NRG Ljubljana - reference manual

Rok Žitko

May 31, 2013

This is the reference manual for the "NRG Ljubljana" package for performing numerical renormalization group (NRG) calculations in quantum impurity physics. The manual provides the documentation for the input files (`param` and `data`) and the structure of the output files (`td`, `custom`, `customfdm`, spectral functions). There is also brief discussion about extending the capabilities of the program.

This manual describes version 2.3.20 of NRG Ljubljana.

Contents

1 Numerical renormalization group

There are several well-written reference papers which thoroughly describe the NRG method: original Wilson's RMP paper [1], the very detailed papers on the single-impurity Anderson model [2, 3], and a recent review [4]. Further works discuss dynamical quantities [5, 6, 7, 8, 9, 10], transport calculations [11, 12, 13, 14, 15, 16], scattering properties [17, 18]. In many cases, density-matrix extension of the NRG is required [19], which may also be implemented for non-Abelian symmetries [20, 21, 22]. The self-energy may be computed as a ratio of two correlators [23]. There are several discretization schemes [24, 25, 26, 27, 28, 29, 30], some suitable for superconducting cases [31, 32, 33]. There are several types of systematic errors in NRG: discretization and truncation errors [29, 30, 34], mass-flow errors [35] (mainly affecting bosonic models).

2 Installation

The installation of the code is described in the `README` file which comes with the distribution. In most cases, it should be sufficient to run the `configure` script and then `make`, followed by `make install`.

3 Invoking the code

There are two parts to each NRG calculation:

1. Initialization of the problem using the command `nrginit`, which starts Mathematica and runs the code in the `initial.m` script; this script reads the parameter file `param` and generates file `data` which contains the full information necessary to start a NRG iteration (eigenvalues, matrix representations of all necessary operators, Wilson chain coefficients). The second-quantization operator algebra library SNEG is used for this purpose [36].

2. Actual NRG iteration which is started using the command `nrgrun`, which reads both `param` and `data` and performs the calculation.

In most cases the first part is fast (a few seconds). It must be run on a computer where Mathematica is available. On clusters, that might be the cluster head node.

The second part may be time demanding, depending on the complexity of the problem. Simple one-channel calculation can be performed in a few seconds, while low-symmetry multi-channel calculations may run for days or weeks.

4 Input file with parameters, `param`

The parameters are documented in the same order as the corresponding definitions appear in the `param.cc` source file. Note that `param.cc` can help as an additional source of information in addition to this manual, since each parameter is documented and commented. In case of uncertainty, the code is relatively well commented; the main code is in `nrg.cc`. At the end of the list, there are also parameters which are only used in the Mathematica part of the NRG Ljubljana code and thus do not appear in `param.cc`.

Some parameters take strings (and are case-sensitive, unless specified otherwise), some take floating point numbers (C-like exponential notation is allowed) or integer numbers, and some are boolean (these take either `true` or `false`).

4.1 `Nmax`, the number of iteration steps performed

`Nmax` controls the number of the NRG steps performed with the C++ part of the code is invoked using `nrgrun`, i.e., `Nmax` controls the lowest energy/temperature scale considered. The name of the parameter has its origin in the variable `N`, which indexes the NRG steps, and during the calculation ranges (typically) from 0 to `Nmax`-1 (including). `Nmax` is thus also associated with the length of the Wilson chain and the

number of the discretization parameters that need to be computed. The Wilson chain sites are numbered from 0 to `Nmax` (included). The number of NRG steps is one less than the Wilson chain length, since the first ("zero-th") site is taken into account in the initialization of the calculation performed using `nrginit`.

The default value of `Nmax` is 50.

See also the following parameter `Tmin`.

4.2 `Tmin`, the lowest effective temperature achieved

If the parameter `Tmin` is defined (and non-zero), the value of `Nmax` in the parameter file is not used, but rather computed from `Tmin`: Nmax is defined so that the effective temperature T_N at the final step is equal or higher than the minimal temperature `Tmin` (but would be lower if one further step were performed).

The idea behind `Tmin` is that it is often interesting to modify the discretization parameter Λ and perform calculations for Wilson chain lengths that correspond to (approximately) the same minimal effective temperature scale. This is easily accomplished by setting `Tmin` to a suitable value, eliminating the need to recompute `Nmax` for each choice of Λ.

The default value of `Tmin` is negative, thus the value of `Nmax` specified in the pameter file is used (or its default value).

See also the previous parameter `Nmin`.

4.3 `Ninit`, the highest index of initial Wilson chain

Usually in NRG, one takes just one (zero-th, f_0) site of the Wilson chain in consideration when initializing the calculation. The parameter `Ninit` controls the highest index of the site taken explicitly into account. The default is the conventional choice of `Ninit`=0.

During the NRG calculation in the C++ part of the code, N thus ranges from `Ninit` to `Nmax`-1. (See above.)

Setting `Ninit` to larger value is useful in two cases:

- For calculating the properties of the system inside the Wilson chain. One can, for example, compute the occupancies for a number of the Wilson chain sites in order to estimate the displaced charge upon coupling the impurity to the conduction band. One can also compute spectral functions inside the Wilson chain. `Ninit` should be chosen so that all the required operators are contained in the initial cluster.

- For testing purposes. This enforces an exact calculation of the eigenstates and the operator matrices for a longer Wilson chain. These can be compared with the results at later NRG steps in a calculation with smaller `Ninit`.

4.4 `Lambda`, the logarithmic discretization parameter

`Lambda` is the NRG logarithmic discretization parameter Λ. After each NRG step, the effective temperature scale is reduced by $\sqrt{\Lambda}$.

We do not discuss Λ further here, since it is addressed in the general NRG literature.

The default value is 1. This is not a value that is actually allowed, thus it has to be explicitly set in the parameter file. Commonly used values are around 2 for single-channel problems, and 3 or 4 for two-channel problems. Depending on the quantities being computed and the nature of the problem (in particular one has to be away from quantum phase transitions), it is sometimes possible to use very large values of Λ (of order 10) and obtain meaningful results after the z-averaing. For such calculations it is strongly recommended to use `discretization=z`.

4.5 `strategy`, which matrix elements need to be recomputed

The parameter `strategy` can take two different values, either `all` or `kept`. This parameter affects what parts of the operator matrices are recomputed from one step to the next. For `strategy=all`, the full matrices are recomputed, for both states that will be kept and those that will be discraded. For

strategy=kept, only those matrix elements are computed which are required in the next NRG step, i.e., those that correspond to the kept states. Since recomputations are somewhat numerically expensive, the calculation proceeds faster for strategy=kept.

When full-Fock-space calculations are performed (CFS or FDM), we need to know all the matrix elements since the spectral functions and expectation values are computed using the matrix elements linking discarded and kept states. When cfs=true, fdm=true or mats=true, the setting all is automatically used, irrespective of what is specified in the parameter file.

Note: this setting also affects the calculation of expectation values of singlet operators, as well as of dynamic quantities using the conventional NRG algorithm (for finite=true) and the DM-NRG algorithm (for dmnrg=true). Nevertheless, the differences are small. In fact, the gain in efficiency is sometimes sufficient to make it possible to increase the number of states kept, thereby compensating further the differences between strategy=all and strategy=kept.

The default value of strategy is kept.

4.6 lastall, which matrix elements are computed in the last iteration

(seldom used)

The parameter lastall is related to the previous one, strategy. It controls the special behavior at the last NRG step. If lastall=true, all elements are kept in the last step even if strategy=kept. This is useful in DM-NRG calculations, where one might want to compute the density matrix from all states in the last step, but only the kept states in other steps (for reasons of performance). The penalty in the accuracy of the results is usually very small.

When full-Fock-space calculations are performed (CFS or FDM), lastall is automatically set to true, unless the parameter lastalloverride is specified (see below).

The default value of lastall is false.

4.7 lastalloverride

(seldom used)

The parameter lastalloverride allows to override the automatic setting of lastall to true for CFS and FDM calculations. This comes handy in multi-channel calculations where keeping all states in the construction of the density matrix would require large ammounts of memory. By setting lastall=false and lastalloverride=true, only the subset of "kept" states is used. This is an approximation, but if used carefully, the error is controlled.

The default value of lastalloverride is false.

4.8 z, twist-parameter for interleaved discretization grids

The parameter z is the z which appears in the discretization formulas when the twist averaging is used. Conventionally, this is a value between 0 and 1. The twist-averaging consists in performing different NRG runs for a range of values of z, then averaging the results to obtain the final, improved answer.

The default value of z is 1, which corresponds to a non-shifted grid as used originially by K. Wilson.

4.9 discretization, choice of the discretization scheme

There are several schemes for logarithmically discretizing the continuum of the bath degrees of freedom:

- Y - the scheme proposed in the paper by Yoshida, Whitaker, Oliveira, Phys. Rev. B 41, 9403 (1990).
- C - the scheme proposed in the paper by Campo, Oliveira, Phys. Rev. B 72, 104432 (2005). It corrects the systematic underestimation of the bath density of states of scheme Y.
- Z - the scheme proposed in the paper by Zitko, Pruschke, Phys. Rev. B 79, 085106 (2009). It corrects the systematic error in the first energy interval of scheme C.

The default value of `discretization` is `Y`. Note that for `discretization=Y` and `z=1` (i.e. for the default values of both parameters) the original Wilson's discretization is performed.

This parameter also affects how the energy scale at the N-th NRG step is computed. For `Y`, the formula is

$$\omega_N = \frac{1+\Lambda^{-1}}{2}\Lambda^{-(N-1)/2+1-z}, \tag{1}$$

while for `C` and `Z` it is

$$\omega_N = \frac{1-\Lambda^{-1}}{\log\Lambda}\Lambda^{-(N-1)/2+1-z}. \tag{2}$$

Generally speaking, `C` is better than `Y`, while `Z` is better than both others (due to less systematic errors). There are some cases, however, where it is recommended to use `Y`, because it has less scatter as a function of z. This is advantageous in vicinity of quantum phase transitions, where the scatter could imply that for different values of z the calculation ends up in different low-T fixed points. Despite larger scatter, away from phase transitions the z-average of the results for `discretization=Z` lie closed to the true results expected in the $\Lambda \to 1$ continuum limit.

4.10 `band`, density of states in the continuum

The parameter `band` is used to choose the density of states (DOS) in the continuum. Two commonly used types are pre-programmed: `flat` for flat DOS:

$$\rho(\omega) = \frac{1}{2D}, \quad \text{for} \quad -D \geq \omega \geq D, \tag{3}$$

and `cosine` for one-dimensional semi-infinite tight-binding chain with cosine dispersion $\epsilon_k = -D\cos k$, which is equivalent to the semicircular local DOS on the first site of the chain:

$$\rho(\omega) = \frac{2}{\pi D}\sqrt{1-\left(\frac{\omega}{D}\right)^2}, \quad \text{for} \quad -D \geq \omega \geq D. \tag{4}$$

In NRG, one usually chooses the half-bandwidth D as the energy unit, $D = 1$.

For using NRG Ljubljana as an impurity solver in the DMFT, one has to set `band=dmft`. This can also be used in other cases where the DOS is defined as a polynomial interpolation based on tabulated data points.

If `band=manual`, the discretization is handled manually outside the NRG initialization code `initial.m`.

For `discretization=Z`, additional possibilities for `band` are:

- `any` - for arbitrary particle-hole symmetric density of states with coefficients loaded from an externally generated file named `FSOL.dat` (see also `dos`, `xmax`, `tabstep`, and `solpath`);
- `asymode` - for arbitrary asymmetric density of states with coefficients loaded from externally generated files named `FSOL.dat` and `FSOLNEG.dat` (see also `dos`, `xmax`, and `solpath`);
- `adapt` - for arbitrary asymmetric density of states and adaptable discretization mesh. The coefficients are loaded from externally generated files named `FSOL.dat`, `GSOL.dat`, `FSOLNEG.dat`, and `GSOLNEG.dat` (see also `dos`, `xmax`, and `solpath`).

Finally, it is also possible to set `band`=filename. In this case, the named module is loaded and executed. The default value of `band` is `flat`.

4.11 `betabar`, the $\bar{\beta}$ parameter for thermodynamics

The parameter `betabar` corresponds to the constant $\bar{\beta}$ as defined by K. Wilson in the early NRG papers. It is used in the calculation of the expectation values of operators as a prefactor in the Boltzmann weights:

$$e^{-\bar{\beta}E_n}, \tag{5}$$

where E_n is the energy of the eigenstate in the characteristic units of the current NRG iteration (ω_N). Both $\bar{\beta}$ and E_n are dimensionless quantities.

In effect, $\bar{\beta}$ determines the effective temperature for computing thermodynamic quantities at iteration N. It is also used for computing the expectation values of singlet operators which are reported in the output file `custom`. Ideally, the results would not depend on the value of $\bar{\beta}$, but they do to some extent.

The commonly used value of 0.46 requires high truncation cutoff energy to obtain converged result, which is sometimes unpractical for multi-channel problems and other memory-constrained problems. The default value is 1, which is somewhat large. The difference in extracted T_K is less than 1% when comparing the results obtained with $\bar{\beta} = 1$ and $\bar{\beta} = 0.75$ with $\Lambda = 2$ and `keepenergy=10`.

See Ref. **??** for a different way of calculating thermodynamic properties, which is based on the FDM algorithm. In this approach there is no tunable parameter such as $\bar{\beta}$.

4.12 `T`, the temperature

The temperature `T` controls the physical temperature which appears as the parameter in the Boltzmann weights for computing the finite-T spectral functions, as well as the finite-T expectation values using the FDM-NRG algorithm (`customdfm` output file). It is expressed in the units of the bandwidth D.

Note that this parameter does not affect the Wilson chain length used which is independently controlled by `Nmax` or `Tmin`, nor the effective temperatures used in the calculation of thermodynamic properties (`td` output file) or the temperature-depedent expectation values (`custom` output file) which are controlled by `betabar`.

The default is to set the value of `T` based on `T_ratio`. In turn, the default for `T_ratio` is such that `T` corresponds to the lowest energy scale of the Wilson chain (see below).

4.13 `T_ratio`, set the temperature according to the Wilson chain length

The parameter `T_ratio` allows to fix the physical temperature `T` based on the Wilson chain length as controlled by `Nmax` and `Tmin`. More accurately, `T` is equal to the energy scale of the last NRG step, multiplied by `T_ratio`.

The default value of `T_ratio` is 1.

4.14 `Tmin_ratio`, chain-length control via `T`

If the physical temperature is set by explicitly giving a value to the parameter `T`, and if `Tmin_ratio` is set, the value of `Tmin` is determined according to `Tmin=Tmin_ratio` $\times$`T` and this, in turn, sets the length of the Wilson chain `Nmax` to a suitable value. The purpose of this parameter is to use the physical temperature `T` to control the length of the Wilson chain. Typically `Tmin_ratio` would be some value sufficiently smaller than 1, such as 0.1, so that the energy scale of the last site on the NRG chain is somewhat lower than the physical temperature `T`. A suitable choice of `Tmin_ratio`, when used, is especially important when calculating spectral functions at finite temperatures using `finite`, `dmnrg` and `cfs`.

The default behavior is that `Tmin_ratio` is not used.

4.15 `keep`, the maximum number of states kept in the truncation

The parameters `keep`, `keepmin`, and `keepenergy` are used to control the truncation of states during the NRG iteration. The first parameter, `keep`, sets the absolute upper limit to the number of states kept. When only `keep` is set (but not `keepmin`), exactly `keep` states are retained. Note that in NRG Ljubljana, the number of states is always interpreted in the sense of the number of multiplets, i.e., the total number of eigenvalues in all of the invariant subspaces without taking into account the multiplicity. To obtain the total number of states in the sense of individual states, one should multiply for each invariant subspace the number of multiplets kept in that subspace by the degeneracy of the subspace (i.e., the multiplicity).

The default value of `keep` is 100.

4.16 `keepenergy`, the cut-off energy for truncation

As an alternative to keeping a fixed number of states, it is possible to truncate at a chosen energy E_{cutoff} (in units of the characteristic energy scale ω_N), set by `keepenergy`. If `keepenergy` is set, then the meaning of `keep` is that of the maximum number of states kept (even if the criterium sey by `keepenergy` is *not* fulfilled). Usually, one set `keepenergy` to some value that guarantees the convergence with respect to the truncation cutoff (10 is a good conservative choice for most types of calculations) and sets `keep` either to some excessively large value (so as not to interfere with the energy cut-off truncation) or to a value which corresponds to the maximal number of states that one can work with within the constraints set by the memory size of the computer (to prevent excessive swapping).

The actual energy of the highest eigenvalue retain is reported in the `log` file as the value of `Emax`. In the same lines, there is also the number of states/multiplets kept, `nrkept`, and the number of states (total number of states taking into account the multiplicities), `nrkeptmult`. In addition, the code reports the percentages of the states kept, for instance `Keep:  500 out of 1869, ratio=0.268`.

By default the truncation is not based on the cut-off energy.

4.17 `keepmin`, the minimal number of states kept

When the truncation uses the cut-off energy, it is possible to specify the minimum number of the states kept by using `keepmin`, even if that would imply keeping states up to an eigenvalue which is much larger than `keepenergy`.

The default value of `keepmin` is 0, i.e., there is no lower bound on the number of states kept.

4.18 `keepblock`

(obsolete, to be removed)

4.19 `keepzrescale`

(seldom used)

4.20 `log`, what information to show in the log file

The parameter `log` defines the verbosity level in the log file. It allows fine-grained control over what is reported. The value of the parameter is a string of letters/numbers/symbols (in arbitrary order) which turn on reporting of various details:

- `i`, report the number of states in each subspace when building the Hamiltonian matrices in different invariant subspaces;
- `s`, when building the Hamiltonian matrices, show the invariant subspaces (and their sizes) which contribute to the matrix in the invariant subspace currently being processed, related to `i`, but more detailed;
- `m`, dump each Hamiltonian matrix after it has been constructed (very verbose);
- `H`, report details about storing the temporary data (density matrices, unitary transformation matrices) on the disk;
- `f`, show details about the recalculation of the matrix elements of the Wilson chain operators, very detailed (for debugging only); even more details can be obtained using `F`;
- `0`, show which operator matrices are being recomputed, more details are provided using `r`, even more with `R`;
- `g`, report which spectral functions are being computed, their type, and the invariant subspaces which contribute (very verbose output);

- `c`, report which spectral functions will be computed and the corresponding filenames;
- `A`, report which eigensolver routing is being used, the matrix size, and the node rank (when MPI parallelization is used);
- `t`, report the time used for the diagonalization;
- `e`, report the first (lowest) `logenumber` eigenvalues for each matrix diagonalized;
- `%`, report the lowest and the highest eigenvalue for each matrix diagonalized;
- `d`, report low-level details about the diagonalization (currently just the temporary workspace sizes);
- `T`, report spectral weight trimming information (see the parameters `discard_immediately` and `discard_trim`);
- , report the details about merging partial spectral function information (on-shell weight, current accumulated weights);
- `w`, report the details about the calculation of density matrices and of the weights w_N that are necessary for the FDM NRG;
- `X`, show the results of the eigenvalue cluster detection algorithm (which is controled by the parameter `fixeps`);
- $, (obsolete);
- `M`, debug the parallelization using MPI (matrix transmission, scheduling of jobs to slave nodes, messages sent);

Some useful combinations are `fr` for debugging the recalculation of the matrix elements, and `ies` for debugging the matrix construction and diagonalization. These are useful for code development. In standard use, one can use `At` for obtaining information about the diagonalization codes, which is particularly important for monitoring big long calculations. If disk usage is an issue, `H` may come handy.

Note that the settings are case-sensitive.

The default value of `log` is an empty string.

4.21 `logall`, set verbosity to maximum

Setting the parameter `logall` to `true` is equivalent to specifying all log letters in the parameter `log`. The default value is `false`.

4.22 `logenumber`, how many eigenvalues to report in log

The parameter `logenumber` is related to the log setting `e`, which enables reporting the eigenvalues from the diagonalization routines. For each Hamiltonian matrix, the first `logenumber` eigenvalues are shown.

The default value for `logenumber` is 10.

4.23 `smooth`, selection of the spectral-curve-smoothing kernel

The spectral function information in the NRG approach is collected in the form of delta peaks (or binned into narrow intervals). To obtain a continuous function, these raw results need to be broadened into a smooth curve. The parameter `smooth` controls how this procedure is performed. The following values are allowed:

- `old` - log-Gaussian kernel above the cutoff, Gaussian kernel below the cutoff;
- `wvd` - modified log-Gaussian kernel above the cutoff, Gaussian kernel below, smooth cross-over between the two, as proposed by Weichselbaum and von Delft;

- `new` - modified log-Gaussian kernel above the cutoff, Gaussian kernel below, smooth cross-over between the two, but the cross-over function takes a different argument;
- `newsc` - slightly modified version of `new`, suitable for problems with a superconducting gap, since it allows to better resolve the in-gap states;
- `lorentz` - Lorentz broadening with fixed width on all energy scales.

In the following, we use the convention that ω is the frequency of the (output) spectral function, E is the energy of the delta peak in the raw spectrum (input).

The kernel for `smooth=old` is:

$$w(\omega, E) = lG(\omega, E)\theta(\omega E)\theta(|\omega| - \Omega) + G(\omega, E)\theta(\Omega - |\omega|). \tag{6}$$

The functions lG and G are the log-Gaussian kernel

$$lG(\omega, E) = \frac{e^{-b^2/4}}{bE\sqrt{\pi}} \exp\left[-\left(\frac{(\log\omega - \log E}{b}\right)^2\right], \tag{7}$$

and the Gaussian kernel

$$G(\omega, E) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{\omega - E}{\sigma}\right)^2\right]. \tag{8}$$

The parameter b is set by `loggauss_b`, while the parameter σ by `gauss_b`. The cross-over energy Ω is set by `limitLL`. This approach is described in Ref. [37].

The kernel for `smooth=wvd` is

$$w(\omega, E) = mlG(\omega, E)h(|E|) + \tilde{G}(\omega, E)[1 - h(|E|)], \tag{9}$$

where the modified log-Gaussian is

$$mLG(\omega, E) = \frac{\theta(\omega E)}{\alpha|\omega|\sqrt{\pi}} \exp\left[-\left(\frac{\log(\omega/E)}{\alpha} - \gamma\right)^2\right], \tag{10}$$

with $\gamma = \alpha/4$, the Gaussian is

$$\tilde{G}(\omega, E) = \frac{1}{\omega_0\sqrt{\pi}} \exp\left[-\left(\frac{\omega - E}{\omega_0}\right)^2\right], \tag{11}$$

and the smooth cross-over function is

$$h(x) = \exp\left[-\left(\frac{\ln(x/\omega_0)}{\alpha}\right)^2\right] \tag{12}$$

for $x < \omega_0$ and $h(x) = 1$ otherwise. The parameter α is set by `alpha`, ω_0 by `omega0` or `omega0_ratio`.

The kernel for `smooth=new` is similar to `wvd`:

$$w(\omega, E) = mlG(\omega, E)h(|\omega|) + \tilde{G}(\omega, E)[1 - h(|\omega|)]. \tag{13}$$

Note the different argument to the cross-over function h.

The kernel for `smooth=newsc` is

$$w(\omega, E) = mlG(\omega, E)\theta(|\omega| - \Omega) + \tilde{G}(\omega, E)\theta(\Omega - |\omega|). \tag{14}$$

Here mlG is the same as for `smooth=new`, the sharp cut-off Ω is set by `brcut`, while the width of the Gaussian ω_0 is controlled by `omega0`.

The kernel for `smooth=lorentz` is simply

$$w(\omega, E) = L(\omega, E) \tag{15}$$

with the Lorentzian function

$$L(\omega, E) = \frac{\eta}{\pi}\frac{1}{(\omega - E)^2 + \eta^2}. \tag{16}$$

The parameter η is set by `eta`.

4.24 `alpha`, width of logarithmic gaussian

The parameter `alpha` is α in Eq. (10). It controls the width of the log-Gaussian broadening kernel which is used at high frequencies. The default value is 0.3.

4.25 `omega0`, broadening kernel cross-over scale

For `smooth=wvd` and `smooth=new`, the parameter `omega0` is ω_0 in Eqs. (11) and (12). It controls the width of the Gaussian broadening kernel which is used at low frequencies and the cross-over scale between log-Gaussian and Gaussian broadening.

For `smooth=newsc`, the parameter `omega0` has only the role of fixing the kernel width.

The default behavior is to set `omega0` automatically based on the physical temperature and the scaling parameter `omega0_ratio`. See below.

4.26 `omega0_ratio`, cross-over scale in relative units

The parameter `omega0_ratio` is used to set `omega0` automatically as `omega0` = `omega0_ratio` $\times T$, where T is the physical temperature used in the construction of the density matrix, as set by the parameter `T`.

The default value of `omega0_ratio` is 1.

4.27 `brcut`, energy range for fixed-width Gaussian broadening

The parameter `brcut` controls the frequency range where fixed-width Gaussian broadening is used in the case of `smooth=newsc`. It corresponds to the parameter Ω in Eq. (14).

The default value of `brcut` is 0.

4.28 `loggauss_b`, width of the logarithmic Gaussian kernel

The parameter `loggauss_b` corresponds to the parameter b in Eq. (7), thus it sets the width of the log-Gaussian kernel which is used at high frequencies for the `smooth=old` broadening type.

The default value of `logguass_b` is 0.3, which is a reasonable choice for $\Lambda = 2$ and $N_z = 2$.

4.29 `gauss_b`, width of the Gaussian kernel

The parameter `gauss_b` corresponds to the parameter σ in Eq. (8), thus it sets the width of the Gaussian kernel which is used at low frequencies for the `smooth=old` broadening type.

The default value of `gauss_b` is 0.1.

4.30 `limitLL`, transition frequency for different kernels

The parameter `limitLL` corresponds to the parameter Ω in Eq. (6). It sets the frequency below which the Gaussian kernel is used and above which the log-Gaussian kernel is used for the `smooth=old` broadening type.

The default value of `limitLL` is zero, thus log-Gaussian broadening is performed for all frequencies (this is suitable only for $T = 0$ calculations).

4.31 `eta`, Lorentzian broadening kernel width

The parameter `eta` corresponds to the parameter η in Eq. (16), i.e., it is the width of the Lorentzian kernel used for broadening the spectral data for `smooth=lorentz`.

The default value for η is 0.

4.32 `finite`, traditional finite-temperature spectral function calculation

If the parameter `finite` is set to `true`, the spectral functions will be computed using the traditional finite-temperature algorithm as prosposed by Costi, Hewson in Phil. Mag. B 65, 1165 (1992) and Costi, Hewson, Zlatic in J. Phys. Cond. Matter 6, 2519 (1994). This approach has been superceeded by density-matrix techniques (see `dmnrg`, `cfs` and `fdm`).

The corresponding output files have names of the form `*_FT_*.[dat|bin]`.

The default value for `finite` is `false`.

4.33 `bins`, performing binning of the spectral data

In NRG, the spectral data is accumulated in the form of weighted delta peaks. In the traditional approach to spectral function calculation, which is a direct application of the Lehmann decomposition and using the Boltzmann weight instead of the density matrix, there is one delta peak for each pair of states coupled by the (creation) operator. In the density-matrix approaches, there is an additional internal sum, thus even more states. For this reason, it is more economical to accumulate the spectral data in narrow bins. The parameter `bins` controls the number of bins per frequency decade.

The default value for `bins` is 100. This is sufficient for general calculations. For "high-resolution" calculations this number may be increased (500, 1000, or so) to improve the cancellation of oscillatory features.

4.34 `discard_trim`, spectral weight bin trimming

The parameter `discard_trim` controls which spectral weight bins are dropped from consideration if they do not contain sufficient spectral weight. This removal is performed at the very end of the calculation. The reason for such trimming is that spectral broadening is a numerically relatively demanding operation which scales with the number of bins and may be significant if the parameter `bins` (bins per decade) is large. A bin is trimmed away if its weight is less than `discard_trim` $\times\Delta E$, where ΔE is the width of the bin. The total error due to trimming is of the order of `discard_trim`. If `discard_trim` is small, this error should be numerically insignificant. It can be monitored by checking if the spectral sum rules in sum-rule-conserving approachs (CFS and FDM NRG) are close to 1.

The default value for `discard_trim` is 10^{-16}, which appears to be a safe choice.

4.35 `discard_immediately`, spectral weight trimming during the calculation

In addition to trimming the bins at the end of the calculation, it is also possible to discard some spectral data on the fly, during the calculation itself. This is controlled by the parameter `discard_immediately`. In this case we are discarding individual delta peaks if their weight is lower than `discard_immediately` $\times E$, where E is the peak energy. The reason for discarding low-weight peaks is to save time adding such peaks to appropriate bins (such binning is fast, but the total time would quickly add up due to a hugh number of peaks that do not contribute a meaningful ammount). The total error due to trimming is estimated to be of the order of `discard_immediately`, although it is difficult to fix a strict bound. If `discard_immediately` is small, this error should be numerically insignificant.

The default value for `discard_immediately` is 10^{-16}, which appears to be a safe choice.

4.36 `NN1`, do $N/N+1$ patching

If the spectral function is computed using the patching approach (which is the case for the traditional and DMNRG algorithms, but not for CFS and FDM algorithms), by setting `NN1` to `true` we perform $N/N+1$ patching rather than $N/N+2$ patching. This is necessary if the calculation is performed for larger values of Λ. The $N/N+2$ patching requires well determined spectral data in the interval [`goodE`:$\Lambda^2\times$`goodE`] (the patching window), while the $N/N+1$ patching requires well determined spectral data in the interval [`goodE`:$\Lambda\times$`goodE`].

For two and more channels, the $N/N+1$ patching is automatically performed, thus this setting is only relevant for single-channel calculations.

The default value for `NN1` is `false`.

4.37 `NN2even`, use even iterations in $N/N+2$ patching

The parameter `NN2even` controls if the spectral data is taken from even iterations (for `true`) or odd iterations (for `false`) in the $N/N+2$ patching approach.

The default value for `NN2even` is `true`.

4.38 `NN2avg`, average even and off $N/N+2$ patching spectra

If the parameter `NN2avg` is `true`, the result of the $N/N+2$ patching is the same as if the results for `NN2even=true` and `NN2even=false` were averaged.

The default value for `NN2avg` is `false`.

4.39 `NNtanh`, use a `tanh` window function in the patching algoritm

In the patching approach it is possible to combine the spectral function information from different chain lengths using different cross-over function. Conventionally, one uses "tent-like" window functions, which are linear. Alternatively, one can use a smoother cross-over function (a tangens hyperbolicus function) by setting the parameter `NNtanh` to `true`. In practice the difference is small.

The default value for `NNtanh` is `false`.

4.40 `goodE`, the energy window for the patching approach

The parameter `goodE` controls the window of energies in the patching approach to spectral function calculation. It is the most important free parameter and must be chosen carefully, possibly by sweeping its value and computing spectral functions for a known problem, then choosing the most appropriate value. In particular, the value `goodE` affects to what degree the Friedel sum rule and the normalization sum rule are fulfilled. Note that this setting is only relevant for the conventional algorithm and the DM-NRG algorithm, while the full-Fock-basis approaches (CFS and FDM) accumulate the raw spectral function data without patching (and thus without any "fudge parameter", which `goodE` essentially is).

The exact meaning of `goodE` depends also on the value of parameter `NN1`. If `NN1` is `false` (which is the default value for single-channel problems), the window of energies is [`goodE`:$\Lambda^2\times$`goodE`]. If `NN1` is `true` (which is the default for multi-channel problems, but has to be manually set for single-channel problems), the window of energies is [`goodE`:$\Lambda\times$`goodE`].

The default value for `goodE` is 2.0, which is a reasonable choice for calculations with $\Lambda = 2$.

4.41 `savebins`, save raw binned data for spectral functions

If `savebins` is `true`, the program will save the raw binned spectral function data in addition to the broadened spectral functions. The binned data is saved in the form of binary files with a suffix `.bin`. (The broadened spectral functions are saved in files with the same name, but with a suffix `.dat`.)

If the value of a suitable broadening parameter is not known in advance, this options allows to determine it without unnecessary recalculations. The broadening can be performed with a stand-alone tool named `bw`, which is also part of NRG Ljubljana.

The default value of `savebins` is `false`.

4.42 `broaden`, save broadened spectral functions

If `broaden` is `true`, the program will save the broadened spectral functions. These files have suffix `.dat`. See option `savebins` for saving the raw (binned but not broadened) data as well.

The default value of `broaden` is `true`.

4.43 `broaden_max`, `broaden_min`, `broaden_ratio`, mesh parameters for the broadened spectral function

The mesh for broadened spectral functions (`*.dat` output files) is defined by the points on the mesh

$$\omega_n = \text{broaden\_max} \times \text{broaden\_ratio}^{-n}, \tag{17}$$

where $n = 0, 1, 2, \ldots, n_{\max}$. Here $n_{\max}$ is defined so that $\omega_{n_{\max}} > \text{broaden\_min}$. The same mesh (up to sign) is used for both positive and negative frequencies.

The default values are `broaden_max`=2.5, `broaden_ratio`=1.05, while `broaden_min` is configured automatically via `broaden_min_ratio`.

4.44 `broaden_min_ratio`, set `broaden_min` automatically

If `broaden_min` is not a positive value (and this is the default), then the parameter `broaden_min_ratio` is used to set `broaden_min` automatically as

$$\text{broaden\_min} = \text{broaden\_min\_ratio} \times \omega_N, \tag{18}$$

where ω_N is the energy scale at the last NRG step, thus the last point of the mesh is comparable to the smallest energy scale of the problem (for `broaden_min_ratio` of order one.

The default value of `broaden_min_ratio` is 3.

4.45 `dumpenergies`, save all energies to a file

If `dumpenergies` is `true`, then all eigenenergies are dumped to a file `energies.nrg` (in the first run) and `energies.dmnrg` (in the second run, when the DM approach is used). The files are ASCII tables, states are separated by the iteration number, and stored for each invariant subspace separately. For each invariant subspace, they are sorted in incremental order.

This option is mostly useful for debugging. See option `dumpannotated` for producing more readable output of the eigenspectra.

The default value of `dumpenergies` is `false`.

4.46 `saveall`, save all energies and multiplicity prefactors

(obsolete, to be removed)

4.47 `dumpannotated`, number of annotated eigenvalues to save to `annotated.dat`

The parameter `dumpannotated` turns on saving of the eigenspectra at each iteration in the form of a human-readable format in `annotated.dat`. The integer value of `dumpannotated` sets the number of states saved at each iterations. The code can automatically detect states which are narrowly clustered (i.e., essentially degenerate) and output them as a single line (see parameter `dumpgroups`). The format of the file is

$$E \quad (\text{qn}1), (\text{qn}2), \ldots \quad [\text{deg}], \tag{19}$$

where E is the energy, $(\text{qn}i)$ is the list of the quantum numbers which uniquely determine the invariant subspace (there can be several degenerate states), while deg is the total degeneracy of the states. The meaning of E depends on the value of `dumpscaled`.

The default value of `dumpannotated` is 0, thus the file `annotated.dat` is not created.

4.48 `dumpscaled`, rescale energies in ω_N units in `annotated.dat`

If `dumpscaled` is `true`, the energies in `annotated.dat` are rescaled in the natural units of ω_N, thus they will be of order 1. If `dumpscaled` is `false`, the energies will be in absolute units of the half-bandwidth D. This latter setting is useful in case of a calculation of an impurity coupled to a superconducting bath to read-off the energies of the sub-gap states (aka Andreev bound states).

The default value of `dumpscaled` is `true`.

4.49 `dumpabs`, use total energies in `annotated.dat`

If `dumpabs` is `true`, when writing energies to `annotated.at` the zero energy at the current shell is not subtracted, thus the output are the actual excitation energies. This options needs to be used in conjunction with `dumpscaled=false`.

The default value of `dumpabs` is `false`.

4.50 `dumpprecision`, number of digits of precision in `annotated.dat`

The parameter `dumpprecision` sets the number of digits of precision for the energies in `annotated.dat`.

The default value of `dumpprecision` is 8.

4.51 `dumpgroups`, group degenerate states in `annotated.dat`

If `dumpgroups` is `true`, states with nearby energies are considered to be degenerate and are grouped together in `annotated.dat`. The criterium for energies to be considered equal is specified by `grouptol`.

The default value of `dumpgroups` is `true`.

4.52 `grouptol`, energy difference for considering two states to be degenerate

The parameter `grouptol` defines the maximal difference between two eigenenergies for the corresponding states to be considered degenerate when saving the spectra to `annotated.dat`.

The default value of `grouptol` is 10^{-6}, which is a suitable choice in most case.

4.53 `dumpdiagonal`, log diagonal matrix elements of singlet operators

The parameter `dumpdiagonal` enables logging of diagonal matrix elements of singlet operators. This is particularly useful in the presence of the gap, where the matrix elements are different in the long-chain limit. They thus correspond to the different many-body states of the system which have different properties. The parameter `dumpdiagonal` takes an integer number which specifies the number of the matrix elements which are saved for each singlet operator. The output goes to the log file (i.e., to the standard output).

The default value of `dumpdiagonal` is 0 (logging turned off).

4.54 `ops`, list of operators to be computed

The parameter `ops` is a list of all operators that need to be computed during the NRG run. This includes the operators whose expectation values are computed (output files `custom` and `customfdm`), as well as operators that are required for computing the matrix elements which appear in the spectral functions. The spectral functions are specified using `specs`, `specd`, `spect`, and `specb` by pairs of operators which should appear in the `ops` list.

[TO DO: automatically add the operators specified in `spec*` to the `ops` list.]

The operators are defined in `operators.m` (bundled with NRG Ljubljana, contains many commonly occuring operators), `customoperators.m` (user defined definitions) or `modeloperators.m` (user defined model-specific definitions).

The default value of `ops` is an empty string.

4.55 specs, list of spectral functions with singlet operators

The parameter specs contains the list of the spectral functions which are computed as correlators of two spin-singlet operators A and B. The calculated spectral function is

$$\rho_{AB}(\omega) = -\frac{1}{\pi}\mathrm{Im}\left[\langle\langle A; B\rangle\rangle_{\omega+i\delta}\right]. \tag{20}$$

The output files are corr_T_dens_A-B.dat. Here T denotes the algorithm used to compute the spectral function (FT for the traditional algorithm, DMNRG for the density-matrix NRG, CFS for the complete-Fock-space algorithm, and FDM for the full-density-matrix approach). The operators A and B need to be specified in ops.

The spectral functions are specified as pairs A-B. For example, if the charge-charge dynamical susceptibility

$$\chi_c(\omega) = -\frac{1}{\pi}\mathrm{Im}\left[\langle\langle n; n\rangle\rangle_{\omega+i\delta}\right] \tag{21}$$

is to be computed, one specifies specs=n_d-n_d and the operator n_d has to be appended to ops.

The default value of specs is an empty string.

4.56 specd, list of spectral functions with doublet operators

The parameter specd contains the list of the spectral functions which are computed as correlators of the spin-doublet operators A and $B^\dagger$. The calculated spectral function is

$$\rho_{AB}(\omega) = -\frac{1}{\pi}\mathrm{Im}\left[\langle\langle A; B^\dagger\rangle\rangle_{\omega+i\delta}\right]. \tag{22}$$

Note that the second operator is conjugated.

For example, if the impurity spectral function

$$A_\sigma(\omega) = -\frac{1}{\pi}\mathrm{Im}\left[\langle\langle d_\sigma; d_\sigma^\dagger\rangle\rangle_{\omega+i\delta}\right] \tag{23}$$

is to be computed, one specifies specd=A_d-A_d and the operator A_d has to be appended to ops.

The output files are named spec_T_dens_A-B.dat. See specs for more details.

For handling spin, there are some differences depending on the symmetry type of the problem (specified by symtype). In particular, for symtype=QSZ the spin is handled automatically. This means that it is only necessary to specify the generic operator A_d and the two spin projections will be output separately in files named spec_T_dens_A-B-u.dat and spec_T_dens_A-B-d.dat for spin-up and spin-down, respectively.

For non-diagonal spectral functions (with $A \neq B$), what is computed is really the symmetrized spectral function

$$\rho_{AB}(\omega) = -\frac{1}{\pi}\frac{1}{2}\mathrm{Im}\left[\langle\langle A; B^\dagger\rangle\rangle_{\omega+i\delta}\langle\langle B; A^\dagger\rangle\rangle_{\omega+i\delta}\right]. \tag{24}$$

The default value of specd is an empty string.

4.57 spect, list of spectral functions with triplet operators

The parameter spect is the list of all spectral functions which are computed as correlators of the spin-triplet operators A and B. The computed function is thus the dynamical spin susceptbility

$$\chi_s(\omega) = -\frac{1}{\pi}\mathrm{Im}\left[\langle\langle A; B\rangle\rangle_{\omega+i\delta}\right], \tag{25}$$

where A and B are the spin operators.

The output files are named spin_T_dens_A-B.dat. See specs for more details.

This is only relevant for symmetry types with full SU(2) spin symmetry, such as QS and SPSU2. In other cases, the susceptibilities can be computed using "singlet" operators and specs. (A "singlet" operator in the NRG Ljubljana code is any operator which is invariant under the symmetry operations. If there is no spin symmetry, the spin operators are thus also classified as singlet operators.)

The default value of spect is an empty string.

4.58 `specb`, list of generalized spectral functions to compute

(obsolete)

This option was used to compute the generalized spectral function F to be used in the self-energy trick $\Sigma = UF/G$. There are more elegant ways to do this, thus `specb` is scheduled to be removed.

4.59 `reim`, output the "imaginary part" of the spectral functions

In some very special cases with complex (as in complex number!) Hamiltonians, the matrix elements which appear in the Lehmann expression for the spectral function may be complex numbers. In such cases, one should take the imaginary parts, do a Kramers-Kronig transformation, and add that to the real part to obtain the true spectral function.

The parameter `reim` controls if the imaginary parts (when present) are saved. They are saved both in ASCII output files (suffix `.dat`) and in raw binary output files (suffix `.bin`).

The default value of `reim` is `false`.

4.60 `trsq`, calculare the expectation values of singlet operators squared

(seldom used)

For the operators listed as the value of `trsq`, the code will compute the expectation values of the operators squared, $\langle O^2 \rangle$. This is seldom needed, since one can (usually) define the squares explicitly.

The results are saved in an output file named `customsq`.

The default value of `trsq` is an empty string.

4.61 `specgt`, list of conductance curves to compute

It is possible to compute the temperature dependence of the conductance through a quantum dot described by the SIAM in a single NRG calculation. The pairs of operators which are listed in `specgt` define spectral functions which are integrated over together with a suitable weight function (derivative of the Fermi-Dirac function) to produce a conductance result according to the Meir-Wingreen formula, for instance

$$G(T) = G_0 \int_{-\infty}^{+\infty} \pi\Gamma \left(-\frac{\mathrm{d}f}{\mathrm{d}\omega} \right) A(\omega, T) \mathrm{d}\omega. \tag{26}$$

More accurately, what the parameter `specgt` enables to compute is

$$I_0(T) = \int_{-\infty}^{+\infty} \left(-\frac{\mathrm{d}f}{\mathrm{d}\omega} \right) A(\omega, T) \mathrm{d}\omega, \tag{27}$$

i.e. the result is the conductance in units of G_0 but without the $\pi\Gamma$ factor.

It turns out that calculating the integrals using the on-shell spectral function data produces surprisingly accurate results, not much different from those obtained by explicitly calculating a range of finite-temperature spectral functions using the FDM approach and then doing the integration explicitly. This is due to the "windowing-effect" of the derivative of the Fermi-Dirac function, which requires the spectral information only over a limited frequency range.

The default value of `specgt` is an empty string.

4.62 `gtp`, control parameter p for conductance-curve calculations

The calculation of conductance via direct integration of the spectral function (enabled by the parameter `specgt`) containes an arbitrary parameter p, similar to the $\bar{\beta}$ parameter in the calculation of thermodynamic quantities. The value of this parameter is set by `gtp`.

The default value of `gtp` is 0.7, which appears to be suitable for $\Lambda = 2$ calculations. For different values of Λ, the parameter has to be determined by checking asymptotic values, for instance the $T = 0$ limit of the conductance.

4.63 `speci1t` and `speci2t`, list of first-moment and second-moment curves to compute

These parameters are related to `specgt`. The pairs of operators which are list in `speci1t` and `speci2t` define spectral functions which are intergrated over together with weight functions (derivative of the Fermi-Dirac function multiplet by ω or ω^2, for `speci1t` and `speci2t`, respectively). In other words, the results are

$$I_i(T) = \int_{-\infty}^{+\infty} \left(-\frac{\mathrm{d}f}{\mathrm{d}\omega}\right) \omega^i A(\omega, T) \mathrm{d}\omega, \tag{28}$$

with $i = 1$ and $i = 2$, respectively. The results for `specgt` correspond to the same expression with $i = 0$. These quantities are required to calculate other transport properties, such as thermopower.

The default value of `speci1t` and `speci2t` is an empty string.

The same value of `gtp` is used as for the conductance curves specified by `specgt`.

4.64 `speckubo`, conductance within the Kubo linear-response formalism

(not working properly, to be removed)

4.65 `cond`, list of correlators for computing linear conductance from the charge fluctuations in the leads

The conductance can be computed within the linear-response formalism as a correlator of total charge operators in the conduction leads. The parameter `cond` is a list of operator pairs which correspond to the correlators. See section 7 about how NRG Ljubljana handles non-local operators (i.e., operators which are defined in the Wilson chains).

This technique is useful for cases where the Meir-Wingreen formula is not applicable and one cannot compute the conductance from the quasiparticle phase shifts either (say due to lack of the left-right symmetry).

The default value of `cond` is an empty string.

4.66 `specchit`, list of temperature-dependent susceptibilities to compute

In the NRG approach, it is straight forward to compute susceptibilities of conserved operators. For example, in case of full SU(2) spin symmetry, S and S_z are conserved quantum numbers and it is easy to compute the thermodynamic spin susceptibility as

$$\chi_{S_z,S_z}(T) = \frac{1}{Z} \sum_n \left[e^{-\bar{\beta} E_n} S_z^2 \right], \tag{29}$$

where E_n are the energies of all states at the current NRG iteration (in units of the characteristic energy ω_N), $\bar{\beta}$ is the rescale factor (see parameter `betabar`), and Z is the partition function

$$Z = \sum_n e^{-\bar{\beta} E_n}. \tag{30}$$

The sum over S_z^2 is easily computed, since the invariant subspeces are indexed by the conserved quantum number S, thus one sums from $S_z = -S$ to $S_z = S$. Because of the SU(2) spin symmetry, the two other susceptibilities, χ_{S_x,S_x} and χ_{S_y,S_y}, can be calculated in exactly the same way and produce the same result.

Now consider the case where the problem has no spin symmetry at all. The spin operators are not conserved. The susceptibility tensor now has to be computed from the definition, for example the out-of-diagonal component can be expressed as

$$\chi_{S_x,S_z}(T) = T \int_0^\beta S_x(\tau) S_z(0) \mathrm{d}\tau. \tag{31}$$

This is what the parameter specchit makes possible. The argument is a list of operators that define the susceptibility functions to be computed. Note that the corresponding spin operators are *global* spin operators which include not only the impurity spin but also the spins of all Wilson-chain sites. See section 7 about how NRG Ljubljana handles non-local operators. The method is described in Ref. [38] (appendix).

The output files are named chit_CHIT_dens_A-B.dat, where A and B are the operators which defined the susceptibility function.

See also chitp and chitp_ratio.

The default value of specchit is an empty string.

4.67 chitp, parameter p in the susceptibility calculations

The parameter chitp controls the effective temperature which appears in the Boltzmann factors in the calculation of general susceptibilities defined in specchit. The effective temperature at the N-th NRG iteration is defined as

$$T_{\text{eff}} = \texttt{chitp} \times \omega_N, \tag{32}$$

where ω_N is the characteristic energy scale at the N-th step.

It is crucial to tune this parameter to an appropriate value. Such tuning can be performed by calculating the susceptibilities using the conventional algorithm and the generalized approach and setting p so that the results fully overlap. In addition, it may be noted that the convergence with respect with the truncation cut-off E_{cutoff} of the susceptibilities computed using the generalized approach is rather slow and values in excess of 10 are absolutely necessary (appropriate values are found to be 14 or even 16).

See also chitp_ratio.

The default value of chitp is 1.

4.68 chitp_ratio, set parameter p in the susceptibility calculations based on $\bar{\beta}$

If the parameter chitp_ratio is set to a positive value, the paramter p (see the description of the parameter chitp) is determined as $p =$chitp_ratio$/\bar{\beta}$. The idea is to relate the parameter $\bar{\beta}$ which controls the effective temperature in the calculations of susceptibilities using the standard algorithm (td outfile file) with those computed using the generalized aapproach (chit* output files).

By default, chitp_ratio is not used.

4.69 dm, enable computation of density matrices

If the parameter dm is set to true, the code will use the density-matrix algorithm [19]. The NRG calculation is then performed in two sweeps. During the first, the Hamiltonian matrices are constructed and diagonalized, the results are stored on disk. Then the density matrices are computed and stored on disk. In the second sweep, the density matrices and eigenstates are read from disk and the results (spectral functions, expectation values) are computed.

The parameter dm is automatically set to true if at least one of the parameters dmnrg, cfs, fdm or mats is enabled. It is seldom necessary to enable dm manually (mostly for debugging purposes).

The default value of dm is false.

4.70 T0opt, enable optimizations for zero-temperature spectral function calculations

(seldom used)

In some spectral function calculation algorithms (finite and dmnrg), the $T = 0$ spectral functions can be computed more efficiently by taking into account that for some excitation energies the Boltzmann weights will be essentially zero. The value of the parameter T0opt sets the maximum number of states from each invariant subspace to take into account. This optimization should be used with care (and only after testing by comparing to reference results!).

The default for T0opt is that the optimization is disabled.

4.71 dmnrg, calculate spectral functions using the density-matrix algorithm

The parameter dmnrg enables the DM-NRG algorithm [19] for calculating spectral functions. This approach must be used whenever the low-energy fixed point affects the high-energy spectral features, for instance in the presence of the magnetic field (but also in many other cases). This algorithm does not fulfill the spectral normalization sum rule, but in most cases the results that it produces are reliable and do not differ much from the results obtained using the CFS or FDM algorithms (but not in all cases: a cautions user will probably want to perform some test calculation with the full Fock space algorithms and compare). It is, however, faster than the CFS and FDM algorithms, since the discarded states do not need to be computed and more efficient diagonalization algorithms may be used (see parameter diag). At finite temperatures, the FDM algorithm is always recommended.

The output spectral functions computed using this approach have _DMNRG_ in their file names.

The default value of dmnrg is false.

4.72 cfs, calculate spectral functions using the complete Fock space algorithm

The parameter cfs enables the CFS algorithm [39] based on the idea of constructing a complete basis of states for the entire Fock space of the problem from the discarded states [40, 41]. The spectral functions computed using this approach fulfill the normalization sum-rule to many digits of precision by construction. This algorithm is equivalent to the FDM at zero temperature.

The output spectral functions computed using the approach have _CFS_ in their file names.

The default value of cfs is false.

4.73 fdm, calculate spectral functions using the full density matrix algorithm

The parameter fdm enables the FDM algorithm [42] based on the complete Fock space concept and with a density matrix with contributions from all energy shells. This is the recommended way of computing finite-temperature spectral functions in NRG.

The output spectral functions computed using the approach have _FDM_ in their file names.

The default value of fdm is false.

4.74 fdmexpv, compute the expectation values using the FDM algorithm

The parameter fdmexpv enables the computation of the expectation values at the physical temperature T (i.e., the one set by the parameter T) using the full-density-matrix algorithm. This is more reliable approach than using the T-dependent values computed using the conventional algorithm which are tabulated in the output file custom and interpolating to the chosen temperature. The result of the FDM algoritm is saved to a different output file named customfdm, which contains a single line with the expectation values at the temperature T.

The default value of fdmexpv is false.

See also fdmexpvn.

4.75 fdmexpvn, step number for computing the expectation values

The parameter fdmexpvn determines the step number where the expectation values are computed using the FDM algorithm. The default value is 0, which is the recommended setting, because in this case the singlet operators do not need to be recomputed for later iterations (unless those same operators are needed for computing spectral functions). It is also possible to set fdmexpvn to -1, in which case the contributions are collected from all NRG steps.

Irrespective of the value of fdmexpvn, the results should be numerically the same (up to round-off errors).

4.76 `mats`, calculate FDM spectral functions on the imaginary frequency axis

The parameter `mats` enables the FDM algorithm for computing the spectral functions. The difference with `fdm` is that the spectral function is computed on a fixed grid of Matsubara frequencies.

The same result may be obtained by running the usual FDM calculation (with `fdm=true`) and saving the raw spectral data (with `savebins=true`), then calculating the imaginary-axis spectral functions using an external tool `mats`. (This alternative approach is, in fact, more flexible and even mcuh faster due an inefficient implementation of `mats` in the present version of the code.)

The output of such calculation are full Green's functions. They have `_MATS_` in their file names.

The default value of `mats` is `false`.

4.77 `nromegan`, the number of Matsubara points in calculation of imaginary-axis spectral functions

The parameter `nromegan` sets the total number of the Matsubara points $\omega_n = iT\pi(2n+1)$ using in the computation of the spectral functions on the imaginary axis when `mats=true`. The numerical effort scales linearly with `nromegan`.

The default value of `nromegan` is 100.

4.78 `width_td`, `prec_td`, formatting of the numercal data in the thermodynamics output file `td`

The parameters `width_td` and `prec_td` control the column width and the number of decimal points of precision for the output of thermodynamics calculation saved to the output file `td`. Typically, the value of `prec_td` should by at least 6 larger than `width_td` to achieve nicely aligned formatting.

The default value of `width_td` is 12, while the default value of `prec_td` is 6.

4.79 `width_custom`, `prec_custom`, formatting of the numercal data in the expectation value output file `custom`

The parameters `width_custom` and `prec_custom` control the column width and the number of decimal points of precision for the output of expectation value calculations saved to the output file `custom`. Typically, the value of `prec_custom` should by at least 6 larger than `width_custom`.

The default value of `width_custom` is 12, while the default value of `prec_custom` is 6.

4.80 `prec_xy`, precision of the broadened spectral functions

The parameter `prec_xy` controls the number of digital positions of precision for the broadened spectral-function data.

The default value of `prec_xy` is 6.

4.81 `safeguard`, keep additional states in case of a near degeneracy

In the NRG calculations, it is found that the states are clustered together. In addition, if there are symmetries which are not taken into account explicitly, then multiple states will be exactly degenerate. If the truncation is performed inside such a cluster of (nearly) degenerate states, this will slightly spoil the symmetry even if the truncation is performed at the top of the spectrum. If the corresponding symmetry breaking is a relevant operator, then the spurious breaking will grow as the iteration proceeds. A case in point are problems with an underscreened impurity in the absence of an external magnetic field, where the impurity moment should remain unscreened and there should be residual entropy. If the symmetry type is, for example, QSZ, this is not guaranteed and small errors in numerics will lead to full polarization of the residual spin and quenching of the residual entropy.

To avoid such issues, one can set the parameter `safeguard` to a sufficiently large value, so that the truncation will always occur in a "gap" having the width of at least `safeguard` (in units of the characteristic energy scale ω_N). To ensure this, additional states are retained (but not more than `safeguardmax` states).

The default value of `safeguard` is 10^{-5}, which is sufficient in most cases. If `safeguard` is too large, large numbers of additional states would have to be retained, which is not always possible (and the clipping by `safeguardmax` will undo any benefit of keeping additional states). If `safeguard` is too small, the errors may persist.

See also `safeguadmax` and `fixeps`.

4.82 `safeguardmax`, maximal number of additional states for `safeguard`

The parameter `safeguardmax` specifies the maximum number of additional states to keep in the algorithm for truncating the spectrum in a "gap". This settings ensures that the memory constraints are not exceeded.

The default value of `safeguardmax` is 200, which is usually sufficient if `safeguard` is set to its default value of 10^{-5}.

4.83 `fixeps`, fix spurious splitting of states due to floating-point round-off errors

In numerical calculations there are always some round-off errors. In NRG, there are some instances where small round-off errors lead to large cumulative errors. This happens, for example, when small differences in energies correspond to the presence of relevant operators. The splitting will then amplify during the iteration and may blow up. To prevent such errors, NRG Ljubljana allows to automatically correct small splittings. The value of parameter `fixeps` indicates the maximum energy difference (in units of the characteristic energy scale ω_N) for states to still be considered degenerate. The energies of all states in a detected degenerate cluster are then replaced by the average energy. This automatically "cures" some effects of the systematic numerical round-off errors.

Usually, `fixeps` should be small. Values up to 10^{-10} are probably safe. Nevertheless, use extreme caution when using this option. Especially, if there is some physical parameter that is small, large `fixeps` will probably wipe it out.

The default value of `fixeps` is 10^{-15}, which is a suitable choice for curing the round-off errors in the 2-3 least significant bits.

4.84 `diag`, select the diagonalization routine to be used

The parameter `diag` allows the user some control over the diagonalization routines used.

For real problems, `diag` may be either `dsyev` or `dsyevr`. The latter routine allows to compute only a part of the spectrum and is faster (for sufficiently large matrices). For full Fock space calculation, one needs to fully diagonalize the matrix, thus one should then use `dsyev`.

For complex problems, `diag` may be either `zheev` or `zheevr`. The latter routine allows to compute only a part of the spectrum and is typically faster. For CFS and FDM methods, one should use `zheev`.

The default value of `diag` is `default`, which currently implies `dsyev` for real problems and `zheev` for complex problems.

See also `diagratio` and `dsyevrlimit`.

4.85 `diagratio`, fraction of states which are computed in the diagonalization

The parameter `diagratio` is the fraction of all states within each invariant subspace that are determined by the diagonalization routine that allows partial diagonalizations (currently `dsyevr` and `zheevr`). The number depends on the number of channels. For single-channel problems, it should be somewhat larger than $1/4$. For two-channel problems, it should be somewhat larger than $1/16$. Suitable numbers can be determined by trial and error.

If `diagratio` is too low and an insufficient number of states is computed to fulfill the truncation criteria, then the ratio is multiplied by two (for the current iteration) and the diagonalizations are restarted. Such recalculation are sometimes required in the initial NRG steps.

The default value of `diagratio` is 1, thus all states are computed.

See also `diag`.

4.86 `dsyevrlimit`, minimal matrix size for `dsyevr` diagonalization

For small matrix sizes it is found that the diagonalizations using `dsyev` are faster. Thus the `dsyevr` routine (when selected using the parameter `diag`) is only used for matrixes larger than `dsyevrlimit`.

The default value of `dsyevrlimit` is 100.

4.87 `restart`, restart diagonalizations if safe truncation not possible

If partial diagonalizations are performed, it is possible that during the truncation step it turns out that in one of the subspaces an insufficient number of states had been computed. In this case, if `restart=true`, a recalculation will be performed automatically.

The default value of `restart` is `true`. There is no good reason to disable this functionality.

See also `restartfactor`.

4.88 `restartfactor`, multiplication factor for `restart`

If recalculation is necessary, the parameter `diagratio` will be multiplied by `restartfactor`. This rescaling is only valid in the current NRG step; `diagratio` is reset to its initial value in the next NRG step.

The default value of `restartfactor` is 2.

4.89 `checkdiag`, perform additional tests of the results from the diagonalization

If `checkdiag=true`, after each diagonalization the computed eigenvalues and eigenvectors are tested for finiteness, and eigenvectors are tested for normalization and orthogonality. This is usually not needed, but may be used for testing if there is any suspicion that the diagonalization routines are producing incorrect results.

The default value of `checkdiag` is `false`.

4.90 `checkrho`, test if density matrices have trace 1

If `checkrho=true`, the normalization condition $\mathrm{Tr}[\rho] = 1$ is tested for the density-matrices used in DMNRG and CFS algorithms.

The default value of `checkrho` is `false`.

4.91 `calc0`, perform calculation at the 0-th step

The parameter `calc0` controls if the calculations are performed also at the 0-th step of the iteration, i.e., immediately after the NRG code starts, before first diagonalizations.

The default value of `calc0` is `true`. There is no good reason to turn this off.

4.92 `tdht`, calculate thermodynamic properties for $T > D$

(rarely needed, undocumented, use with care)

4.93 `spin`, conduction-band electron spin (multiplicity)

The parameter `spin` specifies the multiplicity $2S + 1$ for the particles in the conduction band. For most symmetry types, the value is 2, which is the default. For symmetry types `SL` and `SL3`, i.e., for spinless electrons, it is automatically set to 1. For symmetry type `ANYJ`, the particles in the continuum carry arbitrary spin and the user has to explicitly set a suitable value for `spin`.

This parameter controls the dimension of the Hilbert subspace for the chain site(s) added in each step of the NRG iteration:

$$\dim = 2^{\mathrm{ch}\times(2S+1)}, \tag{33}$$

where ch is the number of channels.

The default value is suitable for all symmetry types, except for `ANYJ`.

4.94 `tri`, which tridiagonalization approach to used

The parameter `tri` selects the tridiagonalization approach for determining the Wilson chain coefficients. The following settings are accepted:

- `old` - direct use of the recursion relations;
- `sc` - direct use of the recusion relations, extended for the superconducting case;
- `cpp` - use the tridiagonalization code compiled in the C++ part of the program;
- `orth` - use recursion relations and make use of the orthogonality requirements (this technique is numerically more stable than `old`);
- `none` - do not output the coefficient table;
- `manual` - load the discretization coefficients from files.

Options `old`, `cpp` and `orth` produce (or should produce) essentially the same result and differ in the numerical procedure used. For `old` and `orth`, the coefficients are computed in Mathematica part of the code. For `cpp`, they are computed in the C++ part of the code. (If the arbitrary-precision numerics library is not available, then the tridiagonalization code is disabled and `cpp` cannot be used.)

For superconducting case, one must use `sc`.

The option `none` instructs `nrginit` to produce the `data` file without the coefficient tables. The idea here is that they will be appended by a stand-alone tool which runs after `nrginit`.

The option `manual` instructs `nrginit` to read the coefficient tables from files. The idea here is that they be computed by a stand-alone tool which runs before `nrginit`.

The default value of `tri` is `old`, which is not the fastest approach, but it is robust.

4.95 `preccpp`, precision for the C++ tridiagonalization code

The parameter `preccpp` controls the number of digits of precision for the arbitrary-precision numerics used in the tridiagonalization code in the C++ part of the code.

A good indicator for too low numerical precision is a deviation of the asymptotic values of the Wilson chain coefficients from values around 1.

The default value of `preccpp` is 2000.

4.96 `checksumrules`, check sum-rules for doublet operators

The parameter `checksumrules` turns on the tests for the sum rules of the doublet oprator matrix elements. This is only useful for low-level debugging.

The default value of `checksumrules` is `false`.

4.97 showEs, show interval boundaries in the patching approach

(rarely needed, undocumented)

4.98 globalB and globalBx, global magnetic field

These two settings add local magnetic fields on the Wilson chain sites. Note that this is not a proper way to model the presence of the magnetic field in the continuum, because it breaks the scale invariance of the problem and opens a gap. Instead, one should separately discretize the spin-dependent densities of states. See also polarized.

The default value of globalB and globalBx is false.

4.99 polarized, enable the support for spin-dependent hybridization functions

For symmetry types QSZ and U1, setting polarized to true permits to use spin-depdendent hybridization functions. Effectively, this setting doubles the number of channels. Thus, as far as the discretization and tridiagonalization code is concerned, separate spins are handled as separate channels.

The indexing is such that the first ch sets of coefficients correspond to spin-up electrons in the different physical channels, and the following ch sets of coefficients corresponds to spin-down electrons in the various physical channels.

For symmetry type SPU1, there is also support for spin-dependent hybridization function, however the off-diagonal matrix elements in the hybridization matrix are identical in both sets.

The default value of polarized is false.

4.100 pol2x2, enable the support for full matrix structure of the hybridization functions in the spin space

For symmetry type U1, setting pol2x2 to true permits to handle the full spin dependence of the hybridization function/matrix. This effectively quadruples the number of effective channels (as regards the calculation of the Wilson chain coefficients).

The indexing is such that the first ch sets of coefficients correspond to spin-up electrons in the different physical channels, the following ch sets of coefficients corresponds to spin-down electrons in the various physical channels, and there are two identical sets of ch coefficients for the out-of-diagonal matrix elements (they are identical because the hybridization operator has to be Hermitian, and we require the coefficients to be real numbers).

The default value of pol2x2 is false.

4.101 mpi, use MPI to parallelize the matrix diagonalizations across the compute nodes

If the NRG Ljubljana code is compiled using the MPI libraries, one can parallelize the diagonalization of matrices accross the compute nodes. The scheduling is such that the slave nodes start diagonalizing the largest matrices, while the main node diagonalizes the small matrices and controls the slave nodes (after each of the small matrices is diagonalized the MPI status is checked).

The default value of mpi is true, but only if the code has been compiled using the MPI libraries. If the code was invoked with mpirun -np 1 or without mpirun, then a single instance of NRG will be started and there is no performance penalty compared with the serial version of the code.

Note also that each diagonalization may use OpenMP parallelization, thus the program supports MPI+OpenMP hybrid parallelization. If the code has been compiled without MPI, then it is still possible to do a two-level parallelization, because NRG Ljubljana supports OpenMP nested parallelism. Of course, this must also be supported by the numerical libraries (Intel MKL is fine).

4.102 saveF, calculate free energy using the complete Fock space basis

(rarely used, untested and undocumented)

See L. Merker, A. Weichselbaum, T. A. Costi, Phys. Rev. B 86, 075153 (2012), for a proper way to compute thermodynamic properties using the complete basis ideas (with the FDM NRG approach).

4.103 trunc, control storing and loading of the truncation data

Using `trunc=save`, it is possible to save the number of states kept in each subspace to a file named `cutoffs.dat`. This is thus the full information about the NRG truncation.

Using `trunc=load` or `trunc`=filename, it is possible to truncate the states according to the specification in the file `cutoffs.dat` (or in the file "filename").

The default value of `trunc` is an empty string: the truncation information is neither saved nor loaded.

This functionality is mostly useful for debugging and for exploring new algorithms. In particular, it allows to compare calculations with exactly the same truncation even if the state energies are somewhat different (and the automatic truncation would keep different numbers of states in various invariant subspaces).

4.104 noimag, do not display the imaginary part of expectation values

Usually, the observables correspond to Hermitian operators, thus the computed expectation values of "singlet operators" are real quantities. The imaginary parts are therefore not shown, because they are of the order of the numerical epsilon (or strictly zero, if complex numerics is not used). The default value of `noimag` is thus `true` and the imaginary parts are not shown (neither in `custom`, not on the standard output).

If the parameter `noimag` is set to `false`, the imaginary parts are also shown. This is mainly useful for troubleshooting (for cases when the imaginary part is expected to be zero), and for the rare case when one wants to evaluate a non-Hermitian operator.

4.105 dumpsubspaces, dump detailed information about the invariant subspaces

If the parameter `dumpsubspaces` is set to `true`, an additional file named `subspaces.dat` is created. For each NRG step, it contains a list of all invariant subspaces. For each subspaces, it shows the number of total states and the number of kept states.

The default value of `dumpsubspaces` is `false`.

4.106 done, create DONE file when finished

If the parameter `done` is set to `true`, then the NRG program will generate an empty file named `DONE` just before exiting. This file can be used as a flag file to signal completion to post processing tools.

The default value of `done` is `false`.

4.107 resume, attempt to resume an interrupted NRG calculation

If the parameter `resume` is set to `true`, the NRG program will detect the presence of temporary `unitary*` files (which contain the full information about eigenvalues and eigenvectors), `rho*` and `rhofdm*` files (which contain the full information about the density matrices) and reuse this information. This is particularly useful in very lengthy calculations which may be interrupted before completion for whatever reason (hardware faults, exceeded allocated time on a job submission system, etc.).

If the interruption occured while generating one of the above cited temporary files, one should delete the corrupted file, otherwise resuming will fail.

The default value of `resume` is `false`.

See also `removefiles`.

4.108 `forcestop`, make the NRG program stop at a chosen step number

If the parameter `forcestop` is set to a positive value N, the NRG iteration will stop just before starting the computations at step number N. The temporary files are retained. This is mostly useful for debugging purposes.

The default value of `forcestop` is negative.

4.109 `stopafter`, stop the NRG program after specific points

If the parameter `stopafter` is set to `nrg`, the program stops just after completing the first NRG sweep. At this point, all the matrices have been diagonalized and the complete information is stored in temporary files `unitary*`.

If the parameter `stopafter` is set to `rho`, the program stops just after computing the density matrices. At this point, there is full information about the eigenstates in files `unitary*`, as well as the density matrices in files `rho*` and, optionally, `rhofdm*`.

This functionality is most useful for debugging and algorithm development.

4.110 `removefiles`, remove temporary files after the calculation

If the parameter `removefiles` is set to `true`, which is the default, the temporary files are removed when they are no longer needed.

Setting `removefiles` to `false` might be useful for long computations which might become interrupted. See also `resume`.

4.111 `store`, storage options for temporary files

If the parameter `store` is `one`, a single file is generated per iteration. It contains matrices for all invariant subspaces.

If the parameter `store` is `split`, there is one file for each invariant subspace. This is needed in cases of very large problems, where one might encounter limitations of the undelying file system for storing large files (say larger than 4GB), or problems in the serialization implementation in the boost library (which might also have limits at 4GB).

The default value of `store` is `one`.

4.112 `symtype`, symmetry type

The parameter `symtype` selects the symmetry type of the calculation, i.e., which are the conserved quantities. Generally, the symmetry type is dictated both by the symmetries of the Hamiltonian and by the symmetries of the operators that need to be evaluated. One has to choose the symmetry which corresponds to that of the *least* symmetric operator that is used. As a trivial example, let us consider the SIAM in the absence of the magnetic field, which has SU(2) spin symmetry. But if (for whatever reason) one wants to compute the expectation value $\langle S_z \rangle$, which is not invariant under spin rotations, than one should use reduced symmetry, for instance U(1) spin symmetry, or even no spin symmetry at all. This example is trivial, because the expectation value is known a-priori. There are, however, less trivial cases, for example calculations of dynamic susceptibility functions which feature low-symmetry operators.

The following symmetry types are implemented:

- `QS`, $\mathrm{U(1)}_{\mathrm{charge}} \times \mathrm{SU(2)}_{\mathrm{spin}}$, i.e., conservation of charge and full invariance in the spin space.
- `QSZ`, $\mathrm{U(1)}_{\mathrm{charge}} \times \mathrm{U(1)}_{\mathrm{spin}}$, suitable for problems with external magnetic field and uniaxial magnetic anisotropy.
- `ISO` and `ISO2`, $\mathrm{SU(2)}_{\mathrm{isospin}} \times \mathrm{SU(2)}_{\mathrm{spin}}$, the two differ in the phase convention in the isospin operator definition between the two leads in the two-channel case. Simplifying a bit, one should use `ISO` for problems where the first sites of the two Wilson chains belong to the same A/B sublattice,

but `ISO2` for problems where they belong to different sublattices. This is appropriate for models at the particle-hole symmetric point (and bipartite lattice as regards the electron hopping terms) with full invariance in the spin space.

- `QSLR`, similar to `QS`, but only for left-right symmetric two-channel problems. There is an additional Z_2 parity quantum number. Not that not only the Hamiltonian, but also all operators need to be reflection invariant. The spectral functions thus need to be computed from even and odd combinations of creation/annihilation operators.
- `QSZLR`, left-right symmetric two-channel variant of `QSZ`.
- `ISOLR` and `ISO2LR`, left-right symmetric two-channel variant of `ISO` and `ISO2`.
- `ISOSZ`, $\mathrm{SU(2)}_{\mathrm{isospin}} \times \mathrm{U(1)}_{\mathrm{spin}}$, for problems at the particle-hole symmetric point and external field.
- `ISOSZLR`, left-right symmetric two-channel variant of `ISOSZLR`.
- `U1`, $\mathrm{U(1)}_{\mathrm{charge}}$, for problems with no spin symmetry.
- `SU2`, $\mathrm{SU(2)}_{\mathrm{isospin}}$, for problems with no spin symmetry at the particle-hole symmetric point.
- `SPSU2`, $\mathrm{SU(2)}_{\mathrm{spin}}$, for problems with full SU(2) spin invariance, but no charge conservation (superconductivity within the BCS theory).
- `SPU1`, $\mathrm{U(1)}_{\mathrm{spin}}$, for problems with U(1) spin symmetry and no charge conservation.
- `SPU1LR`, left-right symmetric two-channel variant of `SPU1`.
- `SL`, spinless fermions with conserved charge, $\mathrm{U(1)}_{\mathrm{charge}}$.
- `SL3`, three-channel spinless fermions with conserved charge in each channel separately.
- `DBLISOSZ`, two-channel problems with isospin symmetry in each channel separately and U(1) spin symmetry, $\mathrm{SU(2)}_{\mathrm{isospin},1} \times \mathrm{SU(2)}_{\mathrm{isospin},2} \times \mathrm{U(1)}_{\mathrm{spin}}$.
- `DBLSU2`, two-channel problems with isospin symmetry in each channel separately, $\mathrm{SU(2)}_{\mathrm{isospin},1} \times \mathrm{SU(2)}_{\mathrm{isospin},2}$, but no spin symmetry.
- `ANYJ`, $\mathrm{U(1)}_{\mathrm{charge}} \times \mathrm{U(1)}_{\mathrm{spin}}$, with arbitrary spin of the conduction-band electrons.
- `NONE`, no symmetry.

It is also possible to set `symtype=runtime`. This means that the symmetry type will be defined as part of the model definition.

The default symmetry type is `QS`.

4.113 `model`, choose the model definition (Hamiltonian)

The parameter `model` selects the model considered. Some examples for model definitions can be found in `models.m`. Additional models can be defined in `custommodels.m` (which has to be put somewhere in the path for the NRG modules). Alternatively, the value of parameter `model` may be a file name for the module definition.

The default value of `model` is `SIAM` (the single-impurity Anderson model).

4.114 `variant`, choose the problem variant

For convenience, there is an additional parameter `variant` for selecting variants of the model. This option has been introduced to allow for reuse of model definitions for slight modifications.

The default value of `variant` is an empty string.

4.115 `options`, options for the Mathematica part of the NRG program

Various options can be specified as a string value of `options`. The options have to be space separated.

The following options are of general interest:

- `WRITE`, write basis states, Hamiltonian matrices and operator matrices to files (prefixed, respectively, by `basis`, `ham`, and `op`, followed by a suffic composed of the model name, variant, and the symmetry type, and then by the invariant subspace).
- `READBASIS`, read the previously generated basis states from the `basis` files.
- `READHAM`, read the previously generated Hamiltonian matrices from the `ham` files. This is useful for extremely complex problems where the generation of the matrix representations of operators is very time consuming.
- `EPSCLIP`, clip nonrepresentably small floating point numbers to zero. This can be used if the C++ parser rejects floating points which cannot be represented as a double number.
- `TEMPLATE`, generate a template for the `data` file instead of the actual `data` file. This can than be used to generate the `data` files on computers without Mathematica.
- `COMPLEX`, enforce the generation of a complex-value version of the input file `data`, even though all numbers are actually real. This is mostly useful for troubleshooting.
- `LRTRICK`, generate symmetry-adapted basis states even if the symmetry type used does not support the Z_2 parity index. Mostly useful only for testing.
- `PARAMPRE`, enforce generation of numerical rather than symbolic matrices.
- `NOSHUR`, use `Eigensystem` rather than `ShurDecomposition` to diagonalize matrices.

4.116 `perturb`, add a perturbation to the Hamiltonian definition

It is possible to modify the Hamiltonian by adding additional terms using the parameter `perturb`. The string should contain a string with a Mathematica expression (using SNEG macros [36]). As an example, the magnetic field could be added to the SIAM Hamiltonian using `perturb=B spinz[d[]]`.

When `perturb` is used, the option `READHAM` does not work.

The default value of `perturb` is an empty string.

See also `model` and `variant`.

4.117 `mmadebug`, verbosity level for Mathematica part of the code

The parameter `mmadebug` controls the quantity of output from the Mathematica initialization code `initial.m`.

For `mmadebug` $= 0$, very little output is produced.

For `mmadebug` ≥ 1, standard output is shown.

For `mmadebug` ≥ 2: parameters defined in `[extra]` are listed; basis sets `bvc` are shown for intermediate steps in the construction of basis sets; symbolic Hamiltonian matrix representations are shown.

For `mmadebug` ≥ 3: eigenvalues are shown after the diagonalization; all constants defined in `params` are listed; operator definitions for `opstot`, `opq`, `ntot`, `neven`, `s2even`, `nodd`, `s2odd`, `seso` are shown; numerical Hamiltonian matrix representations are shown; information about the number of spaces and the numbers of coupled spaces is shown.

For `mmadebug` ≥ 4, the program attempts to rewrite the Hamiltonian definition in terms of higher-level functions (in order to determine if the second-quantization-operator definition really corresponds to user's intentions). Note that this operation may be very slow, because if consists of heuristic attempts to simplify the expressions by rewriting it in terms of higher-level functions.

The default value of `mmadebug` is 1.

4.118 U, Gamma, delta, t, model parameters

For historical reasons, some model parameters are defined in section [param]. Better practice is to put all such parameters in a separate section [extra].

For reference, U is the charge-charge (Hubbard) repulsion U, Gamma is the hybridization strength $\Gamma = \pi\rho V^2$, delta is the level energy, shifted by $U/2$, i.e., $\delta = \epsilon_d + U/2$, and t is the inter-dot hopping in some two-impurity models.

4.119 checkHc, check if the model is Hermitian

The initialization code automatically checks if the Hamiltonian H is hermitian by computing the conjugate and subtracting it from H. Sometimes Mathematica fails to simplify the symbolic expression to 0 even if the Hamiltonian is actually hermitian. In such cases, one can disable this check by setting chechHc to false.

The default value of checkHc is true.

4.120 writedir, writefnprefix, directory and prefix for writing the symbolic matrices

When options contains the option WRITE, the Hamiltonian matrices, the operator matrices, and the basis states are saved to files. The parameter writedir specifies where these files are written, while the parameter writefnprefix specifies an additional prefix for the names of those files.

The default value of writedir is an empty string (i.e., the current directory). The same for writefnprefix (i.e., no additional prefix).

See also options.

4.121 prec, number of digits of precision for arbitrary-precision arithmetic

The parameter prec controls the number of digits of precision which are used in the tridiagonalization code. The defaults are chosen according to discretization scheme (parameter tri) and are in most cases sufficiently large.

See also tri and preccpp.

4.122 nrxi, number of Wilson chain coefficients to compute

The parameter nrxi allows to override the number of Wilson chain coefficients that are computed in the initialization code. The default is to compute as many coefficients as needed for the Wilson chain length determined by Nmax or Tmin.

4.123 dos, filename for tabulated density of states (DOS) data

Some discretization schemes support arbitrary density of states in the continuum which is tabulated as a file. The parameter dos specifies the filename of those tables. This is relevant only for discretization=Z.

See also band.

4.124 xmax, range of the discretization function in FSOL.dat

The parameter xmax controls the maximal x in the discretization=Z approach for arbitrary DOS.

The default value of xmax is 30.

4.125 `solpath`, path to `FSOL.dat` file

The parameter `solpath` specifies the directory where the tabulated discretization functions `FSOL.dat` are located.

The default value of `solpath` is `..` (i.e., the parent directory). Such default is suitable for twist-averaging calculations, where separate calculations are started in numbered subdirectories.

4.126 `bcsgap`, `bcsgap1`, `bcsgap2`, superconducting gap

For problems with a superconducting gap in the continuum of states, the parameters `bcsgap`, `bcsgap1` and `bcsgap2` specify the gap. If `bcsgap` is defined, the same value is used for all channels. If `bcsgap` is not defined, one has to define both `bcsgap1` and `bcsgap2`, one for each channel.

There is no default: if no parameter is defined, the code will complain and exit.

4.127 `mMAX`, dimension of parameter matrices in the Lanczos tridiagonalization

The parameter `mMAX` controls the upper limit of the index m in the Lanczos tridiagonalization code. By default, one has

$$\mathrm{mMAX} = \max\bigl\{80, 2\mathrm{Nmax}\bigr\}. \tag{34}$$

This value is usually suitable, but it may be set manually using `mMAX`.

4.128 `disccheck`, check discretization tables

For debugging purposes, `disckcheck` enables normalization and orthogonality tests for unitary matrices appearing in the tridiagonalization.

5 Format of `data` file

The file `data` is the output from the NRG initialization code, which is implemented as a Mathematica script `initial.m`, that is started using `nrginit` command. In turn, the contents of `data` is used as the input to the NRG iteration code, which is implemented as a C++ program (source code is located in the directory src/), that is started using `nrgrun` command.

The data file starts with a header which specifies the symmetry type of the calculation, the versions of the scripts that were used in producing the data file (which might be important for testing reproducibility of the results), basic information about the model and the parameters, and some information about the discretization method and parameters.

The line starting with #! specifies the version number of the data file (7 for NRG Ljubljana 2.3.20). The data files are not guaranteed to be compatible between different versions of the code.

This is followed by the values of basic parameters, such as the number of channels, ch, impurities, and subspaces, nr. The value of the NRG scaling parameter `SCALE` is then given for informative purposes.

The eigenenergies are listed in blocks of the form:

```
{quantum numbers}
number of states
energy1 energy2 ... energyN
```

There are nr such blocks.

The matrix elements of the f matrices (Wilson chain electron creation operators) are specified as blocks of the form:

```
f channel type
number of blocks
{quantum numbers1} {quantum numbers2}
matrix
```

Parameter `channel` indexes channels, while `type` is used for model types where there are different kinds of f operators (in the absence of spin symmetry, for example, one needs to distinguish particles of different spin). The dimensions of matrices depend on the dimensions of invariant subspaces specified by the two sets of quantum numbers.

The ground state energy (in absolute units) of the initial cluster is given as the number following line containing the token `e`. This energy permits to calculate the absolute ground state energy of the full Wilson chain, which may be used to compute useful quantities by taking appropriate derivatives [43].

This is followed by several (or none) blocks of various operators. They start with a line which specifies the operator transformation properties (first letter) and the operator name (rest of the line). The symmetry types are `s` (singlet, which means singlet with respect to all symmetry operations, i.e., singlet with respect to the dynamical group of the problem), `d` (doublet with respect to $\mathrm{SU(2)_{spin}}$ and/or with respect to $\mathrm{SU(2)_{isospin}}$, usually fermionic particle creation operators), `t` (triplet with respect to $\mathrm{SU(2)_{spin}}$, i.e., the spin operator), `p` (odd-parity singlet operator, i.e., an operator which flips sign if the reflection is performed), `g` (singlet operators which is defined on the Wilson chain, see also Sec. 7), and `v` (doublet operators for self-energy calculations, now obsolete).

Finally, there are blocks containing information about the continuum. The standard Wilson chain coefficients are contained in a block starting with line `z`. First there is a list of hopping matrix elements ξ, then a list of on-site energies ζ. For superconducting problems, there is an additional block starting with line `Z`. There one first lists the on-site pairing term coefficients Δ, then the anomalous hopping coefficients κ.

Alternatively, if the tridiagonalization is performed by the C++ code, instead of the Wilson chain coefficients one has to provide the data about the density of states (DOS) in the continuum in the block starting with line `T`. The values are the (weighted) integrals of the DOS, $\epsilon^{\pm}$ and U_0.

6 Output files

6.1 `log`, `log2`, log files

The output from the NRG code goes to the standard output stream, but it is also saved to a file `log` using the UNIX `tee` command, cf. the script `nrgrun`. The errors from the NRG code (standard error stream) is redirected to the file `log2`.

The data in `log` is mostly useful for following the details of the NRG calculation, but it also provides physically relevant information that is not available from other output files.

The first few lines are the detailed version information, including some information about the code compilation. This is followed by the information about the parallelization (MPI) and, if Intel MKL is used, about the OpenMP parameters.

The values of all parameters are then dumped. Note that these are the parameters after possible automatic modification by the NRG program, thus they may differ from what is specified in the input `param` file. These are thus the values as seen from the running NRG iteration.

The header of the `data` file is also shown, followed by a list of operators read and a dump of the eigenvalues in the invariant spaces. There is also a list of operators which will be recomputed during the iteration. Typically, this includes the singlet operators, as well as all those operators which are used to compute spectral functions.

There is then a list of the expectation values at the initial step of the NRG calculation, i.e., based on the eigenvalues and operator matrices computed in the initialization script.

The NRG iteration proper than starts. For each NRG step, there is one block of log information. The header specifies the iteration number and the corresponding characteristic energy scale. The discretization parameters are listed, followed by statistics about the invariant subspace sizes (minimal, maximal, and a sum of third moments). There is then truncation data (`Emax` is the energy of the last state retained, `nrkept` the number of multiplets kept, and `nrkeptmult` is the total number of states kept taking into account the degeneracies). The next line contains the current estimate of the total energy of the system and the ground state energy shift at the current step (in the characteristic ω_N units). Next line contains summary about the truncation (number of multiplets kept and the corresponding ratio). The expectation values are then listed. The final line contains memory usage and timing statistics.

After the iteration is completed, the spectral function weights and the first four spectral moments are shown.

The line starting with `Total energy:` contains the final estimate for the total energy of the Wilson chain in absolute units.

The following line is only useful for models which undergo $S = 1/2$ single-channel Kondo effect: it contains the estimated of the Kondo temperature according to Wilson's definition, $(g\mu_B)^2\chi_{\rm imp}(T) = 0.07$.

Then there is some information about the calculation of density matrices, followed by a new NRG iteration where the DMNRG, CFS, and FDM spectral functions are computed.

Just before exiting, the NRG program shows a memory usage and timing reports, which are useful for estimating the numerical requirements of further calculations.

The data in `log2` is useful for monitoring the progress of the calculation: it only shows the current step number and errors, if they occur.

6.2 `td`, thermodynamic properties

The output file `td` contains thermodynamic properties of the system tabulated as a function of the temperature. The values of the temperature depend on the discretization scheme and the discretizatiopn parameters Λ and z, as well as on the parameter $\bar{\beta}$ (see `betabar`, Sec. 4.11). There are various schemes to increase smoothness of the results [4]. To obtain finer mesh, one can do z-averaging and combine the results (see the tool `tdavg`).

For all symmetry types, it is possible to compute the on-shell energy expectation value

$$\beta \langle H_N \rangle , \tag{35}$$

(column `<E>`), where H_N is the step-N NRG Hamiltonian, the expectation value of energy squared,

$$\beta^2 \left\langle H_N^2 \right\rangle , \tag{36}$$

(column `<E^2>`), the heat capacity

$$C/k_B = \beta^2 \Big(\left\langle H_N^2 \right\rangle - \langle H_N \rangle^2 \Big), \tag{37}$$

(column `C`), the free energy

$$F = -\ln(Z_N), \tag{38}$$

(column `F`), and the entropy

$$S/k_B = \beta \langle H_N \rangle + \ln(Z_N), \tag{39}$$

(column `S`).

Other quantities computed depend on the symmetry type. Only the expectation values of conserved quantum numbers (and arbitrary functions of these numbers) may be easily calculated.

For symmetry type `QS`, for example, one may compute the expectation values of total charge

$$\langle Q_{\rm total} \rangle , \tag{40}$$

(column `<Q>`), charge squared

$$\left\langle Q_{\rm total}^2 \right\rangle , \tag{41}$$

(column `<Q^2>`), and total spin squared

$$\left\langle S_{{\rm total},z}^2 \right\rangle = \frac{1}{3} \left\langle S_{\rm total}^2 \right\rangle , \tag{42}$$

(column `<Sz^2>`).

For symmetry type `QSZ`, one can in addition compute the expectation value of the total spin projection along the z axis

$$\langle S_{{\rm total},z} \rangle , \tag{43}$$

(column <Sz>).

Usually one wants to calculate the impurity contribution to the thermodynamic quantities. In such case, the best approach is to run a separate NRG calculation for a model without any impurities (such as model CLEAN), then subtracting the results. The tool tdavg can perform both the subtraction and data smoothing.

The output file td starts with a header file with some basic information about the NRG iteration (discretization parameters, truncation parameters, Wilson chain length, $\bar{\beta}$, etc.), followed by a line with column headers. The rest of the file is then space separated table with numerical values.

6.3 custom and customfdm, expectation values of singlet operators

The NRG code can compute the temperature dependence of the expectation values of arbitrary singlet (with respect to the dynamical group) operator using a similar approach as in the calculation of thermodynamic quantities. The temperature grid again depends on the discretization parameters and on $\bar{\beta}$. The expectation values are listed in columns in the output file custom. There is one row for each temperature. Again, it is possible to apply data smoothing (and z averaging) to obtain smoother temperature-dependence curves; one can use the tool tdavg for this purpose, too.

In addition, if the option fdmexpv is set to true, a futher output file customfdm is generate. It is similar to custom, but contains a single line with the results for the expectation value computed using the FDM NRG algorithm at chosen physical temperature T.

The operators to be computed are specified using the parameter ops.

6.3.1 customsq, expectation values of operators squared

The file customsq is analogous to custom

6.4 annotated.dat, eigenvalue spectra

The file annotated.dat contains the eigenvalue spectra for each NRG step. The different NRG steps are separated by empty lines. Each block contains one line for each eigenenergy, the quantum numbers of the eigenstate, and the total degeneracy of the state. In case of degeneracy of several multiples, they are displayed as a single line of the parameter dumpgroups is enabled, which is the default behavior.

The parameters which affect the output to this file are dumpannotated, dumpscales, dumpabs, dumpprecision, dumpgroups, and dumptol.

The contents of annotated.dat can be plotted in form of the eigenvalue flow diagrams which indicate the cross-overs between the different fixed points of the problem.

6.5 Spectral functions

The broadened spectral functions are saved to files with a suffix .dat. These are basically functions tabulated on a logarithmic mesh specified by the parameters broaden_max, broaden_min, broaden_min_ratio, and broaden_ratio. The broadening is described in section 4.23. The output precision is controlled by parameter precxy. The spectral functions to be computed are specified in parameters specs, specd, spect, and specb. The necessary operators need to be listed in ops.

The file names are composed of four parts:

1. Type of spectrum (b, corr, spec, spin).
2. Algorithm used (FT, DMNRG, CFS, FDM).
3. dens, which simply indicates that this is a spectral density.
4. $A - B$, where A and B are the operators appearing in the correlator.

If the parameter `reim` is set to `true`, there are three columns: frequency, spectral weight of the delta peak, and the "imaginary part" of the spectral weight. These imaginary parts need to be post-processed by performing a Kramers-Kronig transformation. See also section 4.59.

These broadened spectral functions are only generated if `broaden=true`.

If `savebins=true`, there will also be binary output files with a suffix `.dat` which contain the raw spectral weights in the form of bins. The format is a sequence of double precision floating point numbers, specifying the interval center and the spectral weight in the interval. If `reim=true`, the output consists of triplets instead of pairs.

For Matsubara spectra, the file contains three columns: the Matsubara frequency (without the imaginary part i), followed by real and imaginary parts of the Green's function at that point. See parameter `mats`.

6.6 Transport properties: conductance and higher moments

The NRG Ljubljana can directly compute the temperature-dependence of the transport integrals

$$I_i(T) = \int_{-\infty}^{+\infty} \left(-\frac{\mathrm{d}f}{\mathrm{d}\omega}\right) \omega^i A(\omega, T) \mathrm{d}\omega. \tag{44}$$

The zero-th integral is related to the conductance $G(T)$, while the first and second moment can be used to compute the (spin) thermopower and the heat conductivity. The method used is described in Refs. [44, 45].

The conductance is saved to a file prefixed by `gt_GT_dens`, the first moment to a file prefixed by `i1t_I1T_dens`, and the second moment to a file prefixed by `i2t_I2T_dens`.

The content of the files is a table of $\{T, I_i(T)\}$ pairs.

See also parameters `specgt`, `speci1t`, and `speci2t`, as well as `gtp`.

A very elegant way to compute the conductance for a general problem is to define an operator `hybopf`, which is a commutator between the hybridization part of the Hamiltinian and the annihilation operator on the zero-th site of the Wilson chain, $[H_{\mathrm{hyb}}, f_{0,\sigma}]$. Then one should compute the I_0 for this operator, multily the data in the output file `gt_GT_dens` by $\pi^2/2$ and the result is the conductance in units of the conductance quantum $G_0 = 2e^2/h$. This approach works for arbitrary model. (Not so) incidentally, the spectral function of the operator `hybopf` is the imaginary part of the T-matrix (up to numeric prefactors).

6.7 `cutoffs.dat`, truncation data

This file contains the number of states retained for each of the invariant subspaces. See parameter `trunc`.

7 Global operators

[46]

References

[1] K. G. Wilson. The renormalization group: Critical phenomena and the Kondo problem. *Rev. Mod. Phys.*, 47:773, 1975.

[2] H. R. Krishna-murthy, J. W. Wilkins, and K. G. Wilson. Renormalization-group approach to the Anderson model of dilute magnetic alloys. I. static properties for the symmetric case. *Phys. Rev. B*, 21:1003, 1980.

[3] H. R. Krishna-murthy, J. W. Wilkins, and K. G. Wilson. Renormalization-group approach to the Anderson model of dilute magnetic alloys. II. static properties for the asymmetric case. *Phys. Rev. B*, 21:1044, 1980.

[4] Ralf Bulla, Theo Costi, and Thomas Pruschke. The numerical renormalization group method for quantum impurity systems. *Rev. Mod. Phys.*, 80:395, 2008.

[5] H. O. Frota and L. N. Oliveira. Photoemission spectroscopy for the spin-degenerate anderson model. *Phys. Rev. B*, 33:7871, 1986.

[6] O. Sakai, Y. Shimizu, and T. Kasuya. Single-particle and magnetic excitation spectra of degenerate anderson model with finite f-f coulomb interaction. *J. Phys. Soc. Jpn.*, 58:3666, 1989.

[7] M. Yoshida, M. A. Whitaker, and L. N. Oliveira. Renormalization-group calculation of excitation properties for impurity models. *Phys. Rev. B*, 41:9403, 1990.

[8] O. Sakai and Y. Shimizu. Excitation spectra of two impurity anderson model. i. critical transition in the two magnetic impurity problem and the roles of the parity splitting. *J. Phys. Soc. Japan*, 61:2333, 1992.

[9] O. Sakai and Y. Shimizu. Excitation spectra of the two impurity anderson model. ii. interplay between kondo effect and the inter site interactions. *J. Phys. Soc. Japan*, 61:2348, 1992.

[10] O. Sakai, Y. Shimizu, and T. Kasuya. Excitation spectra of the impurity anderson model calculated by the numerical renornalization group. *Prog. theor. phys.*, 108:73, 1992.

[11] T. A. Costi and A. C. Hewson. Resistivity cross-over for the non-degenerate anderson model. *Phil. Mag. B*, 65:1165, 1992.

[12] T. A. Costi, A. C. Hewson, and V. Zlatić. Transport coefficients of the anderson model via the numerical renormalization group. *J. Phys.: Condens. Matter*, 6:2519, 1994.

[13] W. Izumida and O. Sakai. Two-impurity kondo effect in double-quantum-dot systems: Effect of interdot kinetic exchange coupling. *Phys. Rev. B*, 62:10260, 2000.

[14] T. A. Costi. Kondo effect in a magnetic field and the magnetoresistivity of kondo alloys. *Phys. Rev. Lett.*, 85:1504, 2000.

[15] T. A. Costi. Magnetotransport through a strongly interacting quantum dot. *Phys. Rev. B*, 64:241310(R), 2001.

[16] W. Izumida and O. Sakai. Kondo effect in quantum dot systems – numerical renormalization group study. *J. Phys. Soc. Japan*, 74:103, 2005.

[17] G. Zaránd, L. Borda, J. von Delft, and N. Andrei. Theory of inelastic scattering from magnetic impurities. *Phys. Rev. Lett.*, 93:107204, 2004.

[18] P. Mehta, N. Andrei, P. Coleman, L. Borda, and G. Zaránd. Regular and singular fermi-liquid fixed points in quantum impurity models. *Phys. Rev. B*, 72:014430, 2005.

[19] Walter Hofstetter. Generalized numerical renormalization group for dynamical quantities. *Phys. Rev. Lett.*, 85:1508, 2000.

[20] Rok Žitko and Janez Bonča. Enhanced conductance through side-coupled double quantum dots. *Phys. Rev. B*, 73:035332, 2006.

[21] A. I. Tóth, C. P. Moca, Ö. Legeza, and G. Zaránd. Density matrix numerical renormalization group for non-abelian symmetries. *Phys. Rev. B*, 78:245109, 2008.

[22] A Weichselbaum. Non-abelian symmetries in tensor networks: A quantum symmetry space approach. *Annals of Physics*, 327:2972, 2012.

[23] R. Bulla, A. C. Hewson, and Th. Pruschke. Numerical renormalization group calculation for the self-energy of the impurity anderson model. *J. Phys.: Condens. Matter*, 10:8365, 1998.

[24] W. C. Oliveira and L. N. Oliveira. Generalized numerical renormalization-group method to calculate the thermodynamical properties of impurities in metals. *Phys. Rev. B*, 49:11986, 1994.

[25] K. Chen and C. Jayaprakash. X-ray-edge singularities with nonconstant density of states: A renormalization-group approach. *Phys. Rev. B*, 52:14436, 1995.

[26] Kevin Ingersent. Behavior of magnetic impurities in gapless fermi systems. *Phys. Rev. B*, 54:11936, 1996.

[27] R. Bulla, Th. Pruschke, and A. C. Hewson. Anderson impurity in pseudo-gap fermi systems. *J. Phys.: Condens. Matter.*, 9:10463, 1997.

[28] V. L. Campo and L. N. Oliveira. Alternative discretization in the numerical renormalization group. *Phys. Rev. B*, 72:104432, 2005.

[29] Rok Žitko and Thomas Pruschke. Energy resolution and discretization artefacts in the numerical renormalization group. *Phys. Rev. B*, 79:085106, 2009.

[30] Rok Žitko. Adaptive logarithmic discretization for numerical renormalization group methods. *Comp. Phys. Comm.*, 180:1271, 2009.

[31] Koji Satori, Hiroyuki Shiba, Osamu Sakai, and Yukihiro Shimizu. Numerical renormalization group study of magnetic impurities in superconductors. *J. Phys. Soc. Japan*, 61:3239, 1992.

[32] Osamu Sakai, Yukihiro Shimizu, Hiroyuki Shiba, and Koji Satori. Numerical renormalization group study of magnetic impurities in supercoductors. ii. dynamical excitations spectra and spatial variation of the order parameter. *J. Phys. Soc. Japan*, 62:3181, 1993.

[33] Tomoki Yoshioka and Yoji Ohashi. Numerical renormalization group studies on single impurity anderson model in superconductivity: a unified treatment of magnetic, nonmagnetic impurities, and resonance scattering. *J. Phys. Soc. Japan*, 69:1812, 2000.

[34] R. Žitko. Quantitiative determination of the discretization and truncation errors in numerical renormalization-group calculations of spectral functions. *Phys. Rev. B*, 84:085142, 2011.

[35] Matthias Vojta, Ralf Bulla, Fabian Güttge, and Frithjof Anders. Mass-flow error in the numerical renormalization-group method and the critical behavior of the sub-ohmic spin-boson model. *Phys. Rev. B*, 81:075122, 2010.

[36] R. Žitko. Sneg – mathematica package for symbolic calculations with second-quantization-operator expressions. *Comp. Phys. Comm.*, 1982:2259, 2011.

[37] R. Bulla, T. A. Costi, and D. Vollhardt. Finite-temperature numerical renormalization group study of the mott transition. *Phys. Rev. B*, 64:045103, 2001.

[38] Rok Žitko, Robert Peters, and Thomas Pruschke. Splitting of the kondo resonance in anisotropic magnetic impurities on surfaces. *New J. Phys.*, 11:053003, 2009.

[39] Robert Peters, Thomas Pruschke, and Frithjof B. Anders. A numerical renormalization group approach to green's functions for quantum impurity models. *Phys. Rev. B*, 74:245114, 2006.

[40] F. B. Anders and A. Schiller. Real-time dynamics in quantum impurity systems: A time-dependent numerical renormalization group approach. *Phys. Rev. Lett.*, 95:196801, 2005.

[41] F. B. Anders and A. Schiller. Spin precession and real-time dynamics in the kondo model: Time-dependent numerical renormalization-group study. *Phys. Rev. B*, 74:245113, 2006.

[42] Andreas Weichselbaum and Jan von Delft. Sum-rule conserving spectral functions from the numerical renormalization group. *Phys. Rev. Lett.*, 99:076402, 2007.

[43] Rok Žitko. Numerical renormalization group calculations of ground-state energy: Application to correlation effects in the adsorption of magnetic impurities on metal surfaces. *Phys. Rev. B*, 79:233105, 2009.

[44] M. Yoshida, A. C. Seridonio, and L. N. Oliveira. Universal the zero-bias conductance for the single-electron transistor. *Phys. Rev. B*, 80:235317, 2009.

[45] A. C. Seridonio, M. Yoshida, and L. N. Oliveira. Universal zero-bias conductance through a quantum wide side-coupled to a quantum dot. *Phys. Rev. B*, 80:235318, 2009.

[46] R. Žitko and Thomas Pruschke. Many-particle effects in adsorbed magnetic atoms with easy-axis anisotropy: the case of fe on the cun/cu(100) surfaces. *New J. Phys.*, 12:063040, 2010.