P}_V$ encoded as a vector of dimension N.
`Op_V%g`	coupling strength
`Op_V%alpha`	constant
`Op_V%type`	integer parameter to set the type of HS transformation (1 = Ising, 2 = Discrete HS, fo
`Op_V%U`	matrix containing the eigenvectors of $\mathbf{O}$
`Op_V%E`	eigenvalues of $\mathbf{O}_V$
`Op_V%N_non_zero`	number of non-vanishing eigenvalues of $\mathbf{O}_V$

Table 1: Components of the `Operator` structure variable `Op_V` One will have to specify N, O, P, g, α and the type. The other variables will be automatically generated in the routine `Op_Set`.

matrix representation of $\mathbf{V}$. $\mathbf{V}$ has dimension $N_{dim} \times N_{dim}$ and we denote a subset of N indices, $\{z_1, \cdots, z_N\}$ for which

$$V_{x,y} = \begin{cases} V_{x,y} & \text{if } x, y \in \{z_1, \cdots z_N\} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

We define the projection matrices $\mathbf{P}$ of dimension $N \times N_{dim}$:

$$\boldsymbol{P}_{i,x} = \delta_{z_i,x} \ , \tag{9}$$

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

$$\boldsymbol{V} = \boldsymbol{P}^T \boldsymbol{O} \boldsymbol{P} \ , \tag{10}$$

such that:

$$\boldsymbol{V}_{x,y} = \sum_{i,j}^{N} \boldsymbol{P}_{i,x} \boldsymbol{O}_{i,j} \boldsymbol{P}_{j,y} = \sum_{i,j}^{N} \delta_{z_i,x} \boldsymbol{O}_{ij} \delta_{z_j,y} \ . \tag{11}$$

Thereby the full information of $\boldsymbol{P}$ can be stored as a vector

$$\vec{P}_i = z_i \tag{12}$$

which reflects the fact that the $\boldsymbol{P}$ matrices have only one non-vanishing entry per column. There are many useful identities which emerge from this structure. For example:

$$e^{\boldsymbol{V}} = e^{\boldsymbol{P}^T \boldsymbol{O} \boldsymbol{P}} = \sum_{n=0}^{\infty} \frac{\left(\boldsymbol{P}^T \boldsymbol{O} \boldsymbol{P}\right)^n}{n!} = \boldsymbol{P}^T e^{\boldsymbol{O}} \boldsymbol{P} \tag{13}$$

since

$$\boldsymbol{P}\boldsymbol{P}^T = 1_{N \times N}. \tag{14}$$

In the code implementation, we define a structure called `Operator` to capture the above. 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 of the operator $\boldsymbol{V}$ are listed in the table 1. `Op_V` describes the operator:

$$\left[\left(\sum_{x,y} \hat{c}^\dagger_x \boldsymbol{V}_{x,y} \hat{c}_y \right) - \alpha \right] \tag{15}$$

where $\boldsymbol{V} = \boldsymbol{P}^T\boldsymbol{O}\boldsymbol{P}$. In general, we will not only have one structure variable `Operator`, but a whole array of these structures, which defines the very Hamiltonian.

In the code, there is one array of operators which defines the hopping $\boldsymbol{T}^{(k,s)}$: `Op_T(`M_T`,`N_{fl}`)` . In this case $\boldsymbol{V}$ corresponds to $\boldsymbol{T}^{(k,s)}$ and $g = -\Delta\tau$, and $\alpha = 0$. The type variable is irrelevant.

There is another array which defines the full interaction, Ising as well as perfect square terms. For this we define the array `Op_V(`M_V`+`M_I`,`N_{fl}`)`). In this context the variable `Op_V%type` specifies the interaction: Ising or a perfect square. If the interaction is of Ising type, then $\boldsymbol{V} = \boldsymbol{I}^{(k,s)}$, $\alpha = 0$ and $g = -\Delta\tau$. If the interaction is a perfect square type, then $\boldsymbol{V} = \boldsymbol{V}^{(k,s)}$, $\alpha = \alpha_{k,s}$ and $g = \sqrt{\Delta\tau U_k}$.

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:

$$\hat{c}_{\vec{i}+\vec{L}_1} = \hat{c}_{\vec{i}+\vec{L}_2} = \hat{c}_{\vec{i}} \tag{16}$$

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:

$$S_{\alpha,\beta}(\vec{k}) = \frac{1}{N_{unit\ cells}} \sum_{\vec{i},\vec{j}} e^{i\vec{k}\cdot(\vec{i}-\vec{j})} \left(\langle \hat{O}_{\vec{i},\alpha} \hat{O}_{\vec{j},\alpha} \rangle - \langle \hat{O}_{\vec{i},\beta} \rangle \langle \hat{O}_{\vec{i},\beta} \rangle \right) \tag{17}$$

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

$$S_{\alpha,\beta}(\vec{k},\tau) = \frac{1}{N_{unit\ cells}} \sum_{\vec{i},\vec{j}} e^{i\vec{k}\cdot(\vec{i}-\vec{j})} \left(\langle \hat{O}_{\vec{i},\alpha}(\tau) \hat{O}_{\vec{j},\alpha} \rangle - \langle \hat{O}_{\vec{i},\beta} \rangle \langle \hat{O}_{\vec{i},\beta} \rangle \right) \tag{18}$$

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

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

$$\mathcal{H} = \sum_{\sigma=1}^{2} \sum_{x,y} c^{\dagger}_{x\sigma} T_{x,y} c_{y\sigma} + \frac{U}{2} \sum_{x} \left[\sum_{\sigma=1}^{2} \left(c^{\dagger}_{x\sigma} c_{x\sigma} - 1/2 \right) \right]^2 . \tag{19}$$

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`. This subroutine sets a number of 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 define 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"  ! Nf = 1, N_sun = 2
/
&VAR_Hubbard
ham_T=1.0D0
ham_chem=0.0D0
ham_U= 4.0
Beta=5.0
dtau= 0.1
/
&VAR_QMC
Nwrap= 10
NSweep=  500
NBin= 2
Ltau= 1
LOBS_ST= 1
LOBS_EN=50
CPU_MAX= 0.1
/
```

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`. It therefore contains only a single `Operator` structure. We set the effective dimension for the hopping term: $N = N_{dim}$ and 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 $\boldsymbol{P} = \mathbb{1}$ such that

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

To will 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 yet used for the Hubbard model, in principle it can be implemented.

3.1.3 Interaction term. `Call Ham_V`

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, $N = 1$, and we allocate and initialize this array of structures 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 projection 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 run is provided. Between time slices between `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 completely determine 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:

$$GR(x,y,\sigma) = \langle c_{x,\sigma} c^{\dagger}_{y,\sigma} \rangle \tag{20}$$

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.

$$\mathcal{H} = \sum_{\sigma=1}^{2} \sum_{x,y} c^{\dagger}_{x\sigma} T_{x,y} c_{y\sigma} + \frac{U}{2} \sum_{x} \left[\sum_{\sigma=1}^{2} \left(c^{\dagger}_{x\sigma} c_{x\sigma} - 1/2 \right) \right]^2 + \xi \sum_{\sigma,\langle x,y \rangle} \hat{Z}_{\langle x,y \rangle} \left(c^{\dagger}_{x\sigma} c_{y\sigma} + h.c. \right) + h \sum_{\langle x,y \rangle} \hat{X}_{\langle x,y \rangle} \tag{21}$$

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

$$C = \{s_{i,\tau}, l_{j,\tau} \text{ with } i = 1 \cdots M_I, j = 1 \cdots M_V, \tau = 1, L_{Trotter}\} \tag{22}$$

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

$$\frac{e^{-S_{0,I}\left(s_{1,\tau},\cdots,-s_{m,\tau},\cdots s_{M_I,\tau}\right)}}{e^{-S_{0,I}\left(s_{1,\tau},\cdots,s_{m,\tau},\cdots s_{M_I,tau}\right)}} \tag{23}$$

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