Manual for the flexible DM-NRG code
Version 1.0.0

Team members:

Ö. Legeza
C. P. Moca
A. I. Tóth
I. Weymann
G. Zaránd

Contents

Chapter 1

Introduction

Quantum impurity models describe interactions between some local degrees of freedom (e.g. a spin) and a continuum of non-interacting fermionic or bosonic states. The investigation of quantum impurity models is a starting point towards the understanding of more complex strongly correlated systems, but quantum impurity models also provide the description of various correlated mesoscopic structures, biological and chemical processes, atomic physics and describe phenomena such as dissipation or dephasing. Prototypes of these models are the Anderson impurity model, or the single- and multi-channel Kondo models. The first two models are classic examples of Fermi liquid models, while the multi-channel Kondo model is the most basic example of a non-Fermi liquid system, and as such, it serves possibly as the simplest realization of a quantum critical state.

The solution of these models for low energies was a major issue in theoretical condensed matter research and led to the development of various non-perturbative techniques. [The interested readers are referred to the seminal book of Hewson [1] and to the extensive review of Cox and Zawadowski [2].] However, despite the extensive effort, many of the methods developed are uncontrolled, while others can be applied only to a subclass of models or to restricted regions of the parameter space. Wilson's numerical renormalization method, originally developed for the Kondo model, remained possibly the most reliable method to study dynamical correlations of generic quantum impurity models as well as their thermodynamic properties and finite size spectra. It is still one of the most popular methods to study quantum impurity models.

Although Wilson's NRG has been used in its original form for a longtime, a number of new developments took place recently: First, a spectral sum-conserving density matrix NRG approach (DM-NRG) has been developed [3, 4], which has very recently been generalized for non-Abelian symmetries [5]. We remark that using symmetries as much as possible is necessary in many cases to perform accurate enough calculations. NRG has been also restructured as a matrix product state approach [6]. These new developments not only made NRG much more reliable than Wilson's original method [5, 7], but they made it possible to extend NRG to study non-equilibrium phenomena [8, 9], and opened the way to use methods familiar from the density matrix renormalization group (DMRG) approach [10].

In this manual we do not intend to give a complete description of the NRG machinery. Rather, we introduce some of the basic concepts that are needed to *use NRG* and to *use the code* we provide for quantum impurity problems of interest. If you want to understand in detail, how NRG and DM-NRG work, we advise you to consult Wilson's original work [11, 12], and the references listed above.

The code we describe in this manual is a free density matrix numerical renormalization group (DM-NRG) code, which can be downloaded from the site `http://www.phy.bme.hu/~dmnrg`. This code is a flexible NRG code, which uses user-defined non-Abelian symmetries dynamically, computes spectral functions, expectation values of local operators for user-defined impurity

models. The code can use a uniform density of states as well as a user-defined density of states. The current version of the code assumes fermionic bath's. It uses any number of $U(1)$, $SU(2)$ charge $SU(2)$ or Z_2 symmetries, but the interested user can teach the code other symmetries too. The code runs using a simple input file. We provide several example input files with the code as well as a few Mathematica files with which these input files can easily be constructed. An energy spectrum analyzer utility is also provided with the code to study finite size spectra too.

Chapter 2

A short introduction to DM-NRG and to the use of symmetries

In this chapter, we give a short overview of Wilson's numerical renormalization group (NRG) and the density matrix NRG (DM-NRG). As already mentioned in the introduction, here we introduce only the basic concepts that are needed to *use NRG* and to *use the code* we provide for quantum impurity problems, but we do not attempt/intend to give a complete overview of the existing literature. To learn more details about NRG and DM-NRG, we recommend to read Wilson's original work [11, 12].

2.1 Wilson's NRG

2.1.1 The simplest example: The Kondo model

Before introducing the general concepts, let us discuss the simplest possible example, the Kondo model. The Kondo model consists of a spin S interacting locally with a non-interacting conduction electron sea,

$$H_{Kondo} = \frac{J}{2}\vec{S}\sum_{\sigma,\sigma'}\psi^\dagger_\sigma\vec{\sigma}_{\sigma,\sigma'}\psi_{\sigma'} + H_{\rm cond}\ . \tag{2.1}$$

Here J denotes the Kondo coupling, $\psi^\dagger_\sigma$ creates a conduction electron at the impurity site, and $\vec{\sigma}_{\sigma,\sigma'}$ is the vector of Pauli spin matrices. The term, $H_{\rm cond}$, describes the conduction electron bath, and its specific form is not very important for us.

Wilson's mapping and iterative diagonalization

From the point of view of local dynamics, $H_{\rm cond}$ contains a lot of redundant information: In fact, since our fermions are non-interacting, the *local density of states* $\varrho(\omega)$, i.e. the spectral function of the unperturbed Green's function, $G_{\psi_\sigma,\psi^\dagger_\sigma}(\omega)$, is generated by $H_{\rm cond}$ with $J = 0$. Note that $\varrho(\omega)$ determines completely the correlation functions of $\psi^\dagger_\sigma$ and the spin dynamics even for the interacting system, $J \neq 0$. The ingenious idea of Wilson was to discretize logarithmically $\varrho(\omega)$ using a discretization parameter $\Lambda > 1$, and thus map approximately the Hamiltonian (2.1) to a semi-infinite chain (see Fig. 2.1 and Fig. 2.2):

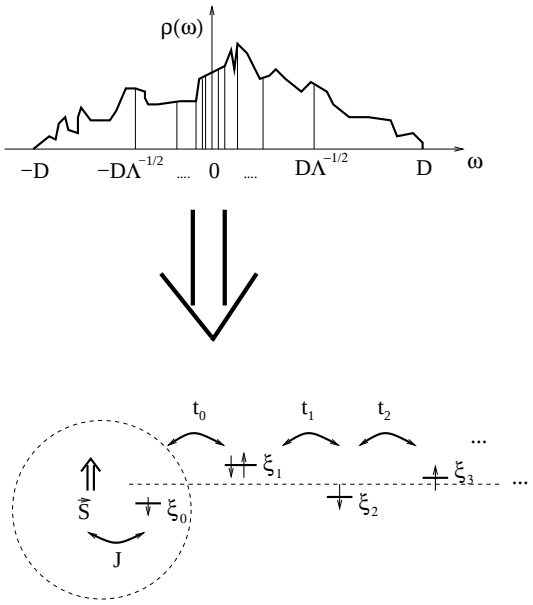

Figure 2.1: Discretization of the conduction band density of states on a logarithmic mesh and mapping onto the Wilson chain (see also Fig. 2.2).

$$H_{Kondo}^{Wilson} = \frac{J}{2}\vec{S}\sum_{\sigma,\sigma'} f_{0,\sigma}^{\dagger}\vec{\sigma}_{\sigma,\sigma'} f_{0,\sigma'} + \sum_{n=0}^{\infty}\sum_{\sigma} \xi_n \; f_{n,\sigma}^{\dagger} f_{n,\sigma} + \sum_{n=0}^{\infty}\sum_{\sigma} t_n \; (f_{n,\sigma}^{\dagger} f_{n+1,\sigma} + h.c.) \; . \quad (2.2)$$

Here the spin interacts only with the fermion $f_{0,\sigma}^{\dagger}$ at the end of the Wilson chain. The operator $f_{0,\sigma}^{\dagger}$ creates a conduction electron at the impurity site, and is essentially identical to the operator $\psi_{\sigma}^{\dagger}$. The on-site energies ξ_n and the hoppings t_n depend solely on $\varrho(\omega)$ and can be determined recursively [11]. Our code takes care of this part of the work: it determines numerically these constants if a density of states $\varrho(\omega)$ is provided (see section 4.4.3 and the description of the utility `he`).
For a flat and symmetrical density of states, $\varrho(\omega) = 1/(2D)$, one has $\xi_n = 0$ and the t_n's can be determined analytically ($t_n \sim \Lambda^{-n/2}$) [11]. Longer and longer chains give more and more accurate description of the infinite chain. This observation led Wilson and his co-workers to solve the Hamiltonian (2.2) iteratively. Introducing the operator

$$H_n \equiv \frac{J}{2}\vec{S}\sum_{\sigma,\sigma'} f_{0,\sigma}^{\dagger}\vec{\sigma}_{\sigma,\sigma'} f_{0,\sigma'} + \sum_{m=0}^{n}\sum_{\sigma} \xi_m \; f_{m,\sigma}^{\dagger} f_{m,\sigma} + \sum_{m=0}^{n}\sum_{\sigma} t_m \; (f_{m,\sigma}^{\dagger} f_{m+1,\sigma} + h.c.) \; . \quad (2.3)$$

one has the obvious recursion relation,

$$H_{n+1} = H_n + \tau_{n,n+1} + \mathcal{H}_{n+1} \; , \quad (2.4)$$

with the notation, $\tau_{n,n+1} = t_n \sum_{\sigma}(f_{n,\sigma}^{\dagger} f_{n+1,\sigma} + h.c.)$ and $\mathcal{H}_{n+1} = \sum_{\sigma} \xi_{n+1} \; f_{n+1,\sigma}^{\dagger} f_{n+1,\sigma}$ (see also Fig. 2.2). Then Wilson's procedure consists of constructing from the low-energy eigenstates, $|u\rangle_n$, of the operator H_n approximate eigenstates, $|\tilde{u}\rangle_{n+1}$, of the operator H_{n+1}. To do this, one takes a definite number of the states $|u\rangle_n$ with the lowest energies and generates new states from them by first adding an empty site, $|u\rangle_n \rightarrow |u\rangle_{n+1}$ and then using the operators $f_{n+1,\sigma}^{\dagger}$ to

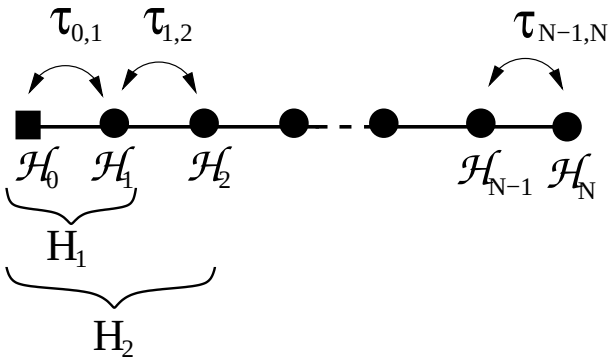

Figure 2.2: Wilson chain of length N. The impurity site is at the origin and is represented by a square, while the sites are represented as circles. For the impurity the Hamiltonian $\mathcal{H}_0$ is identical with H_0. As we add sites the iterative construction of the Hamiltonian at a given site H_n is pictured.

create new states as

$$|u\rangle_n \rightarrow \begin{pmatrix} |u\rangle_{n+1} \\ f^\dagger_{n+1,\uparrow}|u\rangle_{n+1} \\ f^\dagger_{n+1,\downarrow}|u\rangle_{n+1} \\ f^\dagger_{n+1,\uparrow}f^\dagger_{n+1,\downarrow}|u\rangle_{n+1} \end{pmatrix} . \tag{2.5}$$

Then one diagonalizes the Hamiltonian H_{n+1} in this new basis set. To construct the matrix elements of H_{n+1} in this new basis, one needs the following information

1. The eigenvalues E^n_u of H_n,

2. The matrix elements of $\mathcal{H}_{n+1}$ between the *local states*, $|\mu\rangle$, constructed from the vacuum state $|0\rangle$ as

$$\{|\mu\rangle\} \equiv \begin{pmatrix} |0\rangle \\ f^\dagger_{n+1,\uparrow}|0\rangle, \\ f^\dagger_{n+1,\downarrow}|0\rangle \\ f^\dagger_{n+1,\uparrow}f^\dagger_{n+1,\downarrow}|0\rangle \end{pmatrix} \tag{2.6}$$

3. The matrix elements of $f^\dagger_{n+1,\sigma}$ between the *local states*, $|\mu\rangle$, and

4. The matrix elements of $f^\dagger_{n,\sigma}$ between the *block states* $|u\rangle_n$.

Diagonalizing the Hamiltonian H_{n+1} one then obtains the new eigenstates $|\tilde{u}\rangle_{n+1}$, their eigenvalues, $E^{n+1}_{\tilde{u}}$, and one can trivially compute the matrix elements of $f^\dagger_{n+1,\sigma}$ between these states too, to proceed to the next iteration.
If one is to compute the spectral function of a local operator A acting at site $n = 0$, then one has to keep track of the matrix elements ${}_n\langle u|A|v\rangle_n$ of this operator, too. Already the finite size spectrum, i.e. the spectrum of H_n contains a lot of precious information [11]. However, once the matrix elements and the approximate spectrum of the Hamiltonian is at hand, one can go ahead and also compute thermal expectation values or spectral functions from it [13, 14], and thus determine completely the properties of a quantum impurity problem.

Symmetries in the Kondo model

The total spin operator

$$\vec{S}_T = \vec{S} + \frac{1}{2}\sum_{n=0}^{\infty}\sum_{\sigma,\sigma'\in\{\uparrow,\downarrow\}} f^\dagger_{n,\sigma}\vec{\sigma}_{\sigma,\sigma'}f_{n,\sigma'} \tag{2.7}$$

as well as the charge operator,

$$Q = \sum_{n=0}^{\infty}\left(\sum_{\sigma=\{\uparrow,\downarrow\}} f^\dagger_{n,\sigma}f_{n,\sigma} - 1\right) \tag{2.8}$$

commute with Eq. (2.2). This implies that the eigenstates of the Hamiltonian can be classified as *multiplets*. Every multiplet u will be characterized by a charge quantum number, $Q(u)$ and a spin quantum number, $S(u)$. We shall refer to these as *representation indices*. Internal states within a multiplet are labeled by the z-component of the spin, that we shall refer to as *labels*. Similarly, operators can also be characterized by quantum numbers. To give an example, in this simple case, the components of $\{f^\dagger_{n,\uparrow}, f_{n,\downarrow}\}$ and $\{f^\dagger_{n,\downarrow}, -f_{n,\uparrow}\}$ transform under spin rotations as the components $|\pm 1/2\rangle$ of a spin $S_f = 1/2$ state, and have charges $Q_{f^\dagger} = 1/2$ and $Q_f = -1/2$, respectively. The three operators formed from the impurity spin, $\mathcal{S}_m \equiv \{-S^+/\sqrt{2}, S^z, S^-/\sqrt{2}\}$ transform, on the other hand, as the three components of a spin $S_S = 1$ state, $|m = 0, \pm\rangle$, while they have trivially charge $Q_S = 0$. The operators discussed before provide simple examples of irreducible tensor operator multiplets $\{A\}$. Notice the non-trivial prefactors in the previous examples. These prefactors must always be worked out carefully, since the smallest sign mistake may alter the NRG results significantly.

In the presence of symmetries, a powerful theorem, the Wigner–Eckart theorem tells us that, in order to compute the matrix elements of a member of an operator multiplet $\{A\}$ between two states within multiplets u and v a single matrix element is needed, the so-called reduced matrix element [5]:

$$\langle u||A||v\rangle \ . \tag{2.9}$$

Wilson's procedure thus gets a bit modified. First of all, one needs to classify not only the block states (multiplets) but also the local states μ added at site $n+1$ by symmetries. Along the iteration, one constructs from these new states using group-theoretical methods (Clebsch-Gordan coefficients), which are already eigenstates of $\vec{S}^2_T$ and S^z. Apart from this twist, the discussion of the previous subsection is still valid, and gets only slightly modified. We can thus make the following statement: To construct the (reduced) matrix elements of $\langle i||H_{n+1}||j\rangle$ in this basis, one needs the following information

1. The eigenvalues E^n_u of H_n,
2. The matrix elements $\langle\mu||\mathcal{H}_{n+1}||\nu\rangle$ between the *local multiplets*.
3. The reduced matrix elements, $\langle\mu||f^\dagger_{n+1}||\nu\rangle$,
4. The reduced matrix elements ${}_n\langle u||f^\dagger_n||v\rangle_n$ between the *block multiplets*, $|u\rangle_n$.
5. Finally, to compute a spectral function of an operator multiplet A, one also needs ${}_n\langle u||A||v\rangle_n$.

At this point, the reader does not need to know in detail, how the construction of H_{n+1} takes place, since the code does that automatically, but for more details, the interested reader is referred to Ref. [5].

Block states	Q	S_z
$\|1\rangle = \|\Downarrow\rangle$	-1	$-\frac{1}{2}$
$\|2\rangle = \|\Uparrow\rangle$	-1	$\frac{1}{2}$
$\|3\rangle = f^\dagger_{0,\downarrow}\|\Downarrow\rangle$	0	-1
$\|4\rangle = \frac{1}{\sqrt{2}}\left(f^\dagger_{0,\downarrow}\|\Uparrow\rangle - f^\dagger_{0,\uparrow}\|\Downarrow\rangle\right)$	0	0
$\|5\rangle = \frac{1}{\sqrt{2}}\left(f^\dagger_{0,\downarrow}\|\Uparrow\rangle + f^\dagger_{0,\uparrow}\|\Downarrow\rangle\right)$	0	0
$\|6\rangle = f^\dagger_{0,\uparrow}\|\Uparrow\rangle$	0	1
$\|7\rangle = f^\dagger_{0,\uparrow}f^\dagger_{\downarrow}\|\Downarrow\rangle$	1	$-\frac{1}{2}$
$\|8\rangle = f^\dagger_{0,\uparrow}f^\dagger_{\downarrow}\|\Uparrow\rangle$	1	$\frac{1}{2}$

Added/local states	q	s_z
$\|1\rangle = \|0\rangle$	-1	0
$\|2\rangle = f^\dagger_{n+1,\downarrow}\|0\rangle$	0	$-\frac{1}{2}$
$\|3\rangle = f^\dagger_{n+1,\uparrow}\|0\rangle$	0	$\frac{1}{2}$
$\|4\rangle = f^\dagger_{n+1,\uparrow}f^\dagger_{n+1,\downarrow}\|0\rangle$	1	0

Table 2.1: Block states (left) and local/added states (right) for the single channel Kondo model when $U_{charge}(1) \times U_{spin}(1)$ symmetry is used. The first block states are formed from the impurity spin and the conduction electron at site $n = 0$ of the Wilson chain.

Kondo model with $U_{charge}(1) \times U_{spin}(1)$ symmetry

Let us first illustrate the concepts introduced above through the simplest case, when every symmetry used is Abelian. In this case every multiplet consists of a single state, which is characterized by some quantum numbers. Let us thus consider the Kondo model with two $U(1)$ symmetries, generated by the z-component of the total spin, S_z and the charge operators, Q. Now S_z and Q, provide the quantum numbers according to which the multiplets of the Hamiltonian are classified. The classification of the states in this case is rather trivial and the computation of the reduced matrix elements is also simple, since all the Clebsch-Gordan coefficients are 1 or 0. However, without spin $SU_{spin}(2)$ symmetry, the hoppings of $f^\dagger_{n,\uparrow}$ and $f^\dagger_{n,\downarrow}$ are not related by symmetry. There are therefore *two hopping operators* which we need to keep track of, $f^\dagger_{n,\uparrow}$ and $f^\dagger_{n,\downarrow}$.

The initial block states and the matrix elements of the hopping operators can be constructed immediately. A *Mathematica* file, `kondo_U1_charge_U1_spin.nb`, with all the numerical details of this calculation as well as an input file is provided by the package. The corresponding block and the local states are listed in Table 2.1.

In the first iteration, the reduced matrix elements of the Hamiltonian matrix between the block

states (Table 2.1) are given by

$$ {}_0\langle u||H_0||v\rangle_0 = J \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} . \tag{2.10} $$

In general there might be off-diagonal matrix elements in this matrix. The code takes care of this and diagonalizes ${}_0\langle u||H_0||v\rangle_0$ before proceeding to the next iteration.
The matrix elements of the $f^\dagger_{0,\uparrow}$ are

$$ {}_0\langle u||f^\dagger_{0,\uparrow}||v\rangle_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 & 0 \end{pmatrix} , \tag{2.11} $$

while those of $f^\dagger_{0,\downarrow}$ are given as

$$ {}_0\langle u||f^\dagger_{0,\downarrow}||v\rangle_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{pmatrix} . \tag{2.12} $$

These matrix elements are needed to construct the hopping terms in the first iteration. In addition, one also needs the matrix elements of the hopping operators $f^\dagger_{n+1,\uparrow}$ and $f^\dagger_{n+1,\downarrow}$ between the *local states*, listed in Table 2.1. These are given by

$$ \langle \mu||f^\dagger_{n+1,\uparrow}||\nu\rangle = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} , \qquad \langle \mu||f^\dagger_{n+1,\downarrow}||\nu\rangle = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} . \tag{2.13} $$

Finally, if we want to determine the auto correlation function of S_z, e.g., then we also need to compute the matrix elements this operator,

$$ {}_0\langle u||S_z||v\rangle_0 = \begin{pmatrix} -1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 \end{pmatrix} . \tag{2.14} $$

Block states	Q	S
$\|1\rangle = \|\Uparrow\rangle$	-1	$\frac{1}{2}$
$\|2\rangle = \frac{1}{\sqrt{2}}(f^\dagger_{0,\downarrow}\|\Uparrow\rangle - f^\dagger_{0,\uparrow}\|\Downarrow\rangle)$	0	0
$\|3\rangle = f^\dagger_{0,\uparrow}\|\Uparrow\rangle$	0	1
$\|4\rangle = f^\dagger_{0,\uparrow}f^\dagger_{0,\downarrow}\|\Uparrow\rangle$	1	$\frac{1}{2}$

Local states	q	s
$\|1\rangle = \|0\rangle$	-1	0
$\|2\rangle = f^\dagger_{n+1,\uparrow}\|0\rangle$	0	$\frac{1}{2}$
$\|3\rangle = f^\dagger_{n+1,\uparrow}f^\dagger_{n+1\downarrow}\|0\rangle$	1	0

Table 2.2: Block states (left) and local/added states (right) for the single channel Kondo model when $U_{charge}(1) \times SU_{spin}(2)$ symmetry is used. The first block states are formed from the impurity spin and the conduction electron at site $n = 0$ of the Wilson chain.

We remark at this point that, while S_z is an irreducible tensor operator and it has a spin quantum number 0, the operators S_x and S_y are not irreducible tensor operators with respect to the $U_{spin}(1)$ spin rotations, since they transform among each other. They are, however, linear combinations of the irreducible tensor operators, $S_\pm$, which have spin quantum numbers ± 1 under rotations around the z-axis.

Once all this information typed into the input file, `input.dat`, the flexible DM-NRG code is ready to compute the finite size spectrum, the spin's spectral function, and the real part of the retarded spin-spin correlation function using both traditional NRG as well as DM-NRG methods! An example input file for this case is provided in the directory `input_files`.

Kondo model with $U_{charge}(1) \times SU_{spin}(2)$ symmetry

Let us now show on the example of the Kondo model, how non-Abelian symmetries can be used. Let us thus enumerate all necessary information that is needed to perform a calculation for the spin spectral function of the single channel Kondo model with $U_{charge}(1) \times SU_{spin}(2)$ symmetry. A *Mathematica* file (`kondo_U1_charge_SU2_spin.nb`) where these inputs have been computed is also provided by the package.

In the first iteration we have four different block multiplets, $u = 1, \ldots, 4$ formed from the impurity spin and the conduction electron at site $n = 0$, while there are three added multiplets in each iteration, $\mu = 1, 2, 3$. These states and their quantum numbers are listed in Table 2.2 (only highest weight states are given). The reduced matrix elements of the Hamiltonian between the block states then read

$$ {}_0\langle u||H_0||v\rangle_0 = J \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -3/4 & 0 & 0 \\ 0 & 0 & 1/4 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} . \tag{2.15} $$

In this case, the matrix elements of $f^\dagger_{0,\uparrow}$ and $f^\dagger_{0,\downarrow}$ are related with each other by symmetry, and therefore they form a single hopping operator multiplet $f^\dagger_0$ of spin 1/2. The matrix elements of

Block states	Q	S
$\|1\rangle = \frac{1}{\sqrt{2}}(f^\dagger_{0,\downarrow}\|\Uparrow\rangle - f^\dagger_{0,\uparrow}\|\Downarrow\rangle)$	0	0
$\|2\rangle = f^\dagger_{0,\uparrow}\|\Uparrow\rangle$	0	1
$\|3\rangle = f^\dagger_{0,\uparrow}f^\dagger_{0,\downarrow}\|\Uparrow\rangle$	$\frac{1}{2}$	$\frac{1}{2}$

Local states	q	s
$\|1\rangle = f^\dagger_{n+1,\uparrow}\|0\rangle$	0	$\frac{1}{2}$
$\|2\rangle = f^\dagger_{n+1,\uparrow}f^\dagger_{n+1,\downarrow}\|0\rangle$	$\frac{1}{2}$	0

Table 2.3: Block states (left) and local/added states (right) for the single channel Kondo model when $SU_{charge}(2) \times SU_{spin}(2)$ symmetry is used. The first block states are formed from the impurity spin and the conduction electron at site $n = 0$ of the Wilson chain.

$f_0^\dagger$ need to be determined using the Wigner–Eckart theorem, and are given by

$$ {}_0\langle u||f_0^\dagger||v\rangle_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & -\sqrt{3/2} & 0 \end{pmatrix} . \tag{2.16} $$

To generate the hopping, we also need to know the reduced matrix elements of $f^\dagger_{n+1}$ between the added (local) states. These are computed as

$$ \langle\mu||f^\dagger_{n+1}||\nu\rangle \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -\sqrt{2} & 0 \end{pmatrix} . \tag{2.17} $$

Finally, to compute the spin-spin correlation function, we need to compute the reduced matrix elements of the impurity spin. As discussed before, the components of the spin operator are grouped in this case into a single tensor operator, $\mathcal{S}$. The reduced matrix elements of this operator are computed as

$$ {}_0\langle u||\mathcal{S}||v\rangle_0 = \begin{pmatrix} \sqrt{3/4} & 0 & 0 & 0 \\ 0 & 0 & -\sqrt{3/4} & 0 \\ 0 & 1/2 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & \sqrt{3/4} \end{pmatrix} . \tag{2.18} $$

To run the code with $U_{charge}(1) \times SU_{spin}(2)$ symmetry, one only needs to type this information into the input file, `input.dat`, and then run the code. You can find the corresponding input file, `kondo_model_U1_c_SU2_s.dat` in the directory `input_files`.

Kondo model with $SU_{charge}(2) \times SU_{spin}(2)$ symmetry

The Hamiltonian (2.2) is invariant under the action of $SU_{charge}(2)$ rotations in charge space, $\mathcal{U}_c = e^{i\vec{\omega}_Q \vec{Q}}$, generated by the operators $Q^x = (Q^+ + Q^-)/2$, $Q^y = (Q^+ - Q^-)/2i$ and Q^z with

$$\begin{aligned} Q^+ &= \sum_{n=0}^{\infty} (-1)^n f^\dagger_{n,\uparrow} f^\dagger_{n,\downarrow} \,, \\ Q^z &= \frac{1}{2} \sum_{n=0}^{\infty} \left(\sum_{\mu=\{\uparrow,\downarrow\}} f^\dagger_{n,\mu} f_{n,\mu} - 1 \right) \,, \\ Q^- &= \left(Q^+\right)^\dagger \,, \end{aligned} \tag{2.19}$$

and parametrized by real, three-component vectors $\vec{\omega}_Q$. Since the spin symmetry generators commute with the charge symmetry generators, the Hamiltonian (2.2) possesses a symmetry, $SU_{charge}(2) \times SU_{spin}(2)$.

Charge SU(2) symmetries must always be used with care, and one must always carefully check if a given set of operators forms an irreducible tensor operator. One can check, e.g., that the correctly defined hopping operators read, $\gamma^\dagger_n = \{(\gamma_n)_{\sigma\tau}\} = \{f^\dagger_{n,\uparrow},\ f^\dagger_{n,\downarrow},\ (-1)^n\ f_{n,\downarrow},\ (-1)^{n+1}\ f_{n,\uparrow}\}$, and that they indeed transform as a set of spin $S = 1/2$ and charge spin $Q = 1/2$ irreducible tensor operators. Notice the curious sign-dependence of the last two operators and their relative signs. Similar factors of $(-1)^n$ appear in the states generated from the highest weight states in Table 2.3 through the action of the operator Q^-.

In the first iteration we have three different block multiplets, $u = 1, \ldots, 3$ formed from the impurity spin and the conduction electron at site $n = 0$. There are two added (local) multiplets in each iteration, $\mu = 1, 2$. These states and their quantum numbers are listed in Table 2.3 (only highest weight states are given). The reduced matrix elements of the Hamiltonian between the block states then read

$$ {}_0\langle u||H_0||v\rangle_0 = J \begin{pmatrix} -3/4 & 0 & 0 \\ 0 & 1/4 & 0 \\ 0 & 0 & 0 \end{pmatrix} \,. \tag{2.20}$$

The matrix elements of $\gamma^\dagger_0$ need to be determined using the Wiegner-Eckart theorem, and are given by

$$ {}_0\langle u||\gamma^\dagger_0||v\rangle_0 = \begin{pmatrix} 0 & 0 & -\sqrt{2} \\ 0 & 0 & -\sqrt{2} \\ \frac{1}{\sqrt{2}} & -\sqrt{\frac{3}{2}} & 0 \end{pmatrix} \,. \tag{2.21}$$

To generate the hopping, we also need to know the reduced matrix elements of $\gamma^\dagger_1$ between the added (local) states. These are computed as

$$ \langle \mu||\gamma^\dagger_1||\nu\rangle = \begin{pmatrix} 0 & \sqrt{2} \\ -\sqrt{2} & 0 \end{pmatrix} \,. \tag{2.22}$$

Finally, to determine the spin spectral function, we need to compute the reduced matrix elements of the impurity spin operator:

$$ {}_0\langle u||\mathcal{S}||v\rangle_0 = \begin{pmatrix} 0 & -\sqrt{3/4} & 0 \\ 1/2 & 1/\sqrt{2} & 0 \\ 0 & 0 & \sqrt{3/4} \end{pmatrix} \,. \tag{2.23}$$

To use the code with $SU_{charge}(2) \times SU_{spin}(2)$ symmetry, one only needs to type this information into the input file, `input.dat`, and then run the code. You can find the corresponding input file, `kondo_model_SU2_c_SU2_s.dat` in the directory `input_files`, and a Mathematica notebook to compute these matrix elements is also provided.

2.1.2 Extension to arbitrary symmetries

The procedure outlined above can be generalized to essentially any number and type of non-Abelian symmetries and to any Hamiltonian of the form:

$$H = \mathcal{H}_0 + \sum_{n=0}^{\infty} \left(\tau_{n,n+1} + \mathcal{H}_{n+1}\right) \ . \tag{2.24}$$

The first term in Eq. (2.24), $\mathcal{H}_0$, describes the quantum impurity coupled to the 'environment'. This is thus the 'interaction part' of the Hamiltonian. The dynamics of the environment is described by hopping terms that connect nearest-neighbors only, $\tau_{n,n+1}$, and the on-site terms $\mathcal{H}_n$. As already stated before, these on-site terms can account for the breaking of electron-hole symmetry or for the presence of superconducting correlations [15].
The above Hamiltonian can also be diagonalized iteratively using the recursion relation, (2.4). The procedure is essentially the same as the one outlined in the previous subsection, with the modification that we now may have Γ different symmetries, which commute with the Hamiltonian and each other, and which we shall label by an index $\gamma = 1, \ldots, \Gamma$. Correspondingly, multiplets will be labeled by Γ different quantum numbers, $\{Q^\gamma\} = \{Q^1, \ldots, Q^\Gamma\}$, with Q^γ the label of the irreducible representation of symmetry γ. States within a degenerate multiplet can then be labeled by internal quantum numbers, which, by analogy to the spin operators, we shall denote by Q^γ_z. Finally, the representation indices and the internal quantum numbers can both be grouped into vectors, $\underline{Q} = \{Q^1, Q^2, ..., Q^\Gamma\}$ and $\underline{Q^z} = \{Q^1_z, Q^2_z, ..., Q^\Gamma_z\}$.
Similar to multiplets, a set of irreducible tensor operators, $\{A\}$ is also characterized by some quantum numbers $\underline{a}$, and $\underline{a}^z$. The hopping terms, $\tau_{n,n+1}$, can also be decomposed using a set of such irreducible operators, which we shall call 'hopping operators', $f^\dagger \to C$, and label them by λ. Thus the hopping operator multiplet, C_λ has quantum numbers $\underline{c}_\lambda$ and members of this multiplet are labeled by $\underline{c}^z_\lambda$. Ordering the members of a hopping operator multiplet into vectors, $\underline{C}_\lambda \equiv \{C_{c_\lambda, c^z_\lambda}\}$, the hopping can be simply written as

$$\tau_{n,n+1} = \sum_\lambda t_{n,\lambda} \left(\underline{C}_{n,\lambda} \cdot \underline{C}^\dagger_{n+1,\lambda} + h.c. \right) \ . \tag{2.25}$$

Note that here different hopping operator multiplets may have different hopping amplitudes, and also, remember that now the $\underline{C}_{n,\lambda}$'s denote *creation operators*. An example for having two operators has already been for the Kondo model with $U_{charge}(1) \times U_{spin}(1)$ symmetries, but in most cases several hopping operators must be used for multichannel models too.
In our general framework the iteration step $n \to n+1$ reads now as follows:

- Previous block states u can be characterized by their symmetry indices and labels, $\underline{Q}_u$, $\underline{Q}^z_u$, $\left| u, \underline{Q}_u, \underline{Q}^z_u \right\rangle_n$. Similarly, local states μ are also characterized by their symmetry indices $\underline{q}_\mu = \{q^\gamma_\mu\}$, and internal labels $\underline{q}^z_\mu = \{q^\gamma_{\mu,z}\}$: $\left| \mu, \underline{q}_\mu, \underline{q}^z_\mu \right\rangle$.
- The new basis states can be constructed using Clebsch-Gordan coefficients as

$$\left| i, \underline{Q}_i, \underline{Q}^z_i \right\rangle_n = \sum_{\underline{Q}^z_u, \underline{q}^z_\mu} \left\langle \underline{q}_\mu \underline{q}^z_\mu ; \underline{Q}_u \underline{Q}^z_u \middle| \underline{Q}_i \underline{Q}^z_i \right\rangle \left| u, \underline{Q}_u, \underline{Q}^z_u ; \mu, \underline{q}_\mu, \underline{q}^z_\mu \right\rangle_{n-1} \ , \tag{2.26}$$

  where we have also introduced a general notation for the Clebsch–Gordan coefficients as a product of coefficients, one for each symmetry in the problem

$$\left\langle \underline{q}_\mu \underline{q}^z_\mu ; \underline{Q}_u \underline{Q}^z_u \middle| \underline{Q}_i \underline{Q}^z_i \right\rangle = \prod_{\gamma=1}^{\Gamma} \left\langle q^\gamma_\mu q^\gamma_{\mu,z} ; Q^\gamma_u Q^\gamma_{u,z} \middle| Q^\gamma_i Q^\gamma_{i,z} \right\rangle \ . \tag{2.27}$$

- Knowing the reduced matrix elements, ${}_n\langle u \parallel C_{n,\lambda} \parallel v\rangle_n$ and $\langle \mu \parallel C_{n+1,\lambda} \parallel \nu\rangle$, and the eigenvalues E_u^n from iteration n one can construct the Hamiltonian in the new basis.

- Diagonalizing this Hamiltonian, one can determine the new eigenstates, $\left|\tilde{u}, \underline{Q}_{\tilde{u}}, \underline{Q}^z_{\tilde{u}}\right\rangle_{n+1}$ their energies, E_u^{n+1}, and the irreducible matrix elements, ${}_{n+1}\langle \tilde{u} \parallel C_{n+1,\lambda} \parallel \tilde{v}\rangle_{n+1}$. If a spectral function is computed then the corresponding operator's reduced matrix element, ${}_{n+1}\langle \tilde{u} \parallel A \parallel \tilde{v}\rangle_{n+1}$ must also be computed.

- After the iteration some of the high-energy states are also discarded before one proceeds to the next iteration.

Our flexible DM-NRG code implements this rather general procedure dynamically, and in such a way that various symmetries can be taught to the code. If you want to teach a new symmetry to the code, feel free to contact us for help.

2.2 The DM-NRG method

Wilson's procedure gives accurate results for the finite size spectrum, and it can also be used to compute spectral functions [13]. However, there are several problems with the simple extension of Wilson's method. On one hand, in the original approach, spectral functions have been computed using only the approximate ground states in a given iteration. This may give incorrect results if the fine structure of the ground state shows up at subsequent iterations only. Hofstetter proposed to use a density matrix NRG (DM-NRG) method to treat this problem [7]. Furthermore, in the original method one simply chops off parts of the Hilbert space, and therefore spectral sum rules are not respected. This problem has been solved only recently, through the introduction of a *complete basis set* [8]. In this approach, rather than constructing approximate eigenstates for a chain of length n, one constructs approximate eigenstates of H_n that live on a chain of length $N > n$, by adding *environment states*

$$\left|u, \underline{Q}_u, \underline{Q}^z_u\right\rangle_n \to \left|u, \underline{Q}_u, \underline{Q}^z_u; e\right\rangle_n \tag{2.28}$$

where e labels all possible states of the environment, i.e., chain sites from site $n+1$ to N.

The reduced density matrix

Physical quantities are then computed from the density matrix, which is approximated as

$$\varrho = \sum_{n=0}^{N} \varrho_n \ , \tag{2.29}$$

$$\varrho_n = \frac{1}{Z} \sum_e \sum_{u \in \text{Discarded}} \sum_{\underline{Q}^z_u} e^{-\beta E_u^n} \left|u, \underline{Q}_u, \underline{Q}^z_u; e\right\rangle_{n\,n} \left\langle u, \underline{Q}_u, \underline{Q}^z_u; e\right| \ , \tag{2.30}$$

where the partition function is defined as

$$\mathcal{Z} = \sum_u \dim_{\text{loc}}^{N-n} \dim(u)\ e^{-\beta E_u^n} \ , \tag{2.31}$$

where $\dim(u)$ is the dimension of the multiplet u, and $\dim_{\text{loc}} = \sum_\mu \dim(\mu)$ denotes the dimension of the local basis at a given site of the chain. The second summation in Eq. (2.31) runs over states that have been discarded in iteration $n \to n+1$.

The calculation of a spectral function then consists of a forward sweep, where we determine all eigenstates in a given iteration, and a backward sweep, where we determine ϱ and the truncated reduced density matrices, defined as

$$R^{[n]} \equiv \mathrm{Tr}_{N-n}\{\sum_{m\geq n} \varrho_m\} \,. \tag{2.32}$$

The details of the corresponding recursion relations were given in Ref. [5].

Spectral function calculations in the flexible-DM-NRG framework

The main purpose of the code is to compute the retarded Green's function of two local operators, A and B. This is defined for two irreducible tensor operators as

$$G^R_{A_{\underline{a},\underline{a}_z},B^\dagger_{\underline{b},\underline{b}_z}}(t) = -i\left\langle\left[A_{\underline{a},\underline{a}_z}(t), B^\dagger_{\underline{b},\underline{b}_z}(0)\right]\right\rangle \Theta(t)\,. \tag{2.33}$$

where the quantum numbers $\underline{a},\underline{a}_z$ and $\underline{b},\underline{b}_z$ refer to the quantum numbers of operators $A^\dagger$ and $B^\dagger$. For fermionic operators one must replace the commutator above by an anticommutator. By symmetry, this Green's function is non-zero only if the spectral operators A and B transform accordingly to the same representation,

$$G^R_{A_{\underline{a},\underline{a}_z},B^\dagger_{\underline{b},\underline{b}_z}}(t) = G^R_{A,B^\dagger}(t)\;\delta_{\underline{a},\underline{b}}\;\delta_{\underline{a}_z,\underline{b}_z} \quad . \tag{2.34}$$

Although it is a cumbersome expression, for completeness, let us give here the expression obtained generalizing the procedure of Ref.[8].

$$\begin{aligned} G^R_{A,B^\dagger}(z) = &\sum_{n=0}^{N}\sum_{i\in D,K}\sum_{(j,k)\notin(\mathrm{K},\mathrm{K})} {}_n\langle i \parallel R^{[n]} \parallel j\rangle_n \\ &\times\left[\frac{{}_n\langle k \parallel A^\dagger \parallel j\rangle^*_n\;{}_n\langle k \parallel B^\dagger \parallel i\rangle_n}{z+\frac{1}{2}(E^n_i+E^n_j)-E^n_k}\frac{\dim(k)}{\dim(a)} - \xi\,\frac{{}_n\langle j \parallel B^\dagger \parallel k\rangle_n\;{}_n\langle i \parallel A^\dagger \parallel k\rangle^*_n}{z-\frac{1}{2}(E^n_i+E^n_j)-E^n_k}\frac{\dim(i)}{\dim(a)}\right]. \end{aligned} \tag{2.35}$$

Remarkably, this formula contains exclusively the reduced matrix elements and the dimensions of the various multiplets and operator multiplets. Here the second sum is over all the multiplets i,j,k of the given iteration subject to the restriction that j,k do not belong to kept states at the same time. No summation is needed for states within the multiplets. The sign factor ξ is $\xi=1$ for bosonic operators, while it is $\xi=-1$ for fermionic operators, and $\dim(a)=\prod_{\gamma=1}^{\Gamma}\dim(a^\gamma)$ is the dimension of the operator multiplet $A^\dagger_{\underline{a},\underline{a}_z}$. The code computes the spectral function using this formula, and the full Green's function is then computed by doing a Hilbert transform numerically.

Smoothing procedure

In general, the spectral functions are given as weighted sums of δ-functions of the form:

$$\mathcal{A}_{A,B^\dagger}(\omega) = -\frac{1}{\pi}\Im m\, G^R_{A,B^\dagger}(\omega+i\,0^+) \tag{2.36}$$

$$= \sum_i w_i\,\delta(\omega-\omega_i) \tag{2.37}$$

where the weighting coefficients w_i can be expressed from Eq. (2.35) after analytical continuation to the real axis. To get a smooth spectral function, however, the delta functions need to

be replaced by some broadening kernels $K(\omega, \omega_i)$ that fulfill the condition

$$\int d\omega \; K(\omega, \omega_i) = 1 \; . \tag{2.38}$$

This condition guarantees that spectral sum rules remain valid after the broadening procedure. In our code we use two types of kernels to generate the real axis spectral function: (a) the *log-Gaussian* kernel [16] and (b) the *interpolative kernel* that we constructed to avoid certain difficulties with the interpolation around $\omega = 0$.
For the log-Gaussian method the following kernel is used

$$K_{log}(\omega, \omega_i) \equiv \frac{1}{b\sqrt{\pi}} e^{-b^2/4} e^{-(\ln\omega - \ln\omega_i)^2/b^2} \frac{1}{\omega_i}. \tag{2.39}$$

This kernel works rather well at $T = 0$ calculations, however, it extrapolates to the $\omega \to \pm 0$ values in a singular way. Therefore, it has problems at finite temperatures and also in cases where spectral losses lead to an artificial jump in the spectral function at $\omega = 0$.
The interpolative scheme avoids this difficulty by using a kernel

$$K_{int}(\omega, \omega_i) \equiv \frac{1}{b\sqrt{\pi}} e^{-[x(\omega) - x(\omega_i)]^2/b^2} \frac{dx}{d\omega_i} \; , \tag{2.40}$$

with the smoothening function defined as

$$x(\omega) = \frac{1}{2} \; \tanh\left(\frac{\omega}{T_Q}\right) \; \ln\left(\left(\frac{\omega}{T_Q}\right)^2 + e^{\gamma}\right) \; , \tag{2.41}$$

by choosing $\gamma \approx T_Q$ e.g. This formula interpolates smoothly along the real axis for all frequencies. It has the property that for frequencies smaller than the 'quantum temperature', $|\omega| \ll T_Q$, the broadening is Gaussian, while for $|\omega| \gg T_Q$ it becomes log-Gaussian. In this scheme, for zero temperature calculations the quantum temperature T_Q must be the smallest energy scale in the problem, on the other hand for finite temperature calculations, the quantum temperature must be set to be in the range of the temperature itself, $T_Q \sim 0.5T \div T$. In both interpolative methods there is a fitting parameter b, which controls the widths of kernels and in general is Λ dependent, which should be in the range of $b \sim \sqrt{\Lambda}/2$ (for $\Lambda = 2$, $b = 0.5 \div 0.9$).

Chapter 3

Installation and technical support

3.1 Memory and disk space requirements

The resources required by the DM-NRG code depend a lot on the type of the ongoing calculation. Increasing the number of block states that are kept during a run, the CPU time needed for performing the calculation, the necessary memory, as well as the used disk space may increase substantially. As an example in Table 3.1 we present some estimates for a simple model. The following information was extracted by running the code on an Intel(R) Core(TM)2 Duo CPU T7250 @ 2.00GHz processor with 4MB CPU cache. We have used the CentOS 5.1 distribution for the Linux operating system. The code was run by using an input file for the Anderson model with U(1) $\times$ SU(2) symmetries that comes with the distribution. We made a run for a flat band and performed 50 iterations. The compiler used was gcc version 4.1.2.

Model name	number of block states	CPU-type	CPU time (sec)	disk space (MB)	memory (MB)
Anderson model	50	Intel Core(TM)2 Duo	1.27	12	4
U(1) $\times$ SU(2)	100	CPU T7250 @ 2.00GHz	3.56	25	7
symmetries	200	with 4MB cache	14.39	69	12
	500		156.26	341	18

Table 3.1: CPU time, memory and disk space requirements.

Important note: when temporary files are also generated in ASCII format by setting **text_swap_files_flag = ON** in <SECTION-FLAGS> of the input file `input.dat`, approximately four times larger disk space is required than the ones shown in the table.

3.2 Compilation environment

The code can be compiled with either `g++` or with `icc` (intel c++ compiler). The Intel compiler, `icc` is available for free for non-commercial and non-academic use after signing a license agreement.

3.3 Libraries

The Flexible-DM-NRG code is linked with state of the art libraries. The following list of libraries (packages) are needed to have the `flexible-dmnrg` package properly installed. The easiest way to install these packages is by using the `rpm - RPM package manager` method.

For further information and links to the websites from where these libraries can be downloaded please refer to the web page of the code `http://www.phy.bme.hu/~dmnrg/index.html`.
List of required libraries:
• `libg2c` – Fortran 77 Libraries
• `blas` – Basic Linear Algebra Subprograms
• `lapack` – The LAPACK libraries for numerical linear algebra.
• `gmp` – A GNU arbitrary precision library
• `gsl` – GNU Scientific Library
• `gsl-devel` – GNU Scientific Library - development files

3.4 Automatic installation

Once the libraries were successfully installed, run the following shell script in the main directory of the code:
`./configure`
This command must run without errors. It checks your system, searches for the necessary libraries, and sets the library paths. An error message returned while running `./configure` means that some packages might be missing or not properly installed. If the `./configure` has run without errors then the `make.sys` and `make.rules` and `config.h` files have been generated. These files now contain the system variables needed for the installation [1].
Next run the following command:
`make all`
If everything goes smooth and no error messages were generated, then the code is successfully installed. If for some reasons the automatic configuration does not work, please, follow the instructions in the next subsection and install the code manually. If you do not succeed, feel free to contact us.

3.5 Manual installation

Sometimes, for some mysterious reasons, the `autoconf` does not work properly and some of the libraries are not detected. In this case, the `make.sys` file, that can be found in the main folder of the code, needs to be edited manually. Altogether there are ten shell variables that need to be edited.
• `CXX` - the name of the C++ compiler, is usually `g++` or `icc`.
• `C` - the name of the C compiler, usually `gcc`.
• `CC_FLAGS` - flags for the g++ or icc compiler (this variable can be left empty at this stage). You can set here the optimization level to speed up the code.
• `GSL_FLAGS` - path to the gsl include files, usually `-I/usr/include`.
• `GMP_FLAGS` - path to the gmp include files, usually `-I/usr/include`.
• `FLIBS` - fortran libraries, usually `g2c` or `gfortran`.
• `LAPACK_LIBS` - lapack library, usually `-llapack`. For static libraries the full path name needs to be provided sometimes: `/lib/liblapack.a`.
• `BLAS_LIBS` - blas library, usually `-lblas`. For static libraries the full path name needs to be provided sometimes: `/lib/libblas.a`.
• `GSL_LIBS` - gsl library, usually `-lgsl -lgslcblas`. For static libraries the full path name needs to be provided sometimes: `/lib/libgsl.a`.
• `GMP_LIBS` - gmp library, usually `-lgmp`. For static libraries the full path name needs to be provided sometimes: `/lib/libgmp.a`.

[1] The automatic installation checks for the `g++` and `gcc` compilers and does not check for the `icc`. To run the code with `icc`, manual intervention is necessary (see section Manual Installation), and the `CXX` environmental variable needs to be modified from `g++` to `icc`.

Once the `make.sys` file is properly edited run
`make all`
If the compilation runs without errors then the code is successfully installed and all the binaries have been generated.

3.6 Generated binaries

After a successful installation the following binaries are generated:

- **fnrg** - the main program that does the NRG/DM-NRG calculations.
- **sfb** - utility that does the spectral function broadening.
- **he** - utility that computes the hopping for an energy-dependent density of states.
- **es** - utility to generate/analyze the energy spectrum.
- **cgc** - utility to generate/compute the Clebsch-Gordan coefficients

To run the code the `input.dat` file must be set properly. For further details, refer to the next sections of the manual. Example input files are provided with the code for the single band Anderson model with $U(1)$ charge and spin symmetries and for the Kondo model with combinations of $U(1)$ and $SU(2)$ symmetries. These 'ready to run' input files are available in the `./input_files` folder. Furthermore, we also provided the Mathematica files which were used to compute the content of these input files. Some of the technical details of these calculations were summarized in Chapter 2.

3.7 BUG reports

We have made this code publicly available, so if you think you have found a bug in the code, please let us know so we can fix it for future releases.
Please send any bug report to: `dmnrg@neumann.phy.bme.hu`. Use **BUG REPORT** in the subject area.
Please include the followings in any report:
• The version number of the code.
• input.dat file so we can test the code with it.
• A description of what is wrong. If the results are incorrect, in what way. If you get a crash, etc.
• Please do not send core dumps, of executables.
• The name of the compiler that was used and its version.
• The output from running `./configure`
• If the bug is related to configuration, then the contents of `config.log`.
If the bug report is good than we will do our best to fix it. If the bug report is poor we will not do much about it, probably just ask for a better report.
If you think something in this manual is unclear, or downright incorrect, or if the language needs to be improved, please send us a note to the same address.

3.8 Some hints for developers

3.8.1 Organization of the code

The components of the program are located in three main directories. Directory `./src/` contains the `*.cc` source files, `./include/` the corresponding `.h` files and `./input_files/` the user

defined input files. The directory *./doc/* contains the documentation of the code in *LATEX* and *HTML* forms. Table 3.2 gives a hint on the structure of the code.

./	`Makefile`		
./	`DOXYFILE`		configuration for on-line documention
	`src/`	`*.cc`	source code (source files)
	`include/`	`*.h`	source code (header files)
	`input_files/`	`*.dat`	directory for input files
	`results/`		output directory of the code
	`doc/`	`nrg_user_guide.tex`	documentation of the code
		`html/`	on-line documention
		`tex/`	on-line documention

Table 3.2: Directory structure of the code.

The other files in the main directory are configurations files needed for properly installing the code.

3.8.2 On-line documentation of the code

The documentation of the source code is managed with the use of the `doxygen` a free software under `Linux`. At present version 1.4.6 is used. The configuration of `doxygene` is set in file *Doxyfile* and the program is called from shell as `doxywizard`. Description of the classes, variables and functions is included directly in the source code (in the `*.cc` and `*.h` files), therefore, all comments about new variables, functions, classes etc. are documented right away. The on-line documentation file `./doc/html/....html` can be opened with any WEB-browser. The DM-NRG reference manual in pdf form (`refman.pdf`) can be generated by the command: `latex ./doc/latex/refman.tex`.
All the information about the functions (classes) i.e., what it does, how to call, description of the input and output variables must be located in the main body of the functions, otherwise, this information will not be accessible in the output files generated by doxygene! The hierarchy of the classes, dependencies and calling functions are generated automatically by `doxygen`.

3.8.3 Version management

Version management of the code is carried out with the use of `Subversion system(svn)` a free software under `Linux`. This program is similar to `Control Version System (cvs)` but has several additional features. The WEB page of the svn interface can be accessed by the members of the *flexible DM-NRG* developers group only.
The management of this manual – in `LATEX` form–, however, has to be done manually. There are no scripts yet, that would automatically update the manual once the code has been changed. However, since all directories and files – including this manual as well – are checked in to the `svn` it is straightforward to update the manual and all changes of it are documented by the `svn` itself.

Chapter 4

Using the DM-NRG code

4.1 Main features of the code

If the installation was successful then, to run the code for your problem of interest, you first need an input file, `input.dat`. The structure and content of this file shall be described in detail later in this chapter, and a few examples are also provided with the code in the library, `./input_files`. Furthermore, we provided some *Mathematica* files in the `mathematica_files` folder, which were used to generate these input files. For the generation of the irreducible matrix elements it is sometimes important to know, what is the definition of the Clebsch-Gordan coefficients used by the code. For this purpose, a utility called **cgc** is provided in the package that computes the Clebsch-Gordan coefficients.
When running the code, all the results are written into the folder `./results`. The content of these files shall also be described later in this chapter. Finally, these results can further be analyzed with the utilities **sfb** – real axis spectral function generator and **es** – energy spectrum generator, also provided with the code, and shall be described later.

Main features included in version 1.0.0:

- Zero and finite temperature calculations of the spectral function of not necessarily the same operators.
- Static averages.
- NRG type calculations or DM-NRG computations (using the complete basis set).
- The code uses a single input file that can be easily modified. Examples are available for Anderson as well as Kondo Hamiltonians in the presence of $U(1)$ and $SU(2)$ symmetries for spin as well as charge symmetries.
- Flat density of states as well as arbitrary energy-dependent density of states can be considered. In the latter case arbitrary precision routines are used to generate the parameters of the Wilson chain.
- Utility for broadening the spectral function is provided to compute the imaginary and the real parts of the retarded Green's function

$$G^R_{A,B^\dagger}(t) = -i \left\langle \left[A(t),\, B^\dagger(0)\right]_\pm \right\rangle \Theta(t)\,, \tag{4.1}$$

  with the operators A and B not necessarily identical.

- Utility for computing the on-site energies as well as the hoppings for a non-uniform density of states is provided.

- Utility to generate/analyze the finite size spectrum is provided.

Utilities provided in the package:

- **he** - generates the hopping amplitudes along the chain and the on-site energies when an energy-dependent density of states is used. (This utility provides input files that are read directly in the **fnrg** binary code).

- **sfb** - does the broadening of the spectral function and computes the real part of the Green's function by a Hilbert transform. This utility uses the outputs of the **fnrg** to generate these data.

- **notebook examples** - there are some extra *Mathematica 6.0* notebook examples where the matrices for Hamiltonians as well as for some hopping and spectral operators are computed. These notebooks can easily be extended to other impurity models.

- **es** - generates the energy spectrum. This utility uses the output of the **fnrg** to generate the data.

- **cgc** - an interactive utility, which generates the Clebsch-Gordan coefficients used by the code.

4.2 Running the code

In this section we shall give some information on how exactly to run the code. At this point we suppose that the code was successfully compiled and linked properly with the required libraries. If this is the case there should be five binary codes in the main folder, **fnrg**, **sfb**, **he**, **cgc**, and **es**. Depending on the type of calculation that is performed, the following steps need to be followed:

4.2.1 Flat density of states

Most NRG calculations are done for a flat density of states, $\varrho(\omega) = \frac{1}{2}$, with $\omega \in [-1, 1]$. In this case there is no need for the **he** routine, since the hoppings along the chain are computed on the fly. In this case the on-site energies are all zero as a consequence of electron-hole symmetry. For this type of calculation, first the *hoppings_on_site_energies_flag* in the <SECTION-FLAGS> needs to be set to 'OFF' in the file `input.dat`. Then the **fnrg** must be run without any arguments in the command line. The name of the input file must always be `input.dat`, since the code looks for this particular file to read in the data.
Outputs are saved in a folder in `results/`. Results for the spectral function can be generated by running the **sfb** binary without any argument after exiting normally from the run. Note that **sfb** reads the same input file, `input.dat`, so do not modify it between the two runs except for the part <SECTION-SPECTRAL_FUNCTION_BROADENING>. Depending on the flags set in <SECTION-SPECTRAL_FUNCTION_BROADENING>, spectral functions, Green's functions and static averages are computed. Note that the real part of a Green's function can only be computed if the corresponding spectral function has previously been computed.

4.2.2 Energy dependent density of states

The user can also provide an arbitrary density of states (DOS) for the code. In the present version of the code, the DOS is read from a file, `dos.dat`. This file must be in the folder, `./dos_mapping`, and it must be provided by the user. This file must contain at least two columns, one of them being the energy, and the others being the corresponding density of states. There should be as many density of states columns as hopping operators are defined in the input file.
There are some constraints that the DOS must satisfy

- The DOS needs to be rescaled such that all energies are in the interval, $E \in [-1, 1]$. In other words, the energy unit of the NRG calculation (=1) must be larger than the range of the density of states.
- The DOS needs to be normalized to 1.
- The energies must be ordered.
- Each density of states column corresponds to a given hopping operator in the `input.dat` file. The columns must be ordered accordingly to the order of the hopping operators in the `input.dat` file.

There is no further requirement for the mesh, both uniform and non-uniform meshes can be provided.
When the NRG calculation is performed with an energy-dependent DOS, the hoppings and the on-site energies must be computed first. This must be done interactively, using the binary **he**. When running **he**, one needs to specify an iteration number. This can be larger than the number of iterations specified in the `input.dat` file. After a successful run of **he**, the hopping amplitudes and the on-site energies are saved in files having descriptive names such as `hopping_couplings_*.dat` and *on_site_energies_*.dat* in the folder `./dos_mapping/`.
Next, the `input.dat` file needs to be set-up: There, make sure that the `hoppings_on_site_energies_flag` in the <SECTION-FLAGS> is set to 'ON' for this type of run.
Once the hopping amplitudes and the on-site energies were computed, the same steps must be followed as in the case of a flat density of states. The code **fnrg** must be run without arguments.

`Note:` The routine **he** is using arbitrary precision libraries for computing the hoppings and the on-site energies, however due to the accumulation of the errors during the calculations sometimes the **he** utility may give unexpected results for the hoppings. Therefore, we advise the user to check the results for the hoppings before pluging them into **fnrg** for any kind of nrg calculations.

4.3 Initialization and the input file

In the initialization step – the zero'th NRG iteration step – one has to define the Hamiltonian. Therefore, for the iterative process, among others, the user has to provide the following input parameters:

1. One needs to specify the number (*symmetry_no*) and type of symmetries.
2. One needs to specify the number (*block_state_number*) of initial *block states*, i.e., multiplets that span the Hilbert space for $n = 0$ (we denote this number in this manual by K). One also needs to define the quantum numbers $\underline{Q}_u$ of the block states, $\left|u, \underline{Q}_u, \underline{Q}^z_u\right\rangle_0$.
3. *Block Hamiltonian matrix* - The matrix elements of the $K \times K$ Hamiltonian matrix H_0, defined on the K *block states*.

4. *block hopping operators* - The reduced matrix elements of all hopping operators between the block states. These are $K \times K$ matrices.

5. *spectral operators* - The reduced matrix elements of the operators, whose spectral functions we compute. These are also $K \times K$ matrices.

6. *local state number* - The number of multiplets that span the local Hilbert space (we label this number L).

7. *local states* - Quantum numbers $\underline{q}_\mu$ of the local states, $\left|\mu, \underline{q}_\mu, \underline{q}^z_\mu\right\rangle$.

8. *local hopping operators* - The reduced matrix elements of the hopping operators. These are $L \times L$ matrices.

This list is rather incomplete, and explains only briefly the meaning of the most basic variables. In addition, many other, less relevant parameters must be specified for the NRG process (broadening, maximum iteration number, etc.). All these input parameters are read from file, `input.dat`, which must be constructed for every model separately. Some examples of such input files can be found in the folder, `./input_files`, and also in the appendix of this manual. A detailed description of these parameters shall be provided in forthcoming subsections.

Structure of the input file

The input file is organized along sections, and contains a lot of comments. As a general rule, any line that starts with the symbol '#' is treated as a comment line in these input files and is ignored by the code. You can thus add additional comment lines for yourself using this convention.
The input file has the following overall structure (that we shall describe in detail in subsequent subsections):

1. <SECTION-PARAMETERS>
2. <SECTION-FLAGS>
3. <SECTION-SYMMETRIES>
4. <SECTION-BLOCK_STATES>
5. <SECTION-LOCAL_STATES>
6. <SECTION-LOCAL_STATES_SIGNS>
7. <SECTION-BLOCK_HAMILTONIAN>
   (a) <BLOCK_HAMILTONIAN_TERM>
      i. <BLOCK_HAMILTONIAN_NAME>
      ii. <BLOCK_HAMILTONIAN_COUPLING>
      iii. <BLOCK_HAMILTONIAN_REPRESENTATION_INDEX>
      iv. <BLOCK_HAMILTONIAN_MATRIX>
8. <SECTION-HOPPING_OPERATORS>
   (a) <HOPPING_OPERATOR>
      i. <HOPPING_OPERATOR_NAME>
      ii. <HOPPING_OPERATOR_REPRESENTATION_INDEX>
      iii. <HOPPING_OPERATOR_SIGN>

iv. <HOPPING_OPERATOR_MATRIX>

9. <SECTION-SPECTRAL_OPERATORS>
   (a) <SPECTRAL_OPERATOR>
      i. <SPECTRAL_OPERATOR_NAME>
      ii. <SPECTRAL_OPERATOR_REPRESENTATION_INDEX>
      iii. <SPECTRAL_OPERATOR_SIGN>
      iv. <SPECTRAL_OPERATOR_MATRIX>
10. <SECTION-STATIC_OPERATORS>
   (a) <STATIC_OPERATOR>
      i. <STATIC_OPERATOR_NAME>
      ii. <STATIC_OPERATOR_REPRESENTATION_INDEX>
      iii. <STATIC_OPERATOR_SIGN>
      iv. <STATIC_OPERATOR_MATRIX>
11. <SECTION-SPECTRAL_FUNCTION>
12. <SECTION-SPECTRAL_FUNCTION_BROADENING>
13. <SECTION-LOCAL_HOPPING_OPERATORS>
   (a) <LOCAL_HOPPING_OPERATOR>
      i. <LOCAL_HOPPING_OPERATOR_NAME>
      ii. <LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX>
      iii. <LOCAL_HOPPING_OPERATOR_SIGN>
      iv. <LOCAL_HOPPING_OPERATOR_MATRIX>
      v. <LOCAL_ON_SITE_ENERGY_MATRIX>

4.3.1 <SECTION-PARAMETERS>

In this section, the main parameters of the given model and also those of the DM-NRG method are set. The order of the parameters does not matter as long as none of them is missing.

Model parameters

• **model:** The name of the model under investigation. It can be any string. This name will be automatically included in the name of the folder, where output data are saved.
• **lambda:** The value for the logarithmic discretization parameter, Λ. It is usually fixed between 2 and 3.
• **symmetry_no:** The number of the symmetries used. The present version of the code handles $U(1)$, $SU(2)$, $Z(2)$, $charge\_SU(2)$ symmetries. The properties of the symmetries are set in <SECTION-SYMMETRIES> by defining their names and setting upper and lower bounds on their representation indices. Later versions of the code will include $SU(3)$ symmetry and crystal field symmetries too.
• **coupling_no:** The number of independent coupling constants. In general, the local Hamiltonian H_0 can contain several couplings. The on-site part of the Anderson model, e.g., can be written as

$$H_0 = U(n-1)^2 + (\epsilon_d + U/2)\, n + V \sum_\sigma (d_\sigma^\dagger f_{0,\sigma} + h.c.) \,,$$

i.e. it contains three different couplings, U, ϵ_d and V. These couple to the operators,$(n-1)^2$, n, and $(d^\dagger_\sigma f_{0,\sigma} + h.c.)$. Thus the local Hamiltonian, H_0, can be written as a sum of $coupling\_no = 3$ independent terms,

$$H_0 = \sum_{\delta=1}^{coupling\_no} G_\delta \, \mathcal{H}_\delta \, , \tag{4.2}$$

where $\mathcal{H}_\delta$'s denote the independent local terms and G_δ stand for the corresponding couplings. The matrix elements of $\mathcal{H}_\delta$ are specified in <SECTION-BLOCK_HAMILTONIAN>.
`Important note`: The number of <BLOCK_HAMILTONIAN_TERM> items must agree with the 'coupling_no' set in this section.
• **spectral_operator_no:** The number of the operators for which one wants to compute the spectral functions. The detailed parametrization of these operators is carried out in <SECTION-SPECTRAL_OPERATORS>. The number of <SPECTRAL_OPERATOR> items must agree with the 'spectral_operator_no' set in this section.
• **static_operator_no:** The number of operators for which static averages are computed. The detailed parametrization of these operators is carried out in <SECTION-STATIC_OPERATORS>. The number of <STATIC_OPERATOR> items there must agree with the 'static_operator_no' set in this section.
• **hopping_operator_no:** The number of independent hopping operators. The detailed parametrization of these operators is carried out in <SECTION-HOPPING_OPERATORS> and <SECTION-LOCAL_HOPPING_OPERATORS>.
• **block_state_no:** The number of multiplets that span the Hilbert space for iteration $n = 0$. The detailed parametrization of the states is carried out in <SECTION-BLOCK_STATES> by setting the representation indices of all the states corresponding to all the symmetries used during the calculation.
• **local_state_no:** The number of local multiplets, added in each iteration. In the present version of the code this is the same for all sites, $0 < i \leq N$. The detailed parametrization of local states is carried out in <SECTION-LOCAL_STATES>.
• **local_coupling_no:** The number of independent terms in the on-site Hamiltonian operators, $\mathcal{H}_n$. In the presence of electron-hole symmetry of the conduction band and without any other correlations (superconducting like) the on-site terms are all zero. If we have only on-site energies, ξ_n, then **local_coupling_no**$= 0$.
• **spectral_function_no:** The number of the spectral functions that should be calculated. One can compute several spectral functions within the same run, at the expense of using more memory. The detailed parametrization of the operators is carried out in <SECTION-SPECTRAL_OPERATORS>.

DM-NRG parameters

• **max_state_no:** The upper limit for the number of kept block multiplets.
• **iteration_no:** The maximum number of the NRG iteration steps, N. The program terminates once this value is reached.
• **interval_no:** The energy grid used in each NRG iteration step when spectral functions are calculated. Note that in DM-NRG a very large number of delta peaks is generated in every iteration. Therefore, these delta peaks are stored on a finite mesh. In the present version of the code, the difference between the maximum and minimum of the energy spectrum at each iteration is divided by *interval_no* to obtain the size of the mesh, $d\omega$. Typically $interval\_no \sim$ 1000.
• **degeneracy_threshhold:** If two states are such that the difference between their energies is less than `degeneracy_threshhold` these states are considered to be degenerate.
When discarding states, those lowest in energy are kept while the higher energy states are discarded, such that at the next iteration no more than `max_state_no` are kept. Degenerate

states are block discarded.
• **temperature:** The temperature used for the calculation. When the temperature is fixed to zero, $temperature = 0$, the code runs by using the number of iterations provided by the user. In case of a finite temperature, $temperature > 0$, the variable *iteration_no* is determined and modified during the run to correspond to the temperature specified, using the well-known relationship, $T_n \simeq (1 + \frac{1}{\Lambda})\,\Lambda^{-n/2}$.

4.3.2 <SECTION-FLAGS>

In this section various flags controlling the DM-NRG method are set.
• **dmnrg_flag:** This flag controls whether to perform the backward sweep and use the DM-NRG method or not. Allowed tags are 'ON' or 'OFF'. In case of 'ON', the DM-NRG backward sweep is carried out and the complete set of eigenstates is used to calculate the spectral functions. In case of 'OFF' the usual NRG procedure is performed only.
• **text_swap_files_flag:** This flag controls whether to generate various temporary scratch files as readable text files in addition to the binary files. Allowed tags are 'ON' or 'OFF'.
• **hoppings_on_site_energy_flag:** This flag is important when energy-dependent density of states are considered. This flag controls whether the hoppings and the on-site energies are read from a file or not. If set to 'ON', the hoppings as well as the on-site energies must be computed using the '**he**' utility that comes with the code. This utility generates some data files in the `./dos_mapping` folder, which are next read by the **fnrg** binary. When a flat band is considered, this flag can be set to 'OFF' and then the hoppings are generated on the fly.

4.3.3 <SECTION-SYMMETRIES>

In this section the types of symmetries used can be set. As many symmetries must be specified as were set by the 'symmetry_no' variable. Each line must contain the name of a symmetry, and a minimum and maximum value for the corresponding representation index. Those states for which these bounds are exceeded are simply discarded, and to avoid such uncontrolled truncation, a large value ($\simeq 20$) should be used for most symmetries. The possible symmetry types and the corresponding tags that the present version of the code can handle are as follows: '$U(1)$', '$SU(2)$', '$Z(2)$', '$charge\_SU(2)$'. Other symmetries will also be included in later versions of the code.

4.3.4 <SECTION-BLOCK_STATES>

In this section the quantum numbers of the block states are set. A matrix of dimensions $block\_state\_no \times symmetry\_no$ with integer entries enumerating the representation indices of the block states must be provided. The number of rows has to be equal to the value 'block_state_no' set in <SECTION-PARAMETERS>. Integers in a row u ($u = 1, \ldots, block\_state\_no$) correspond to the representation indices (quantum numbers) of the state u. Thus each row contains $symmetry\_no$ integer entries. Note that it is necessary to provide the double of the usual quantum numbers in case of $SU(2)$ symmetries in order to be able to work with integer numbers.
It is important that the states must be *ordered* by symmetries: The ordering is done with respect to the first symmetry, then with respect to the second symmetry used and so on. While reading the input file, the code checks if the quantum numbers are in ascending order, and if not, then an error message is generated and the run stops.

4.3.5 <SECTION-LOCAL_STATES>

In this section the quantum numbers are set. An integer type matrix of dimensions local_state_no×symmetry_no must be provided. The number of rows must be equal to the value

'local_state_no' set in <SECTION-PARAMETERS>. Otherwise, the rules are the same as for <SECTION-BLOCK_STATES>.

4.3.6 <SECTION-LOCAL_STATES_SIGNS>

In this section the signs of the local states must be set, and a vector of length 'local_state_no', containing ±1-s must be provided. The sign of the local state is **-1 / +1** if it contains an **odd / even** number of fermions.

4.3.7 <SECTION-BLOCK_HAMILTONIAN>

This section contains the definition of the interaction Hamiltonians, $\mathcal{H}_\delta$.

- <**BLOCK_HAMILTONIAN_TERM**> tag denotes the beginning of the definition of a new interaction Hamiltonian term.
- <**BLOCK_HAMILTONIAN_NAME**> is a character string of maximum length char[256] to define the name of the $\mathcal{H}_\delta$ term. It could be any string.
- <**BLOCK_HAMILTONIAN_COUPLING**> a real*8 number, which sets the coupling strength, (G_δ), corresponding to $\mathcal{H}_\delta$. In this line you also specify the *name* (notation) of the coupling. By writing a line "J = 0.736", e.g., you teach the code that this coupling is called J, and its value is 0.736.
- <**BLOCK_HAMILTONIAN_REPRESENTATION_INDEX**>
  the quantum numbers of the Hamiltonian term, an integer vector of length 'symmetry_no'. Under any circumstances the quantum numbers must be 0 for all the symmetries, otherwise the problem is badly posed. The representation index of the Hamiltonian may look therefore redundant at first, but to have a uniform treatment of all operators in the input file, we decided to keep this line.
- <**BLOCK_HAMILTONIAN_MATRIX**> This is a matrix of dimension block_state_no × block_state_no. This matrix contains the matrix elements of the Hamiltonian H_δ. between the states (multiplets) enumerated in <SECTION-BLOCK_STATES>. Some *Mathematica 6.0* notebooks are provided in directory `mathematica_files` which can be used to generate these matrices in a routine way.

The above series of entries must be repeated for *every* local Hamiltonian term, H_δ. The number of these structures must be equal to **coupling_no** defined in <SECTION-PARAMETERS>.

4.3.8 <SECTION-HOPPING_OPERATORS>

This section provides information on the hopping operators. As a rule, we provide information only on creation operators. No information on annihilation operators is needed.

- <**HOPPING_OPERATOR**> tag denotes the beginning of the definition of a new hopping operator.
- <**HOPPING_OPERATOR_NAME**> is a character string with maximum length char[256] to define the name of the corresponding hopping operator. (Could be something like "spin_up_hopping", e.g.)
- <**HOPPING_OPERATOR_REPRESENTATION_INDEX**> the quantum numbers of the hopping operator, an integer vector of dimension 1 × 'symmetry_no' must be given.

- <**HOPPING_OPERATOR_SIGN**> The sign is -1 for a fermionic operator and 1 for a bosonic operator.

- <**HOPPING_OPERATOR_MATRIX**> Contains the reduced matrix elements of the hopping operator ($\sim f_0^\dagger$) between the block states specified in <SECTION-BLOCK_STATES>. It is a block_state_no $\times$ block_state_no matrix. We provided a few *Mathematica* notebooks which you can use to generate these matrix elements in a simple way.

The above series of entries must be repeated as many times as many hopping operators one has. The number of these structures must thus agree with the 'hopping_operator_no' set in <SECTION-PARAMETERS>.

4.3.9 <SECTION-SPECTRAL_OPERATORS>

This section provides information on the spectral operators, i.e., operators, whose spectral functions are computed. As a rule, we provide information only on the 'creation operators', and no information on annihilation operators is given (see Section 2.2 for a precise definition). The following structure must be repeated for every spectral operator. Clearly, the number of structures must be equal to the parameter **spectral_operator_no** set in section <SECTION-PARAMETERS>.

- <**SPECTRAL_OPERATOR**> tag denotes the beginning of the definition of a new spectral operator.

- <**SPECTRAL_OPERATOR_NAME**> is a character string with maximum length char[256] to define the name of the corresponding spectral operator. (E.g., "spin operator")

- <**SPECTRAL_OPERATOR_REPRESENTATION_INDEX**> the quantum numbers of the spectral operator. An integer like vector of dimension 1 $\times$ symmetry_no is expected.

- <**SPECTRAL_OPERATOR_SIGN**> The sign is -1 for a fermionic-like operator and 1 for a bosonic operator.

- <**SPECTRAL_OPERATOR_MATRIX**> A matrix of dimensions block_state_no $\times$ block_state_no is expected, containing the reduced matrix elements of the spectral operator between the block states specified in <SECTION-BLOCK_STATES>. We included in this package a few examples of *Mathematica* notebooks, which can be used to generate these matrices relatively easily.

4.3.10 <SECTION-STATIC_OPERATORS>

This section provides information on the operators for which static averages are evaluated. The following structure must be repeated for every static operator.

- <**STATIC_OPERATOR**> tag denotes the beginning of the definition of a static operator.

- <**STATIC_OPERATOR_NAME**> is a character string with maximum length char[256], defines the name of the static operator being defined.

- <**STATIC_OPERATOR_REPRESENTATION_INDEX**> quantum numbers of the static operator, an integer type vector of dimension 1 $\times$ symmetry_no is expected.

- <**STATIC_OPERATOR_SIGN**> This sign is -1 for a fermionic and 1 for a bosonic operator.
- <**STATIC_OPERATOR_MATRIX**> Contains the reduced matrix elements of the static operator between the block states specified in <SECTION-BLOCK_STATES>. It is a block_state_no × block_state_no matrix. For the calculation of these matrix elements, one can use the mathematica codes provided with the package.

The number of structures shown above must agree with the variable 'static_operator_no' set in <SECTION-PARAMETERS>.

4.3.11 <SECTION-LOCAL_HAMILTONIAN>

- <**LOCAL_HAMILTONIAN_TERM**> tag denotes the beginning of the definition of a local Hamiltonian, i.e. a term that occurs in $\mathcal{H}_i$,

$$\mathcal{H}_i = \sum_{\alpha=1}^{local\_coupling\_no} g_{\alpha,i}\, \mathcal{H}_{i,\alpha} \ ,$$

  with $\mathcal{H}_{i,\alpha} = \mathcal{H}_\alpha(\{\underline{C}_{i,\lambda}\})$. In other words, the local Hamiltonian $\mathcal{H}_{i,\alpha}$ depends on the site index i only through the creation operators acting at site i, otherwise its structure is fixed. A trivial example of such operators are the energies, $\xi_\lambda \leftrightarrow g$, $\underline{C}_{i,\lambda}\underline{C}^\dagger_{i,\lambda} \leftrightarrow \mathcal{H}_{i,\alpha}$. In the present version of the code, we only handle *uniform local couplings*, $g_{\alpha,i} = g_\alpha$, however, later versions shall also be able to treat a user-specified coupling set, $g_{\alpha,i}$. Note that the on-site energy terms are specified under the section <SECTION-LOCAL_HOPPING_OPERATORS>, not here. Here you can define additional local terms in the Hamiltonian, e.g., superconducting pairing terms.
- <**LOCAL_HAMILTONIAN_NAME**> is a character string with maximum length char[256], defines the name of the H_i term.
- <**LOCAL_HAMILTONIAN_COUPLING**> is a real*8 number to set the coupling strength, g_α, corresponding to the local term, $\mathcal{H}_{i,\alpha}$. In the present version of the code the coupling strength does not change along the Wilson chain.
- <**LOCAL_HAMILTONIAN_REPRESENTATION_INDEX**> the quantum numbers of the Hamiltonian term, an integer type vector with dimension 1 × 'symmetry_no'. Similar to the Hamiltonian, these quantum numbers must be all 0, otherwise the Hamiltonian is not invariant under the symmetry assumed.
- <**LOCAL_HAMILTONIAN_MATRIX**> It is a local_state_no × local_state_no matrix, defined on the basis given in <SECTION-LOCAL_STATES>. This contains the (reduced) matrix elements of the coupling Hamiltonians, $\mathcal{H}_\alpha$.

4.3.12 <SECTION-LOCAL_HOPPING_OPERATORS>

Here we specify matrix elements of the local hopping operators between the local states enumerated in <SECTION-LOCAL_STATES>. As a rule, we provide the information only for the creation operators.

- <**LOCAL_HOPPING_OPERATOR**> denotes the beginning of the definition of a hopping operator.
- <**LOCAL_HOPPING_OPERATOR_NAME**> is a character string with maximum length char[256]. Defines the name of the local hopping operator.

- <**LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX**> the quantum numbers of the local hopping operator, an integer type vector with dimension 1 × 'symmetry_no' must be given.
- <**LOCAL_HOPPING_OPERATOR_SIGN**> This sign is -1 for a fermionic and 1 for a bosonic operator.
- <**LOCAL_HOPPING_OPERATOR_MATRIX**> Contains the reduced matrix elements of the static operator between the local states specified in <SECTION-LOCAL_STATES>. It is a local_state_no × local_state_no matrix. For the calculation of these matrix elements, one can use the mathematica codes provided with the package.
- <**LOCAL_ON_SITE_ENERGY_MATRIX**> Contains the reduced matrix elements of the on-site energy term between the local states specified in <SECTION-LOCAL_STATES>. It is a local_state_no × local_state_no matrix. For the calculation of these matrix elements, one can use the mathematica codes provided with the package.

The number of structures shown above must agree with the variable 'hopping_operator_no' set in the <SECTION-PARAMETERS>.

4.3.13 <SECTION-SPECTRAL_FUNCTION>

In this section one specifies the pairs of spectral operators, whose spectral function is computed. This section must contain 'spectral_function_no' rows, each of them corresponding to a given pair of operators.
To compute the spectral function of the Green's function of the form

$$G^R_{A,B^\dagger}(t) = -i\left\langle \left[A(t),\, B^\dagger(0)\right]_\xi \right\rangle \Theta(t), \tag{4.3}$$

one must add a row of the form

$$\{\text{name\_of\_}A^\dagger; \text{name\_of\_}B^\dagger\}\ ,$$

where 'name_of_$A^\dagger$' and 'name_of_$B^\dagger$' are the names of the 'creation operators' set up under <SECTION-SPECTRAL_OPERATORS>.

4.3.14 <SECTION-SPECTRAL_FUNCTION_BROADENING>

Here you specify the parameters of the broadening, used to evaluate the spectral functions. Note that the code **fnrg** does not generate the spectral functions, you need to run the utility **sfb** for that immediately after the run. This part of the input file is thus used by **sfb**. Note that one can improve/change the broadening parameters after running the code too, and then run **sfb** without running **fnrg** again. For details on the broadening, see Section 2.2).
The following parameters must be set in this section:

- **broadening_method:** the method that is used for the broadening of the spectral function along the real axis. Two methods are available: LOG_GAUSS and INTERPOLATIVE_LOG_GAUSS.
- **broadening_parameter:** The value of the broadening parameter (half-width of the log Gauss function in both the log-Gauss and interpolative log-Gauss type of broadening). This parameter is used no matter what method is used.
- **broadening_energy_minim:** The minimum energy for which the broadening is performed ($broadening\_energy\_minim < |\omega| < broadening\_energy\_maxim$). This parameter is used for both methods. This quantity must be positive.

- **broadening_energy_maxim:** The maximum energy for which the broadening is performed ($broadening\_energy\_minim < |\omega| < broadening\_energy\_maxim$). This quantity must be positive.

- **broadening_grid_mesh:** The number of points along the real axis, for which the broadened spectral function is computed. The binary **sfb** generates a logarithmic mesh between [-broadening_energy_maxim, -broadening_energy_minim] and [broadening_energy_minim, broadening_energy_maxim] for which the spectral function is computed.

- **quantum_temperature:** This parameter is used only with the INTERPOLATIVE_LOG_GAUSS method. It sets the scale below which a linear interpolation is used rather than a logarithmic interpolation.

- **spectral_function_nrg_even:** This is a flag that is relevant for NRG runs only. Allowed tags are 'YES' or 'NO': In case of 'YES' the spectral function is computed from the even NRG iterations, and a corresponding data file is generated. If set to 'NO', the code skips this calculation.

- **spectral_function_nrg_odd:** This is a flag that is relevant for NRG runs only. Allowed tags are 'YES' or 'NO': In case of 'YES' the spectral function is also computed from odd NRG iterations. If set to 'NO', the code skips this calculation.

- **spectral_function_dmnrg:** This is a flag parameter that controls whether the spectral function from the full NRG spectrum using the density matrix formalism is computed. Allowed tags are 'YES' or 'NO'. In case of 'YES' the DM-NRG calculation is also performed.

- **static_average_dmnrg:** Flag parameter that controls whether the static average of the form $\langle A^\dagger B\rangle$ is also computed by **sfb** for the two spectral operators, in addition to the calculation of the spectral functions (see the notations in the previous section). Allowed tags are 'YES' or 'NO'.

- **green_function_nrg_even:** This flag controls whether the real part of the Green's function is computed from the even part of the spectral function generated from the nrg data, by performing a Hilbert transform. Allowed tags are 'YES' or 'NO'. The real part of the Green's function can be computed only after the calculation of the spectral function, i.e. if the **spectral_function_nrg_even** flag is set to YES.

- **green_function_nrg_odd:** This flag controls whether the real part of the Green's function is computed from the odd part of the spectral function generated from the nrg data, by performing a Hilbert transform. Allowed tags are 'YES' or 'NO'. The real part of the Green's function can be computed only after the calculation of the spectral function, i.e. if the **spectral_function_nrg_odd** flag is set to YES.

- **green_function_dmnrg:** This flag controls whether the real part of the Green's function is computed from the dmnrg spectral function by performing a Hilbert transform. Allowed tags are 'YES' or 'NO'. The real part of the Green's function can be computed only after the calculation of the spectral function, i.e. if the **spectral_function_dmnrg** flag is set to YES.

- **green_function_grid_mesh:** The number of points on a log-mesh used for doing the Hilbert transform. Once the spectral function for a combination of spectral operators is computed, the real part of corresponding Green's function is generated by performing a Hilbert transformation. To compute the Hilbert transform, a logarithmic mesh is generated. The energy limits for the real/imaginary part of the Green's functions are the same as those for the spectral function , *i.e.* [-broadening_energy_maxim,

-broadening_energy_minim] and [broadening_energy_minim, broadening_energy_maxim]. The energy values for which the Green's function is computed correspond to those for the spectral function. Cubic splines are used for the interpolation of the spectral function. Therefore, for too small green_function_grid_mesh values sometimes oscillations are observed in the energy dependence of the real part of the Green's function. In this case, $green\_function\_grid\_mesh$ must be increased until the oscillations vanish. Increasing this parameter has only a small impact on computing time.

4.4 Outputs of the DM-NRG code

4.4.1 Output directory structure

The output of the code has the following directory structure:

./results/			all results of the code
./results/	automatic_directory_name/		all data for a given run
	(code generated)		
	log_file.dat		
		input.dat	
		spectral_function_dmnrg_*	
		spectral_function_even_*	
		spectral_function_odd_*	
		green_function_dmnrg_*	
		green_function_even_*	
		green_function_odd_*	
		static_average_dmnrg*	
		energy_levels_even*	
		energy_levels_odd*	
		data/	spectral_operators/spectral_operator_*.dat
			static_operators/static_operator_*.dat
			sectors/sectors_*.dat
			transformations/transformation_*.dat
			states/states_*.dat
			delta_peaks/delta_peaks_*.dat
			spectral_functions/spectral_function_*.dat
			mapping/dos.dat
			mapping/hopping_couplings*.dat
			mapping/on_site_energies*.dat
			mapping/density_of_states*.dat
			m_star.dat

Table 4.1: Output directory structure of the code. Directory automatic_directory_ name/ is generated automatically from the parameters in the input file. The "*" in the file names also stands for a list of the parameters (names of the couplings etc.) generated automatically.

All outputs of the code, which are saved on disk, are located in the directory results. In this directory the program generates automatically a directory corresponding to the parameters of the given run. The name of this directory, automatic_directory_name/, contains the values of the following input variables: *model, lambda, max_state_no, iteration_no, temperature, symmetry_no, symmetry_types, coupling_no*, and a file log_file.dat containing the log/error messages during the run.
The automatic_directory_name/, folder has the following structure:
• **input.dat** is a copy of the original input file.
• Various ***.dat** files contain the final results for the **spectral functions**, **static averages**, **green's functions** and the **energy levels**. The names of these files are descriptive, and contain some additional tags such as, e.g., the names of quantum operators for which the spectral functions have been computed.
After running **fnrg**, only a directory **data** containing the raw data is generated. The spectral functions, Green's functions, static averages files are generated only later by running the **sfb** utility. The energy level files are also generated separately, by running the **es** utility.
• Directory **data** contains the data files generated by **fnrg**. Since there are hundreds of files, this directory is divided into subdirectories with descriptive names and containing the following

files:

mapping/
sectors/sectors_*.dat
states/states_*.dat
delta_peaks/delta_peaks_dmnrg_*
delta_peaks/delta_peaks_nrg_*
spectral_operators/spectral_operator_*
static_operators/static_average_dmnrg_*
static_operators/static_average_nrg_even_*
static_operators/static_average_nrg_odd_*
transformations/transformation_*.dat
spectral_functions/spectral_function_*.dat
m_star.dat Here again, *'s stand for operator names and code-generated tags such as the iteration number etc.

4.4.2 Detailed description of the output files in the folder ./results/automatic_directory_name/data

This folder contains the original data generated/used by the code **fnrg** and the utilities **he** and **sfb**. It contains the following files/folders:

• **mapping/**: This folder contains information on the hopping parameters and on site energies, in case a user-defined density of states has been used. This directory is a copy of the directory *dos_mapping*, located in the main directory. It contains the following files:

• **mapping/hopping_couplings*.dat**: Contains the data for the hopping couplings along the chain as computed by the **he** (hopping energies) utility. Each line contains information for one iteration, the first column being the iteration number and subsequent columns representing the values of the hopping amplitudes along the Wilson chain. Each column corresponds to a hopping operator, so the number of columns has to agree with the 'hopping_operator_no'. The tag "self-consistency" in the file name has no relevance here, it was included for later purposes.

• **mapping/on_site_energies*.dat** contains the data for the on-site energies along the chain as computed by the **he** (hopping energies) utility. Each line contains information for each iteration, the first column being the iteration number and the subsequent columns being the values of the on-site energies. Each column corresponds to a hopping operator, so the number of columns has to agree with the 'hopping_operator_no'. The tag "self-consistency"in the file name has no relevance here, it was included for later purposes.

• **sectors/sectors_(iteration_no).dat** The code is organized using so-called sectors. A given sector is characterized by a list of representation labels, specifying the symmetries of the states of that sector. Similarly, operators are also organized using the notion of sectors. The files `sectors_(iteration_no).dat` contain the information on the size and symmetry of sectors in iteration *iteration_no*.
This file has two separate parts. The first part describes the sectors in the new basis at iteration *iteration_no*. The first row of the file contains the total number of the sectors in the new basis (*new_block_sector_no*). The subsequent *new_block_sector_no* lines contain the information for each sector (from 0 to *new_block_sector_no-1*): the representation indices of the sectors, the dimensions of the sectors, the first states of the sectors among the list of all the basis states (multiplets), and the degeneracies of the sectors. Note that the sum of the dimensions multiplied with the corresponding degeneracy values must be equal to the total number of basis states of the new block state basis.

The second part of the files `sectors_(iteration_no).dat` contains the same information on the states kept from the previous iteration (iteration *iteration_no-1*).

• **states/states_(iteration_no).dat**: These files contain the information for the new basis states at iteration *iteration_no*, and the states kept from the previous iteration (iteration *iteration_no*-1). States know about their parent states, and this information is also displayed in these files.
The first row of the file contains the total number of the new basis states (*new_block_state_no*) The next *new_block_state_no* lines contain the information on the multiplets after the diagonalization has been done. Each line contains the number (label) of the sector the state belongs to, the index of the state within this sector, the list of representation indices, the labels of the parent and local states the state was constructed from, the sign of the state (this is 0 for block states), whether the state is kept (K) or eliminated (E) after truncation, and finally the eigen-energy of the state. Note that the sector number cannot be less than zero and it must be smaller than *new_block_sector_no*, specified in file *sectors_(iteration_no).dat*. Also the variable *sector_index* must be non-negative and less than the dimension of the sector given also in file `sectors_(iteration_no).dat`.
The second part of the file `states/states_(iteration_no).dat` gives the same information on the states kept from the previous iteration.

• **transformations/transformation_(iteration_no).dat**: These files contain the information on the matrix O_n that is used to diagonalize the Hamiltonian, and to transform the hopping operators at site *iteration_no*, and the spectral operators (acting at site 0) to the new basis as

$$A_n \rightarrow O_n A_n O_n^\dagger \tag{4.4}$$

These files are only generated if the flag *text_swap_files_flag* in the file `input.dat` is set ON.
The first row of the file contains the total number of the sectors in the new basis in iteration *iteration_no*, and this is followed by three terms for each sector: the sector number, the size of the sector, and finally the real*8 matrix of size sector_size $\times$ sector_size, transforming the given sector.

• **transformations/transformation_binary_(iteration_no).dat** contains the information for the O matrix performing the diagonalization at iteration *iteration_no* in a binary form. These files are only generated if the flag *binary_swap_files_flag* in the file `input.dat` is set to ON.

• **spectral_operators/spectral_operator_(*operator_name*)_(iteration_no).dat** contains the matrix elements of a spectral operator acting at site 0 of the chain (for example $d_0^\dagger$ or S_z) in a sector-decomposed form, after the diagonalization in iteration step *iteration_no* has been done. These operators (which are 'creation operators') are defined and initialized in the `input.dat` file, <SECTION-SPECTRAL_OPERATORS>. These files are only generated if the flag *text_swap_files_flag* in the file `input.dat` is set ON.
The matrix of a spectral operator can be decomposed to sector_no by sector_no small blocks. Depending on the quantum numbers of the operator, several sectors (actually most of them) contain only zero elements. These are considered as non-active sectors. The first line of the file contains the total number of the active sectors (those with proper change of quantum numbers). Then for each active sector the row and column index of the sector is given and is followed by the row and column dimension of the sector, and the matrix elements of the operator in that sector.

• **spectral_operators/spectral_operator_binary_(*operator name*)_(iteration_no).dat**: These files contain the same information as the files **spectral_operator_(*operator name*)_*.dat**,

excepting that the data are in a binary form. These files are only generated if the flag *binary_swap_files_flag* in the file `input.dat` is set to ON.

• **static_operators/static_operator_(*operator name*)_(iteration_no).dat**: They contain the matrix elements of a static operator acting at site 0 of the chain (for example n_0) in a sector-decomposed form, after the diagonalization in iteration step *iteration_no* has been done. These operators (which are 'creation operators') are read from the `input.dat` file, <SECTION-STATIC_OPERATORS>. These files are only generated if the flag *text_swap_files_flag* in the file `input.dat` is set ON.

Static operators must have a block-diagonal structure by symmetry, therefore only these sectors are active. These files are organized otherwise in the same way as the files `spectral_operator_*.dat`.

These files are only generated if the flag *text_swap_files_flag* in the file `input.dat` is set to ON. Furthermore corresponding binary files are also generated if the flag *binary_swap_files_flag* in the file `input.dat` is set to ON.

• **delta_peaks/delta_peaks_dmnrg_(*operator 1*)_(*operator 2*)_(*iteration_no*).dat** contain the raw spectral function peaks for the spectral function of *operator 1* and *operator 2*, generated from iteration *iteration_no*. These files contain two columns: the first one is the energy and the second one the contribution to the amplitude of the spectral function at that energy in iteration *iteration_no*. The calculation is performed at a temperature T specified in the input file, `input.dat`. These output files are used by the utility **sfb** to compute the DM-NRG spectral functions, combining the data from all iterations.

• **delta_peaks/delta_peaks_nrg_(*operator name 1*)_(*operator name 2*)_(*iteration_no*) .dat** contains peaks for the spectral function of *operator 1* and *operator 2*, generated from iteration *iteration_no*. These files contain two columns, the first one being the renormalization energy and the second one the amplitude of the spectral function at the corresponding energy and iteration *iteration_no*. The calculation is performed at a temperature T specified in the input file, `input.dat`. These files are used to compute the NRG spectral function combining data from either even or odd iterations.

• **spectral_functions/spectral_function_dmnrg_(*operator 1*)_(*operator 2*) _(*broadening_par*)_(*iteration*) .dat** contains two columns, the first one being the energy and the second one the real axis value of the spectral function corresponding to the two operators from the filename at the iteration *iteration_no*. Data for negative as well as positive energies are saved into this file. These particular files are relatively important for analyzing the contribution of each iteration to the total spectral function.

• **static_operators/static_average_nrg_even_(operator_name).dat** and **static_average_nrg_odd_(operator_name).dat**: These files contain average of a static operator as a function of temperature, computed by the standard NRG prescription, from the even/odd iterations, respectively.

• **static_operators/static_average_dmnrg_(operator_name).dat** and **static_average_dmnrg_(operator1)_(operator2).dat**: These files contain the expectation values of the static operators and of the pairs of spectral operators specified in the file `input.dat`, computed using the DM-NRG procedure at a temperature T, also specified in the file `input.dat`.

• **m_star.dat**: m_star is the value of the iteration_index at which truncation of the states begins (new_state_no > max_state_no), i.e., for iterations larger then m_star the states, transformation matrices, spectral operators have to be saved. m_star remains fixed for the rest of the NRG steps. This file is set on input and by function `get_m_star()` in `cut_off.cc` and read by various functions when spectral functions are generated later.

4.4.3 Analyzing the data

There are several utilities that we provide to analyze the data you generated.

- First of all, spectral functions and their Hilbert transforms can be generated/analyzed by the utility **sfb**. The broadening parameters must be set in the file `input.dat` sitting in the main folder. You can set there the broadening method, as described in Chapter 4.3. (See also the last section of Chapter 2.) Depending on the parameters set this utility generates the following files.

  spectral_function_dmnrg_(*operator1*)_(*operator2*)_(*broadening*).dat
  These files contain the broadened spectral function of operators *operator1* and *operator2*. Correspondingly, in case the Hilbert transform flag is ON in the file `input.dat`, the real part of the corresponding Green's function is also generated in file **green_function_dmnrg_*.dat**.

  spectral_function_even_(*operator1*)_(*operator2*)_(*broadening*).dat and
  spectral_function_odd_(*operator1*)_(*operator2*)_(*broadening*).dat:
  These files contain the broadened spectral functions of *operator1* and *operator2*, as computed by the traditional NRG method. Data are collected from even and odd iterations respectively. If the Hilbert transform flag is ON in the file `input.dat`, then the real parts of the corresponding Green's functions are also generated and stored in the files **green_function_even_*.dat** and **green_function_odd_*.dat**.

- The energy spectrum (finite size spectrum) can be analyzed by running the utility **es**. While running this utility you can specify quantum number filtering as well as the number of states that you want to keep track of. This is a useful utility to obtain the fixed point spectrum of a model, identify the corresponding conformal field theory, the dimensions of the irrelevant operators, or to extract Fermi liquid parameters. The results are stored separately for even and odd iterations in the files **energy_levels_even*** and **energy_levels_odd***.

Acknowledgments

We are deeply indebted to a number of people for the useful discussions and comments that helped us to create this flexible code. First of all, we would like to thank Laszlo Borda, with whom we created the first NRG code in Hungary many years ago, and whose experience, advices and constant support were crucial to write this flexible code. We benefited a lot from discussions with other NRG experts too, including Mikito Koga, Ralf Bulla, Frithjof Anders, and Walter Hofstetter. We are also indebted to Jan von Delft and Andreas Weichselbaum for many long and constructive discussions. Finally, we would like to thank Laszlo Udvardi for taking care of our computer cluster at the TU Budapest (BUTE), where most of the work has been carried out.

This work was supported by the Hungarian Research Fund (OTKA) under Grant Nos. NF61726, T046303, K68340, and K73361. We are also grateful to the Institute of Mathematics of the BUTE for providing access to their computer cluster (supported by OTKA under grant No. 63066). C.P.M. was partially supported by the Romanian Grant No. CNCSIS 780/2007. I.W. acknowledges support from the Foundation for Polish Science.

Bibliography

[1] A. C. Hewson, *The Kondo Problem to Heavy Fermions* (Cambridge University Press, Cambridge, 1993).

[2] D. L. Cox and A. Zawadowski, Adv. Phys. **47**, 599 (1998).

[3] Robert Peters, Thomas Pruschke and Frithjof B. Anders, Phys. Rev. B **74**, 245114 (2006).

[4] A. Weichselbaum and J. von Delft, Phys. Rev. Lett. **99**, 076402 (2007).

[5] A. I. Toth, C. P. Moca, O. Legeza, G. Zarand, `arXiv:0802.4332`.

[6] H. Saberi, A. Weichselbaum and J. von Delft, `arXiv:0804.0193`.

[7] S. R. White, Phys. Rev. Lett. **69**, 2863 (1992); Phys. Rev. B **48**, 10345 (1993); W. Hofstetter, Phys. Rev. Lett. **85**, 1508 (2000).

[8] Frithjof B. Anders and Avraham Schiller Phys. Rev. Lett. **95**, 196801 (2005).

[9] F. B. Anders, `arXiv:0802.0371`.

[10] U. Schollwöck, Rev. Mod. Phys. **77**, 259 (2005).

[11] K. G. Wilson, Rev. Mod. Phys. **47**, 773 (1975).

[12] Krishna-murthy, H. R., J. W. Wilkins, and K. G. Wilson, Phys. Rev. B **21**, 1003 (1980); Krishna-murthy, H. R., J. W. Wilkins, and K. G. Wilson, Phys. Rev. B **21**, 1044 (1980).

[13] T. A. Costi and A. C. Hewson, Phil. Mag. B **65**, 1165 (1992); T. A. Costi, A. C. Hewson and V. Zlatić, J. Phys.: Condens. Matter **6**, 2519 (1994).

[14] R. Bulla and A. C. Hewson, Z. Phys. B **104**, 333 (1997).

[15] T. Hecht, A. Weichselbaum, J. von Delft and R. Bulla J. Phys.: Condens. Matter **20**, 275213 (2008).

[16] R. Bulla, T. A. Costi and D. Vollhardt, Phys. Rev. B **64**, 045103 (2001).

Appendix A

Input file for the Kondo model with $U_{\mathrm{charge}}(1) \times U_{\mathrm{spin}}(1)$ symmetries

```
# The comments must start with the symbol # .
# Empty lines are ignored.


############################################################
<SECTION-PARAMETERS>
############################################################

# Explanation of variables

# 'model' represents the type of the model that is
# used for the calculations.  It can be any name.

# 'lambda' represents the value for the logarithmic
# discretization parameter.  Usually it is fixed between 2 and 3.

# 'max_state_no' is the number of kept states
# after an iteration.

# 'iteration_no' is the number of iterations performed
# along the NRG run.  Typically it is set to 40 - 70.

# 'symmetry_no' specifies how many symmetries
# are used.  The symmetries are characterized
# in <SECTION-SYMMETRIES>

# 'coupling_no' specifies how many couplings
# are present in the interaction Hamiltonian.

# 'local_coupling_no' is the number of
# Hamiltonians at the sites we add to the Wilson chain

# 'spectral_operator_no' is the number of spectral operators
# acting on site 0, we keep track of.
# These spectral operators are characterized
```

```
# in section <SECTION-SPECTRAL-OPERATORS>

# 'static_operator_no' is the number of static operators
# acting on site 0, we keep track of.
# These static operators are characterized
# in section <SECTION-STATIC-OPERATORS>

# 'hopping_operator_no' is the number of hopping operators.
# These hopping operators are characterized
# in section <SECTION-HOPPING-OPERATORS>

# 'block_state_no' specifies the number
# of initial states/multiplets.  These states are characterized
# by the representation indices and are listed
# in section <SECTION-BLOCK_STATES>


############################################################

############################################################

model = kondo_model # the name of the present model

lambda = 2.0 # discretization parameter

max_state_no = 100 # maximum number of kept multiplets

iteration_no = 20 # allowed number of iterations

symmetry_no = 2 # the number of symmetries

coupling_no = 2 # no of Hamiltonian terms, i.e.  J and B in this case

spectral_operator_no = 3 # no of operators for which correlation functions are computed

static_operator_no = 1 # no of operators for which the static average is computed.

hopping_operator_no = 2 # the number of hopping operators

block_state_no = 8 # the number of initial block states

local_state_no = 4 # the number of local states added in each iteration

local_coupling_no = 1 # the number of Hamiltonian terms on the added site

spectral_function_no = 3 # the number of spectral functions that need to be generated

interval_no = 1000 # mash grid between the maximum and minimum energy at each iteration

degeneracy_threshhold = 1e-6 # energy thresh-hold for discarding states

temperature = 0.0 # temperature used for the run


############################################################
</SECTION-PARAMETERS>
```

```
##########################################################


##########################################################
<SECTION-FLAGS>
##########################################################


# dmnrg_flag controls whether the backward procedure
# has to be done and the full set of eigen-states be
# used for the calculation of the spectral function.
# ON - backwards iteration is done.
# OFF - only the nrg calculation is performed.

# text_swap_files_flag = flag that fixes if the text mode is
# used to save the files.  The files will be saved in binary mode anyway.
# ON - the files are saved in the text mode and kept on the disk.
# OFF= the files are not saved in the text mode.

# binary_swap_files_flag = flag that controls whether the
# unnecessary binary files will be removed or not not after the calculation is done.
# ON - the files will be kept on the disk.
# OFF- the files will be removed from the disk.

# hoppings_on_site_energies_flag = flag that controls
# the reading of the hoppings and the on-site energies.
# ON - the hoppings and the on site energies
# are read from the files in the results/mapping/ folder
# These files are generated by the he utility.
# OFF - the hoppings are computed on the fly
# assuming a flat density of states (DOS = 0.5) on [-1, 1],
# and the on-site energies are set to zero.

dmnrg_flag = ON
text_swap_files_flag = OFF
binary_swap_files_flag = ON
hoppings_on_site_energies_flag = OFF


##########################################################
</SECTION-FLAGS>
##########################################################


############################################################
<SECTION-SYMMETRIES>
############################################################

# As many symmetries must be defined below
# as set by the variable 'symmetry_no'
# in the previous section.

# The possible symmetry types are the following:
# U(1), SU(2), Z(2), charge_SU(2).
#
# Next the limits for the representation index
```

```
# are set.  All states with smaller/larger
# representation indices will be cut off

# For U(1) symmetries the representations are labeled
# by integers (2 S_z) and for SU(2) by non-negative
# integers (2S).

# For the Z(2) symmetry the representations are labeled
# by integers
# 0 - even parity states
# 1 - odd parity states


############################################################


# charge symmetry corresponding to Q
U(1) -10 10

# spin symmetry corresponding to 2 S_z
U(1) -20 20


############################################################
</SECTION-SYMMETRIES>
############################################################


############################################################
<SECTION-BLOCK_STATES>
############################################################
# This section gives the representation indices of the
# block states of the first iteration.
# The number of rows has to be equal
# with the 'block_state_no' in the first section,
# and the number of columns to the 'symmetry_no'.
# Each column contains indices corresponding to a symmetry.
############################################################
 -1  -1
 -1   1
  0  -2
  0   0
  0   0
  0   2
  1  -1
  1   1
############################################################
</SECTION-BLOCK_STATES>
############################################################


############################################################
<SECTION-LOCAL_STATES>
############################################################
# This section gives the representation indices
# of the local states.  The number of rows has to be equal
# with the 'local_state_no' in the first section
```

```
# and the number of columns to the 'symmetry_no'.
# Each column contains indices corresponding to a symmetry.
############################################################
 -1   0
  0   -1
  0    1
  1    0
############################################################
</SECTION-LOCAL_STATES>
############################################################


############################################################
<SECTION-LOCAL_STATES_SIGNS>
############################################################
# These are the signs of the added states.
# The number of signs has to be equal with
# the number of 'local_state_no'.
# All the signs must be enumerated in one line.
# For even electron number, the sign is 1,
# while for odd electron number the sign is -1.
############################################################

1 -1 -1 1

############################################################
</SECTION-LOCAL_STATES_SIGNS>
############################################################


############################################################

<SECTION-BLOCK_HAMILTONIAN>

# The total Hamiltonian will be written as a sum
# of the Hamiltonian terms.
H_0 = \sum_{\delta}^{coupling\_no} G_{\delta} H_{\delta}
# There has to be as many <HAMILTONIAN_TERM> items
# as 'coupling_no' were set in the first section.
# The first tag is the name of the Hamiltonian term,
# the second one gives the representation indices and
# at last comes the matrix elements that will be
# read into the sectors.


# first term in the Hamiltonian

<BLOCK_HAMILTONIAN_TERM>

<BLOCK_HAMILTONIAN_NAME>
H_Kondo
</BLOCK_HAMILTONIAN_NAME>

<BLOCK_HAMILTONIAN_COUPLING>

# J - exchange coupling
J = 0.5
```

</BLOCK_HAMILTONIAN_COUPLING>

<BLOCK_HAMILTONIAN_REPRESENTATION_INDEX>

0 0

</BLOCK_HAMILTONIAN_REPRESENTATION_INDEX>

<BLOCK_HAMILTONIAN_MATRIX>

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0.25	0	0	0	0	0
0	0	0	−0.75	0	0	0	0
0	0	0	0	0.25	0	0	0
0	0	0	0	0	0.25	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

</BLOCK_HAMILTONIAN_MATRIX>

</BLOCK_HAMILTONIAN_TERM>

second term in the Hamiltonian

<BLOCK_HAMILTONIAN_TERM>

<BLOCK_HAMILTONIAN_NAME>
H_Zeeman
</BLOCK_HAMILTONIAN_NAME>

<BLOCK_HAMILTONIAN_COUPLING>

# B - external magnetic field
B = 0.01

</BLOCK_HAMILTONIAN_COUPLING>

<BLOCK_HAMILTONIAN_REPRESENTATION_INDEX>

0 0

</BLOCK_HAMILTONIAN_REPRESENTATION_INDEX>

<BLOCK_HAMILTONIAN_MATRIX>

−0.5	0	0	0	0	0	0	0
0	0.5	0	0	0	0	0	0
0	0	−0.5	0	0	0	0	0
0	0	0	0	0.5	0	0	0
0	0	0	0.5	0	0	0	0
0	0	0	0	0	0.5	0	0
0	0	0	0	0	0	−0.5	0
0	0	0	0	0	0	0	0.5

</BLOCK_HAMILTONIAN_MATRIX>

</BLOCK_HAMILTONIAN_TERM>

##
</SECTION-BLOCK_HAMILTONIAN>
##

##

<SECTION-HOPPING_OPERATORS>

There must be as many <HOPPING_OPERATOR> items
as 'hopping_operator_no' were set in the first section
The matrices must contain the Reduced Matrix Elements.
##

here is the first hopping operator

<HOPPING_OPERATOR>

<HOPPING_OPERATOR_NAME>

f_N_dagger_up

</HOPPING_OPERATOR_NAME>

<HOPPING_OPERATOR_REPRESENTATION_INDEX>

1 1

</HOPPING_OPERATOR_REPRESENTATION_INDEX>

<HOPPING_OPERATOR_SIGN>

The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</HOPPING_OPERATOR_SIGN>

<HOPPING_OPERATOR_MATRIX>

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
−0.707106781186	0	0	0	0	0	0	0
0.707106781186	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0.707106781186	0.707106781186	0	0	0

</HOPPING_OPERATOR_MATRIX>

</HOPPING_OPERATOR>

```
# here is the second hopping operator

<HOPPING_OPERATOR>

<HOPPING_OPERATOR_NAME>

f_N_dagger_down

</HOPPING_OPERATOR_NAME>

<HOPPING_OPERATOR_REPRESENTATION_INDEX>

1 -1

</HOPPING_OPERATOR_REPRESENTATION_INDEX>

<HOPPING_OPERATOR_SIGN>

# The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</HOPPING_OPERATOR_SIGN>

<HOPPING_OPERATOR_MATRIX>
```

$$
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.707106781186 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.707106781186 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.707106781186 & -0.707106781186 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0
\end{array}
$$

```
</HOPPING_OPERATOR_MATRIX>


</HOPPING_OPERATOR>


############################################################
</SECTION-HOPPING_OPERATORS>
############################################################



############################################################
<SECTION-SPECTRAL_OPERATORS>
############################################################
# In this section we set up the matrices for the
# operators for which the spectral function is computed
# Matrices must contain the reduced matrix elements.
############################################################

# here comes the first spectral operator.
```

<SPECTRAL_OPERATOR>

<SPECTRAL_OPERATOR_NAME>

S_z

</SPECTRAL_OPERATOR_NAME>

<SPECTRAL_OPERATOR_REPRESENTATION_INDEX>
0 0
</SPECTRAL_OPERATOR_REPRESENTATION_INDEX>

<SPECTRAL_OPERATOR_SIGN>

The sign is 1 for bosonic operators and -1 for fermionic operators.
1

</SPECTRAL_OPERATOR_SIGN>

<SPECTRAL_OPERATOR_MATRIX>

$$\begin{array}{cccccccc}
-0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -0.5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5
\end{array}$$

</SPECTRAL_OPERATOR_MATRIX>

</SPECTRAL_OPERATOR>

here comes the second spectral operator.
<SPECTRAL_OPERATOR>

<SPECTRAL_OPERATOR_NAME>

f_0_dagger_up

</SPECTRAL_OPERATOR_NAME>

<SPECTRAL_OPERATOR_REPRESENTATION_INDEX>
1 1
</SPECTRAL_OPERATOR_REPRESENTATION_INDEX>

<SPECTRAL_OPERATOR_SIGN>

The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</SPECTRAL_OPERATOR_SIGN>

<SPECTRAL_OPERATOR_MATRIX>

```
 0                  0  0  0                  0                  0  0  0
 0                  0  0  0                  0                  0  0  0
 0                  0  0  0                  0                  0  0  0
 −0.707106781186    0  0  0                  0                  0  0  0
 0.707106781186     0  0  0                  0                  0  0  0
 0                  1  0  0                  0                  0  0  0
 0                  0  1  0                  0                  0  0  0
 0                  0  0  0.707106781186     0.707106781186     0  0  0
</SPECTRAL_OPERATOR_MATRIX>

</SPECTRAL_OPERATOR>

# here comes the third spectral operator.


<SPECTRAL_OPERATOR>

<SPECTRAL_OPERATOR_NAME>

f_0_dagger_down

</SPECTRAL_OPERATOR_NAME>

<SPECTRAL_OPERATOR_REPRESENTATION_INDEX>

1 -1

</SPECTRAL_OPERATOR_REPRESENTATION_INDEX>

<SPECTRAL_OPERATOR_SIGN>

# The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</SPECTRAL_OPERATOR_SIGN>

<SPECTRAL_OPERATOR_MATRIX>

 0  0                0  0                0                  0    0  0
 0  0                0  0                0                  0    0  0
 1  0                0  0                0                  0    0  0
 0  0.707106781186   0  0                0                  0    0  0
 0  0.707106781186   0  0                0                  0    0  0
 0  0                0  0                0                  0    0  0
 0  0                0  0.707106781186   −0.707106781186    0    0  0
 0  0                0  0                0                  −1   0  0
</SPECTRAL_OPERATOR_MATRIX>

</SPECTRAL_OPERATOR>

##########################################################
</SECTION-SPECTRAL_OPERATORS>
##########################################################


##########################################################
<SECTION-STATIC_OPERATORS>
```

```
#########################################################
# In this section we set up the matrices for the
# operators for which the static values are computed
# These operators need to have representation
# indices always zero!
#########################################################

<STATIC_OPERATOR>

<STATIC_OPERATOR_NAME>

S_z

</STATIC_OPERATOR_NAME>

<STATIC_OPERATOR_REPRESENTATION_INDEX>
0 0
</STATIC_OPERATOR_REPRESENTATION_INDEX>

<STATIC_OPERATOR_SIGN>

# The sign is 1 for bosonic operators and -1 for fermionic operators.
1

</STATIC_OPERATOR_SIGN>

<STATIC_OPERATOR_MATRIX>
```

$$
\begin{array}{cccccccc}
-0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -0.5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5
\end{array}
$$

```
</STATIC_OPERATOR_MATRIX>

</STATIC_OPERATOR>

#########################################################
</SECTION-STATIC_OPERATORS>
#########################################################


#########################################################
<SECTION-LOCAL_HOPPING_OPERATORS>
#########################################################

<LOCAL_HOPPING_OPERATOR>

# here comes the first local hopping operator
<LOCAL_HOPPING_OPERATOR_NAME>

f_N+1_dagger_up
```

</LOCAL_HOPPING_OPERATOR_NAME>

<LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX>

1 1

</LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX>

<LOCAL_HOPPING_OPERATOR_SIGN>

The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</LOCAL_HOPPING_OPERATOR_SIGN>

<LOCAL_HOPPING_OPERATOR_MATRIX>

0 0 0 0
0 0 0 0
1 0 0 0
0 1 0 0

</LOCAL_HOPPING_OPERATOR_MATRIX>

<LOCAL_ON_SITE_ENERGY_MATRIX>

0 0 0 0
0 0 0 0
0 0 1 0
0 0 0 1

</LOCAL_ON_SITE_ENERGY_MATRIX>

</LOCAL_HOPPING_OPERATOR>

here comes the second local hopping operator

<LOCAL_HOPPING_OPERATOR>

<LOCAL_HOPPING_OPERATOR_NAME>

f_N+1_dagger_down

</LOCAL_HOPPING_OPERATOR_NAME>

<LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX>

1 -1

</LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX>

<LOCAL_HOPPING_OPERATOR_SIGN>

The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</LOCAL_HOPPING_OPERATOR_SIGN>

<LOCAL_HOPPING_OPERATOR_MATRIX>

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{matrix}$$

</LOCAL_HOPPING_OPERATOR_MATRIX>

<LOCAL_ON_SITE_ENERGY_MATRIX>

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

</LOCAL_ON_SITE_ENERGY_MATRIX>

</LOCAL_HOPPING_OPERATOR>

```
############################################################
</SECTION-LOCAL_HOPPING_OPERATORS>
############################################################

############################################################

<SECTION-LOCAL_HAMILTONIAN>
# The total Local Hamiltonian of the added site
# will be written as a sum of Local Hamiltonian terms
# acting on site n:
```
$H_n = \sum_i g_{i,n} O_{i,n}$
```
# The number of <LOCAL_HAMILTONIAN_TERM> items
# must be equal with the variable 'local_coupling_no'
# set in the first section

# The first tag is the name of the local Hamiltonian term
# the second one gives the representation indices
# last come the matrix elements.

############################################################
# first term in the Local_Hamiltonian
############################################################

<LOCAL_HAMILTONIAN_TERM>

<LOCAL_HAMILTONIAN_NAME>
H_1
</LOCAL_HAMILTONIAN_NAME>

<LOCAL_HAMILTONIAN_COUPLING>

#irrelevant
0.0

</LOCAL_HAMILTONIAN_COUPLING>

<LOCAL_HAMILTONIAN_REPRESENTATION_INDEX>
```

```
0 0

</LOCAL_HAMILTONIAN_REPRESENTATION_INDEX>

<LOCAL_HAMILTONIAN_MATRIX>

 1   0   0   0
 0   0   0   0
 0   0   1   0
 0   0   0   0

</LOCAL_HAMILTONIAN_MATRIX>

</LOCAL_HAMILTONIAN_TERM>

</SECTION-LOCAL_HAMILTONIAN>

############################################################
<SECTION-SPECTRAL_FUNCTION>
############################################################

# In this section we set up the spectral function
# calculation for different types of operators.
# All the time we provide in the input file the
# matrices for the A^dagger and B^dagger operators
# but the spectral function is computed for the
# combination <A, B^dagger>
# spectral function for <f_0_up, f_0_dagger_up>

{f_0_dagger_up; f_0_dagger_up}

# spectral function for <f_0_down, f_0_dagger_down>

{f_0_dagger_down; f_0_dagger_down}

#spectral function for <S_z, S_z>

{S_z; S_z}

############################################################
</SECTION-SPECTRAL_FUNCTION>
############################################################


# The section below is not needed by the fnrg binary.
# It is needed when doing the broadening of the spectral function
# with the utility sfb.


############################################################
<SECTION-SPECTRAL_FUNCTION_BROADENING>
############################################################
# In this section we include broadening parameters that
# are used for computing the broaden spectral function
# The methods that can be used are either LOG_GAUSS or
# INTERPOLATIVE_LOG_GAUSS.
```

```
# These procedures are described in the manual.

# The quantum temperature is relevant only
# for the INTERPOLATIVE_LOG_GAUSS method,
# otherwise only the 'broadening_parameter' parameter
# setting the width of the log-normal
# distribution function is relevant.

# The quantum temperature must be always lower than T
# for finite temperature calculations.
# A good choice is to set the quatum temperature in the range
# 0.5T - T.

# The three parameters spectral_function_*
# specify which spectral functions are being calculated.
# The accepted values are YES or NO only.

# The static_average_dmnrg flags specifies whether the expectation values
# of the static operators are computed.
#The accepted values are YES or NO only.

# The green_function_dmnrg specifies whether the real part of the
# Green's function is computed from the spectral function data.

broadening_method = INTERPOLATIVE_LOG_GAUSS # method used either LOG_GAUSS or INTERPOLATIVE_LOG_GAUSS
broadening_parameter = 0.70 # parameter used for the log gaussian distribution
broadening_energy_minim = 1e-6 # minimum energy for which the broadening is performed
broadening_energy_maxim = 4.0 # maximum energy for which the broadening is performed
broadening_grid_mesh = 100 # grid mesh on a log scale between the minimum and maximum energies
quantum_temperature = 1e-7 # used for the INTERPOLATIVE_LOG_GAUSS method only.

spectral_function_nrg_even = YES # spectral function for the even iterations is computed
spectral_function_nrg_odd = YES # spectral function for the odd iterations part is computed
spectral_function_dmnrg = YES # spectral function for the dmnrg calculation is computed.

static_average_dmnrg = YES # average functions similar with the case of spectral function.

green_function_nrg_even = YES # calculation of the real/imaginary part of the Green's function
from the even part of the nrg spectral function
green_function_nrg_odd = YES # calculation of the real/imaginary part of the Green's function
from the odd part of the nrg spectral function
green_function_dmnrg = NO # calculation of the real/imaginary part of the Green's function
from the spectral function
green_function_grid_mesh = 400 # points for the mesh when doing Hilbert transform

############################################################
</SECTION-SPECTRAL_FUNCTION_BROADENING>
############################################################
```

Appendix B

Input file for the Kondo model with $U_{\mathrm{charge}}(1) \times SU_{\mathrm{spin}}(2)$ symmetries

```
# The comments must start with the symbol # .
# Empty lines are ignored.


############################################################
<SECTION-PARAMETERS>
############################################################

# Explanation of variables

# 'model' represents the type of the model that is
# used for the calculations.  It can be any name.

# 'lambda' represents the value for the logarithmic
# discretization parameter.  Usually it is fixed between 2 and 3.

# 'max_state_no' is the number of kept states
# after an iteration.

# 'iteration_no' is the number of iterations performed
# along the NRG run.  Typically it is set to 40 - 70.

# 'symmetry_no' specifies how many symmetries
# are used.  The symmetries are characterized
# in <SECTION-SYMMETRIES>

# 'coupling_no' specifies how many couplings
# are present in the interaction Hamiltonian.

# 'local_coupling_no' is the number of
# Hamiltonians at the sites we add to the Wilson chain

# 'spectral_operator_no' is the number of spectral operators
# acting on site 0, we keep track of.
# These spectral operators are characterized
```

```
# in section <SECTION-SPECTRAL-OPERATORS>

# 'static_operator_no' is the number of static operators
# acting on site 0, we keep track of.
# These static operators are characterized
# in section <SECTION-STATIC-OPERATORS>

# 'hopping_operator_no' is the number of hopping operators.
# These hopping operators are characterized
# in section <SECTION-HOPPING-OPERATORS>

# 'block_state_no' specifies the number
# of initial states/multiplets.  These states are characterized
# by the representation indices and are listed
# in section <SECTION-BLOCK_STATES>


############################################################

############################################################

model = kondo_model # the name of the present model

lambda = 2.0 # discretization parameter

max_state_no = 100 # maximum number of kept multiplets

iteration_no = 20 # allowed number of iterations

symmetry_no = 2 # the number of symmetries

coupling_no = 1 # no of Hamiltonian terms, i.e.  J in this case

spectral_operator_no = 2 # no of operators for which correlation functions are computed

static_operator_no = 0 # no of operators for which the static average is computed.

hopping_operator_no = 1 # the number of hopping operators

block_state_no = 4 # the number of initial block states

local_state_no = 3 # the number of local states added in each iteration

local_coupling_no = 1 # the number of Hamiltonian terms on the added site

spectral_function_no = 2 # the number of spectral functions that need to be generated

interval_no = 1000 # mash grid between the maximum and minimum energy at each iteration


degeneracy_threshhold = 1e-6 # energy thresh-hold for discarding states

temperature = 0.0 # temperature used for the run


############################################################
```

```
</SECTION-PARAMETERS>
############################################################


############################################################
<SECTION-FLAGS>
############################################################


# dmnrg_flag controls whether the backward procedure
# has to be done and the full set of eigen-states be
# used for the calculation of the spectral function.
# ON - backwards iteration is done.
# OFF - only the nrg calculation is performed.

# text_swap_files_flag = flag that fixes if the text mode is
# used to save the files.  The files will be saved in binary mode anyway.
# ON - the files are saved in the text mode and kept on the disk.
# OFF= the files are not saved in the text mode.

# binary_swap_files_flag = flag that controls whether the
# unnecessary binary files will be removed or not not after the calculation is done.
# ON - the files will be kept on the disk.
# OFF- the files will be removed from the disk.

# hoppings_on_site_energies_flag = flag that controls
# the reading of the hoppings and the on-site energies.
# ON - the hoppings and the on site energies
# are read from the files in the results/mapping/ folder
# These files are generated by the he utility.
# OFF - the hoppings are computed on the fly
# assuming a flat density of states (DOS = 0.5) on [-1, 1],
# and the on-site energies are set to zero.

dmnrg_flag = ON
text_swap_files_flag = OFF
binary_swap_files_flag = ON
hoppings_on_site_energies_flag =OFF


############################################################
</SECTION-FLAGS>
############################################################


############################################################
<SECTION-SYMMETRIES>
############################################################

# As many symmetries must be defined below
# as set by the variable 'symmetry_no'
# in the previous section.

# The possible symmetry types are the following:
# U(1), SU(2), Z(2), charge_SU(2).
#
```

```
# Next the limits for the representation index
# are set.  All states with smaller/larger
# representation indices will be cut off

# For U(1) symmetries the representations are labeled
# by integers (2 S_z) and for SU(2) by non-negative
# integers (2S).

# For the Z(2) symmetry the representations are labeled
# by integers
# 0 - even parity states
# 1 - odd parity states


############################################################


# charge symmetry corresponding to Q
U(1) -10 10

# spin symmetry corresponding to S
SU(2) 0 20


############################################################
</SECTION-SYMMETRIES>
############################################################


############################################################
<SECTION-BLOCK_STATES>
############################################################
# This section gives the representation indices of the
# block states of the first iteration.
# The number of rows has to be equal
# with the 'block_state_no' in the first section,
# and the number of columns to the 'symmetry_no'.
# Each column contains indices corresponding to a symmetry.
############################################################
 -1   1
 0    0
 0    2
 1    1
############################################################
</SECTION-BLOCK_STATES>
############################################################


############################################################
<SECTION-LOCAL_STATES>
############################################################
# This section gives the representation indices
# of the local states.  The number of rows has to be equal
# with the 'local_state_no' in the first section
# and the number of columns to the 'symmetry_no'.
# Each column contains indices corresponding to a symmetry.
############################################################
```

```
 −1   0
 0    1
 1    0
############################################################
</SECTION-LOCAL_STATES>
############################################################


############################################################
<SECTION-LOCAL_STATES_SIGNS>
############################################################
# These are the signs of the added states.
# The number of signs has to be equal with
# the number of 'local_state_no'.
# All the signs must be enumerated in one line.
# For even electron number, the sign is 1,
# while for odd electron number the sign is -1.
############################################################

1 -1 1

############################################################
</SECTION-LOCAL_STATES_SIGNS>
############################################################


############################################################

<SECTION-BLOCK_HAMILTONIAN>

# The total Hamiltonian will be written as a sum
# of the Hamiltonian terms.
H_0 = \sum_{\delta}^{coupling\_no} G_{\delta} H_{\delta}
# There has to be as many <HAMILTONIAN_TERM> items
# as 'coupling_no' were set in the first section.
# The first tag is the name of the Hamiltonian term,
# the second one gives the representation indices and
# at last comes the matrix elements that will be
# read into the sectors.



# first term in the Hamiltonian

<BLOCK_HAMILTONIAN_TERM>

<BLOCK_HAMILTONIAN_NAME>
H_Kondo
</BLOCK_HAMILTONIAN_NAME>

<BLOCK_HAMILTONIAN_COUPLING>

# J - exchange coupling
J = 0.5

</BLOCK_HAMILTONIAN_COUPLING>
```

<BLOCK_HAMILTONIAN_REPRESENTATION_INDEX>

0 0

</BLOCK_HAMILTONIAN_REPRESENTATION_INDEX>

<BLOCK_HAMILTONIAN_MATRIX>

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & -0.75 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

</BLOCK_HAMILTONIAN_MATRIX>

</BLOCK_HAMILTONIAN_TERM>

```
##########################################################
</SECTION-BLOCK_HAMILTONIAN>
##########################################################

##########################################################
```

<SECTION-HOPPING_OPERATORS>

```
# There must be as many <HOPPING_OPERATOR> items
# as 'hopping_operator_no' were set in the first section
# The matrices must contain the Reduced Matrix Elements.
##########################################################

# here is the first hopping operator
```

<HOPPING_OPERATOR>

<HOPPING_OPERATOR_NAME>

f_N_dagger

</HOPPING_OPERATOR_NAME>

<HOPPING_OPERATOR_REPRESENTATION_INDEX>

1 1

</HOPPING_OPERATOR_REPRESENTATION_INDEX>

<HOPPING_OPERATOR_SIGN>

```
# The sign is 1 for bosonic operators and -1 for fermionic operators.
-1
```

</HOPPING_OPERATOR_SIGN>

<HOPPING_OPERATOR_MATRIX>

```
 0  0                0                   0
 1  0                0                   0
 1  0                0                   0
 0  0.707106781186   −1.224744871392     0
</HOPPING_OPERATOR_MATRIX>


</HOPPING_OPERATOR>

############################################################
</SECTION-HOPPING_OPERATORS>
############################################################



############################################################
<SECTION-SPECTRAL_OPERATORS>
############################################################
# In this section we set up the matrices for the
# operators for which the spectral function is computed
# Matrices must contain the reduced matrix elements.
############################################################

# here comes the first spectral operator.


<SPECTRAL_OPERATOR>

<SPECTRAL_OPERATOR_NAME>

S

</SPECTRAL_OPERATOR_NAME>

<SPECTRAL_OPERATOR_REPRESENTATION_INDEX>
0 2
</SPECTRAL_OPERATOR_REPRESENTATION_INDEX>

<SPECTRAL_OPERATOR_SIGN>

# The sign is 1 for bosonic operators and -1 for fermionic operators.
1

</SPECTRAL_OPERATOR_SIGN>

<SPECTRAL_OPERATOR_MATRIX>

 0.866025403784   0     0                   0
 0                0     −0.866025403784     0
 0                0.5   0.707106781186      0
 0                0     0                   0.866025403784
</SPECTRAL_OPERATOR_MATRIX>

</SPECTRAL_OPERATOR>

# here comes the second spectral operator.
<SPECTRAL_OPERATOR>
```

<SPECTRAL_OPERATOR_NAME>

f_0_dagger

</SPECTRAL_OPERATOR_NAME>

<SPECTRAL_OPERATOR_REPRESENTATION_INDEX>
1 1
</SPECTRAL_OPERATOR_REPRESENTATION_INDEX>

<SPECTRAL_OPERATOR_SIGN>

The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</SPECTRAL_OPERATOR_SIGN>

<SPECTRAL_OPERATOR_MATRIX>

0	0	0	0
1	0	0	0
1	0	0	0
0	0.707106781186	−1.224744871392	0

</SPECTRAL_OPERATOR_MATRIX>

</SPECTRAL_OPERATOR>

##
</SECTION-SPECTRAL_OPERATORS>
##

##
<SECTION-LOCAL_HOPPING_OPERATORS>
##

<LOCAL_HOPPING_OPERATOR>

here comes the first local hopping operator
<LOCAL_HOPPING_OPERATOR_NAME>

f_N+1_dagger

</LOCAL_HOPPING_OPERATOR_NAME>

<LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX>

1 1

</LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX>

<LOCAL_HOPPING_OPERATOR_SIGN>

The sign is 1 for bosonic operators and -1 for fermionic operators.

-1

</LOCAL_HOPPING_OPERATOR_SIGN>

<LOCAL_HOPPING_OPERATOR_MATRIX>

$$\begin{matrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1.4142135623731 & 0 \end{matrix}$$

</LOCAL_HOPPING_OPERATOR_MATRIX>

<LOCAL_ON_SITE_ENERGY_MATRIX>

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

</LOCAL_ON_SITE_ENERGY_MATRIX>

</LOCAL_HOPPING_OPERATOR>

##
</SECTION-LOCAL_HOPPING_OPERATORS>
##

##

<SECTION-LOCAL_HAMILTONIAN>

The total Local Hamiltonian of the added site
will be written as a sum of Local Hamiltonian terms
acting on site n:
$H_n = \sum_i g_{i,n} O_{i,n}$
The number of <LOCAL_HAMILTONIAN_TERM> items
must be equal with the variable 'local_coupling_no'
set in the first section

The first tag is the name of the local Hamiltonian term
the second one gives the representation indices
last come the matrix elements.

##
first term in the Local_Hamiltonian
##

<LOCAL_HAMILTONIAN_TERM>

<LOCAL_HAMILTONIAN_NAME>
H_1
</LOCAL_HAMILTONIAN_NAME>

<LOCAL_HAMILTONIAN_COUPLING>

#irrelevant
0.0

```
</LOCAL_HAMILTONIAN_COUPLING>

<LOCAL_HAMILTONIAN_REPRESENTATION_INDEX>

0 0

</LOCAL_HAMILTONIAN_REPRESENTATION_INDEX>

<LOCAL_HAMILTONIAN_MATRIX>

 1   0   0
 0   0   0
 0   0   1

</LOCAL_HAMILTONIAN_MATRIX>

</LOCAL_HAMILTONIAN_TERM>

</SECTION-LOCAL_HAMILTONIAN>

############################################################
<SECTION-SPECTRAL_FUNCTION>
############################################################

# In this section we set up the spectral function
# calculation for different types of operators.
# All the time we provide in the input file the
# matrices for the A^dagger and B^dagger operators
# but the spectral function is computed for the
# combination <A, B^dagger>
# spectral function for <f_0, f_0_dagger>

{f_0_dagger; f_0_dagger }

#spectral function for spin <S, S>

{S; S}

############################################################
</SECTION-SPECTRAL_FUNCTION>
############################################################


# The section below is not needed by the fnrg binary.
# It is needed when doing the broadening of the spectral function
# with the utility sfb.


############################################################
<SECTION-SPECTRAL_FUNCTION_BROADENING>
############################################################
# In this section we include broadening parameters that
# are used for computing the broaden spectral function
# The methods that can be used are either LOG_GAUSS or
# INTERPOLATIVE_LOG_GAUSS.
# These procedures are described in the manual.
```

```
# The quantum temperature is relevant only
# for the INTERPOLATIVE_LOG_GAUSS method,
# otherwise only the 'broadening_parameter' parameter
# setting the width of the log-normal
# distribution function is relevant.

# The quantum temperature must be always lower than T
# for finite temperature calculations.
# A good choice is to set the quatum temperature in the range
# 0.5T - T.

# The three parameters spectral_function_*
# specify which spectral functions are being calculated.
# The accepted values are YES or NO only.

# The static_average_dmnrg flags specifies whether the expectation values
# of the static operators are computed.
# The accepted values are YES or NO only.

# The green_function_dmnrg specifies whether the real part of the
# Green's function is computed from the spectral function data.

broadening_method = INTERPOLATIVE_LOG_GAUSS # method used either LOG_GAUSS or INTERPOLATIVE_LOG_GAUSS
broadening_parameter = 0.70 # parameter used for the log gaussian distribution
broadening_energy_minim = 1e-6 # minimum energy for which the broadening is performed
broadening_energy_maxim = 4.0 # maximum energy for which the broadening is performed
broadening_grid_mesh = 100 # grid mesh on a log scale between the minimum and maximum energies
quantum_temperature = 1e-7 # used for the INTERPOLATIVE_LOG_GAUSS method only.

spectral_function_nrg_even = YES # spectral function for the even iterations is computed
spectral_function_nrg_odd = YES # spectral function for the odd iterations part is computed
spectral_function_dmnrg = YES # spectral function for the dmnrg calculation is computed.

static_average_dmnrg = YES # average functions similar with the case of spectral function.

green_function_nrg_even = YES # calculation of the real/imaginary part of the Green's function
from the even part of the nrg spectral function
green_function_nrg_odd = YES # calculation of the real/imaginary part of the Green's function
from the odd part of the nrg spectral function
green_function_dmnrg = NO # calculation of the real/imaginary part of the Green's function
from the spectral function
green_function_grid_mesh = 400 # points for the mesh when doing Hilbert transform

############################################################
</SECTION-SPECTRAL_FUNCTION_BROADENING>
############################################################
```

Appendix C

Input file for the Kondo model with $SU_{\mathrm{charge}}(2) \times SU_{\mathrm{spin}}(2)$ symmetries

```
# The comments must start with the symbol # .
# Empty lines are ignored.


############################################################
<SECTION-PARAMETERS>
############################################################

# Explanation of variables

# 'model' represents the type of the model that is
# used for the calculations.  It can be any name.

# 'lambda' represents the value for the logarithmic
# discretization parameter.  Usually it is fixed between 2 and 3.

# 'max_state_no' is the number of kept states
# after an iteration.

# 'iteration_no' is the number of iterations performed
# along the NRG run.  Typically it is set to 40 - 70.

# 'symmetry_no' specifies how many symmetries
# are used.  The symmetries are characterized
# in <SECTION-SYMMETRIES>

# 'coupling_no' specifies how many couplings
# are present in the interaction Hamiltonian.

# 'local_coupling_no' is the number of
# Hamiltonians at the sites we add to the Wilson chain

# 'spectral_operator_no' is the number of spectral operators
# acting on site 0, we keep track of.
# These spectral operators are characterized
```

```
# in section <SECTION-SPECTRAL-OPERATORS>

# 'static_operator_no' is the number of static operators
# acting on site 0, we keep track of.
# These static operators are characterized
# in section <SECTION-STATIC-OPERATORS>

# 'hopping_operator_no' is the number of hopping operators.
# These hopping operators are characterized
# in section <SECTION-HOPPING-OPERATORS>

# 'block_state_no' specifies the number
# of initial states/multiplets.  These states are characterized
# by the representation indices and are listed
# in section <SECTION-BLOCK_STATES>


############################################################

############################################################

model = kondo_model # the name of the present model

lambda = 2.0 # discretization parameter

max_state_no = 100 # maximum number of kept multiplets

iteration_no = 20 # allowed number of iterations

symmetry_no = 2 # the number of symmetries

coupling_no = 1 # no of Hamiltonian terms, i.e.  J in this case

spectral_operator_no = 2 # no of operators for which correlation functions are computed

static_operator_no = 0 # no of operators for which the static average is computed.

hopping_operator_no = 1 # the number of hopping operators

block_state_no = 3 # the number of initial block states

local_state_no = 2 # the number of local states added in each iteration

local_coupling_no = 1 # the number of Hamiltonian terms on the added site

spectral_function_no = 2 # the number of spectral functions that need to be generated

interval_no = 1000 # mash grid between the maximum and minimum energy at each iteration

degeneracy_threshhold = 1e-6 # energy thresh-hold for discarding states

temperature = 0.0 # temperature used for the run


############################################################
</SECTION-PARAMETERS>
```

```
##########################################################


##########################################################
<SECTION-FLAGS>
##########################################################


# dmnrg_flag controls whether the backward procedure
# has to be done and the full set of eigen-states be
# used for the calculation of the spectral function.
# ON - backwards iteration is done.
# OFF - only the nrg calculation is performed.

# text_swap_files_flag = flag that fixes if the text mode is
# used to save the files.  The files will be saved in binary mode anyway.
# ON - the files are saved in the text mode and kept on the disk.
# OFF= the files are not saved in the text mode.

# binary_swap_files_flag = flag that controls whether the
# unnecessary binary files will be removed or not not after the calculation is done.
# ON - the files will be kept on the disk.
# OFF- the files will be removed from the disk.

# hoppings_on_site_energies_flag = flag that controls
# the reading of the hoppings and the on-site energies.
# ON - the hoppings and the on site energies
# are read from the files in the results/mapping/ folder
# These files are generated by the he utility.
# OFF - the hoppings are computed on the fly
# assuming a flat density of states (DOS = 0.5) on [-1, 1],
# and the on-site energies are set to zero.

dmnrg_flag = ON
text_swap_files_flag = OFF
binary_swap_files_flag = ON
hoppings_on_site_energies_flag = OFF


##########################################################
</SECTION-FLAGS>
##########################################################


############################################################
<SECTION-SYMMETRIES>
############################################################

# As many symmetries must be defined below
# as set by the variable 'symmetry_no'
# in the previous section.

# The possible symmetry types are the following:
# U(1), SU(2), Z(2), charge_SU(2).
#
# Next the limits for the representation index
```

```
# are set.  All states with smaller/larger
# representation indices will be cut off

# For U(1) symmetries the representations are labeled
# by integers (2 S_z) and for SU(2) by non-negative
# integers (2S).


# For the Z(2) symmetry the representations are labeled
# by integers
# 0 - even parity states
# 1 - odd parity states

############################################################


# charge symmetry corresponding to Q
charge_SU(2) 0 20

# spin symmetry corresponding to S
SU(2) 0 20


############################################################
</SECTION-SYMMETRIES>
############################################################


############################################################
<SECTION-BLOCK_STATES>
############################################################
# This section gives the representation indices of the
# block states of the first iteration.
# The number of rows has to be equal
# with the 'block_state_no' in the first section,
# and the number of columns to the 'symmetry_no'.
# Each column contains indices corresponding to a symmetry.
############################################################
 0  0
 0  2
 1  1
############################################################
</SECTION-BLOCK_STATES>
############################################################


############################################################
<SECTION-LOCAL_STATES>
############################################################
# This section gives the representation indices
# of the local states.  The number of rows has to be equal
# with the 'local_state_no' in the first section
# and the number of columns to the 'symmetry_no'.
# Each column contains indices corresponding to a symmetry.
############################################################
 0  1
 1  0
```

```
##########################################################
</SECTION-LOCAL_STATES>
##########################################################


##########################################################
<SECTION-LOCAL_STATES_SIGNS>
##########################################################
# These are the signs of the added states.
# The number of signs has to be equal with
# the number of 'local_state_no'.
# All the signs must be enumerated in one line.
# For even electron number, the sign is 1,
# while for odd electron number the sign is -1.
##########################################################

-1 1

##########################################################
</SECTION-LOCAL_STATES_SIGNS>
##########################################################


##########################################################

<SECTION-BLOCK_HAMILTONIAN>

# The total Hamiltonian will be written as a sum
# of the Hamiltonian terms.
H_0 = \sum_{\delta}^{coupling\_no} G_{\delta} H_{\delta}
# There has to be as many <HAMILTONIAN_TERM> items
# as 'coupling_no' were set in the first section.
# The first tag is the name of the Hamiltonian term,
# the second one gives the representation indices and
# at last comes the matrix elements that will be
# read into the sectors.




# first term in the Hamiltonian

<BLOCK_HAMILTONIAN_TERM>

<BLOCK_HAMILTONIAN_NAME>
H_Kondo
</BLOCK_HAMILTONIAN_NAME>

<BLOCK_HAMILTONIAN_COUPLING>

# J - exchange coupling
J = 0.5

</BLOCK_HAMILTONIAN_COUPLING>

<BLOCK_HAMILTONIAN_REPRESENTATION_INDEX>
```

```
0 0

</BLOCK_HAMILTONIAN_REPRESENTATION_INDEX>

<BLOCK_HAMILTONIAN_MATRIX>
```

$$\begin{matrix} -0.75 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0 \end{matrix}$$

```
</BLOCK_HAMILTONIAN_MATRIX>

</BLOCK_HAMILTONIAN_TERM>


############################################################
</SECTION-BLOCK_HAMILTONIAN>
############################################################

############################################################



<SECTION-HOPPING_OPERATORS>

# There must be as many <HOPPING_OPERATOR> items
# as 'hopping_operator_no' were set in the first section
# The matrices must contain the Reduced Matrix Elements.
############################################################

# here is the first hopping operator

<HOPPING_OPERATOR>

<HOPPING_OPERATOR_NAME>

f_N_dagger

</HOPPING_OPERATOR_NAME>

<HOPPING_OPERATOR_REPRESENTATION_INDEX>

1 1

</HOPPING_OPERATOR_REPRESENTATION_INDEX>

<HOPPING_OPERATOR_SIGN>

# The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</HOPPING_OPERATOR_SIGN>

<HOPPING_OPERATOR_MATRIX>
```

$$\begin{matrix} 0 & 0 & -1.4142135623731 \\ 0 & 0 & -1.4142135623731 \\ 0.707106781186 & -1.224744871391 & 0 \end{matrix}$$

```
</HOPPING_OPERATOR_MATRIX>


</HOPPING_OPERATOR>

############################################################
</SECTION-HOPPING_OPERATORS>
############################################################



############################################################
<SECTION-SPECTRAL_OPERATORS>
############################################################
# In this section we set up the matrices for the
# operators for which the spectral function is computed
# Matrices must contain the reduced matrix elements.
############################################################

# here comes the first spectral operator.


<SPECTRAL_OPERATOR>

# here comes the spectral operator.
<SPECTRAL_OPERATOR>

<SPECTRAL_OPERATOR_NAME>

f_0_dagger

</SPECTRAL_OPERATOR_NAME>

<SPECTRAL_OPERATOR_REPRESENTATION_INDEX>
1 1
</SPECTRAL_OPERATOR_REPRESENTATION_INDEX>

<SPECTRAL_OPERATOR_SIGN>

# The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</SPECTRAL_OPERATOR_SIGN>

<SPECTRAL_OPERATOR_MATRIX>
```

$$\begin{array}{lll} 0 & 0 & -1.4142135623731 \\ 0 & 0 & -1.4142135623731 \\ .707106781186 & -1.224744871391 & 0 \end{array}$$

```
</SPECTRAL_OPERATOR_MATRIX>

</SPECTRAL_OPERATOR>


# here comes the second spectral operator.


<SPECTRAL_OPERATOR>
```

```
<SPECTRAL_OPERATOR>

<SPECTRAL_OPERATOR_NAME>

S

</SPECTRAL_OPERATOR_NAME>

<SPECTRAL_OPERATOR_REPRESENTATION_INDEX>
0 2
</SPECTRAL_OPERATOR_REPRESENTATION_INDEX>

<SPECTRAL_OPERATOR_SIGN>

# The sign is 1 for bosonic operators and -1 for fermionic operators.
1

</SPECTRAL_OPERATOR_SIGN>

<SPECTRAL_OPERATOR_MATRIX>
```

0	−0.866025403784	0
0.5	0.707106781186	0
0	0	0.866025403784

```
</SPECTRAL_OPERATOR_MATRIX>

</SPECTRAL_OPERATOR>


############################################################
</SECTION-SPECTRAL_OPERATORS>
############################################################


############################################################
<SECTION-LOCAL_HOPPING_OPERATORS>
############################################################

<LOCAL_HOPPING_OPERATOR>

# here comes the first local hopping operator
<LOCAL_HOPPING_OPERATOR_NAME>

f_N+1_dagger

</LOCAL_HOPPING_OPERATOR_NAME>

<LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX>

1 1

</LOCAL_HOPPING_OPERATOR_REPRESENTATION_INDEX>

<LOCAL_HOPPING_OPERATOR_SIGN>
```

```
# The sign is 1 for bosonic operators and -1 for fermionic operators.
-1

</LOCAL_HOPPING_OPERATOR_SIGN>

<LOCAL_HOPPING_OPERATOR_MATRIX>
```

$$\begin{matrix} 0 & 1.4142135623731 \\ -1.4142135623731 & 0 \end{matrix}$$

```
</LOCAL_HOPPING_OPERATOR_MATRIX>

<LOCAL_ON_SITE_ENERGY_MATRIX>
```

$$\begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}$$

```
</LOCAL_ON_SITE_ENERGY_MATRIX>

</LOCAL_HOPPING_OPERATOR>



############################################################
</SECTION-LOCAL_HOPPING_OPERATORS>
############################################################

############################################################

<SECTION-LOCAL_HAMILTONIAN>
# The total Local Hamiltonian of the added site
# will be written as a sum of Local Hamiltonian terms
# acting on site n:
```

$H_n = \sum_i g_{i,n} O_{i,n}$

```
# The number of <LOCAL_HAMILTONIAN_TERM> items
# must be equal with the variable 'local_coupling_no'
# set in the first section

# The first tag is the name of the local Hamiltonian term
# the second one gives the representation indices
# last come the matrix elements.

############################################################
# first term in the Local_Hamiltonian
############################################################

<LOCAL_HAMILTONIAN_TERM>

<LOCAL_HAMILTONIAN_NAME>
H_1
</LOCAL_HAMILTONIAN_NAME>

<LOCAL_HAMILTONIAN_COUPLING>

#irrelevant
0.0

</LOCAL_HAMILTONIAN_COUPLING>
```

```
<LOCAL_HAMILTONIAN_REPRESENTATION_INDEX>

0 0

</LOCAL_HAMILTONIAN_REPRESENTATION_INDEX>

<LOCAL_HAMILTONIAN_MATRIX>

 1   0
 0   0

</LOCAL_HAMILTONIAN_MATRIX>

</LOCAL_HAMILTONIAN_TERM>

</SECTION-LOCAL_HAMILTONIAN>

############################################################
<SECTION-SPECTRAL_FUNCTION>
############################################################

# In this section we set up the spectral function
# calculation for different types of operators.
# All the time we provide in the input file the
# matrices for the A^dagger and B^dagger operators
# but the spectral function is computed for the
# combination <A, B^dagger>
# spectral function for <f_0, f_0_dagger>

{f_0_dagger; f_0_dagger }

# spectral function for spin <S, S>

{S; S }

############################################################
</SECTION-SPECTRAL_FUNCTION>
############################################################


# The section below is not needed by the fnrg binary.
# It is needed when doing the broadening of the spectral function
# with the utility sfb.


############################################################
<SECTION-SPECTRAL_FUNCTION_BROADENING>
############################################################
# In this section we include broadening parameters that
# are used for computing the broaden spectral function
# The methods that can be used are either LOG_GAUSS or
# INTERPOLATIVE_LOG_GAUSS.
# These procedures are described in the manual.

# The quantum temperature is relevant only
# for the INTERPOLATIVE_LOG_GAUSS method,
```

```
# otherwise only the 'broadening_parameter' parameter
# setting the width of the log-normal
# distribution function is relevant.

# The quantum temperature must be always lower than T
# for finite temperature calculations.
# A good choice is to set the quatum temperature in the range
# 0.5T - T.

# The three parameters spectral_function_*
# specify which spectral functions are being calculated.
# The accepted values are YES or NO only.

# The static_average_dmnrg flags specifies whether the expectation values
# of the static operators are computed.
# The accepted values are YES or NO only.

# The green_function_dmnrg specifies whether the real part of the
# Green's function is computed from the spectral function data.

broadening_method = INTERPOLATIVE_LOG_GAUSS # method used either LOG_GAUSS or INTERPOLATIVE_LOG_GAUSS
broadening_parameter = 0.70 # parameter used for the log gaussian distribution
broadening_energy_minim = 1e-6 # minimum energy for which the broadening is performed
broadening_energy_maxim = 4.0 # maximum energy for which the broadening is performed
broadening_grid_mesh = 100 # grid mesh on a log scale between the minimum and maximum energies
quantum_temperature = 1e-7 # used for the INTERPOLATIVE_LOG_GAUSS method only.

spectral_function_nrg_even = YES # spectral function for the even iterations is computed
spectral_function_nrg_odd = YES # spectral function for the odd iterations part is computed
spectral_function_dmnrg = YES # spectral function for the dmnrg calculation is computed.

static_average_dmnrg = YES # average functions similar with the case of spectral function.

green_function_nrg_even = YES # calculation of the real/imaginary part of the Green's function
from the even part of the nrg spectral function
green_function_nrg_odd = YES # calculation of the real/imaginary part of the Green's function
from the odd part of the nrg spectral function
green_function_dmnrg = NO # calculation of the real/imaginary part of the Green's function
from the spectral function
green_function_grid_mesh = 400 # points for the mesh when doing Hilbert transform

############################################################
</SECTION-SPECTRAL_FUNCTION_BROADENING>
############################################################
```

Appendix D

License agreements

GNU LESSER GENERAL PUBLIC LICENSE Version 3, 29 June 2007
Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.
This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.
0. Additional Definitions.
As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.
"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.
An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.
A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".
The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.
The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.
1. Exception to Section 3 of the GNU GPL.
You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.
2. Conveying Modified Versions.
If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:
a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.
3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:
a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
b) Accompany the object code with a copy of the GNU GPL and this license document.
4. Combined Works.
You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:
a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
d) Do one of the following:
0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)
5. Combined Libraries.
You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:
a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.
6. Revised Versions of the GNU Lesser General Public License.
The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published

version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.
If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.