PointGroupNRG Manual

Aitor Calvo-Fernández, acalvo049@ehu.eus

June 20, 2024

Contents

1 Overview

This manual describes how to use the `PointGroupNRG` code. Some context and theory are introduced at certain points, but it does not serve as a comprehensive guide to the NRG technique. For a comprehensive review of the NRG method, see Ref. [1]. For a description of the theory and implementation specific to `PointGroupNRG`, see Refs. [2] and [new arxiv], which also contain useful references specific to the applications of symmetry in the NRG.

This manual starts in Section 2 by describing the typical steps in the preparation and execution of an NRG calculation. Section 3 describes the format for the Clebsch-Gordan coefficients that have to be provided in order to use finite point or double group symmetries. Sections 4 and 5 deal with the main functions provided by the code and are organized by describing first the required arguments and then the optional ones, introducing the necessary information along the way. Section 4 covers the `compute_multiplets` function of the `PointGroupNRG.MultipletCalculator` submodule, which constitutes the first step in the setup of a model. Section 5 gives information about the `nrgfull` function of the `PointGroupNRG.NRGCalculator` submodule, which constructs the model and solves it using the NRG.

The functions `compute_multiplets` and `nrgfull` as they are defined in their respective submodules have more optional keyword arguments than those described here. The arguments not described here are in general related to features in development and changing their value might cause the program to crash.

2 Workflow

The number of steps required for an NRG calculation varies depending on the amount of setup information that is already available. Here we describe all the necessary steps for a full calculation starting from scratch for a model requiring Clebsch-Gordan coefficients for a point or double group. The workflow consists of three main steps:

1. Prepare the directory containing the Clebsch-Gordan coefficients following the format specificed in Section 3. These coefficients will then be read by the functions executed in the next steps. This step is not required for models with total angular momentum conservation, since the Clebsch-Gordan coefficients for the $SU(2)$ spin-orbital symmetry group are computed by the program.

2. Compute the multiplet states for the electronic degrees of freedom of the conduction band and, if necessary, for the impurity (see Section 5.2.2).

This is achieved with the `compute_multiplets` function described in Section 4, which calculates the multiplet states and stores them into an appropriate format. This step has to be performed only once for each irrep of the electronic degrees of freedom of a given model, so it is recommended to include it in an independent script separate from step 3.

3. Construct the model and perform the NRG calculation using the `nrgfull` function described in Section 5.2.2.

3 Clebsch-Gordan coefficients

Calculations for models with point or double group symmetries require the Clebsch-Gordan coefficients for the group given in the following format: The coefficients are organized according to the reduction of irrep products. If `A` and `B` are the user-given names of two irreps of the finite group G, then their decomposition is

$$\mathtt{A} \boxtimes \mathtt{B} = \oplus_{\mathtt{C}} \mathtt{L(C)C}, \tag{1}$$

where `C` are irreps of G and `L(C)` is the number of times `C` appears in the decomposition. On top of the names, the irreps of G must be arbitrarily numbered by the user, starting from 0. If `A`, `B` and `C` are the `a`-th, `b`-th and `c`-th irreps of G, respectively, then the Clebsch-Gordan coefficients of the irrep decomposition of `A`⊠`B` must be contained in a file called `axb_AxB.txt`. That file should be organized in paragraphs separated by blank lines, each paragrah containing the Clebsch-Gordan coefficients associated to the subspaces belonging to each irrep in the following format:

```
c C l
( 1 1 | 1 ) = <( A, 1 ; B, 1 | C, 1 )>
...
```

where `l`= 1,...,`L` labels the distinct subspaces belonging to `C` that arise in the decomposition of `A`⊠`B`, and `<( A, 1 ; B, 1 | C, 1 )>` is meant to be substituted by the value of the Clebsch-Gordan coefficient given in any format that Julia can parse into a complex number. The dots `...` have to be filled with the rest of the Clebsch-Gordan coefficients following the same format,

```
( gA gB | gC ) = <( A, gA ; B, gB | C, gC )
```

where `gA`, `gB` and `gC` are the partner numbers of the irreps `A`, `B` and `C`, respectively. Only the non-zero coefficients are necessary. To compare this with the notation in Refs. [2] and [new arxiv], substitute the irreps (partners) following the rule $\Gamma_A \leftrightarrow$`A` ($\gamma_A \leftrightarrow$`gA`), $\Gamma_B \leftrightarrow$`B` ($\gamma_B \leftrightarrow$`gB`) and $\Gamma_C \leftrightarrow$`C` ($\gamma_C \leftrightarrow$`gC`). The files for all the irrep combinations `A`⊠`B` with `a`≥`b` (to avoid redundancy) must be stored into a directory.

4 Multiplet calculation: `compute_multiplets`

The function `compute_multiplets` computes the many-body multiplet states arising from the combination of electrons in states belonging to the same one-electron irrep. If the impurity and conduction electrons occupy states belonging to irreps Γ_1 and Γ_2, for instance, with dimensions $\dim(\Gamma_1)$ and $\dim(\Gamma_2)$, respectively, then `compute_multiplets` has to be run once for Γ_1 and once for Γ_2. This is also the case if there are several multiplet subspaces belonging to Γ_1, for example, because the resulting states are distinguished only by an outer multiplicity label that is taken care of by `nrgfull` (see Section 5). The function is defined as itemize

```
function compute_multiplets(
    symmetry::String ;
    irrep::SF="" ,
    multiplets_path::String="multiplets" ,
    clebschgordan_path::String=""
) where {SF<:Union{String,Float64}}
```

4.1 Necessary arguments

4.1.1 `symmetry::String`

Determines the type of symmetry. The accepted values are:

- `"pointspin"` or `"PS"`: Uses the symmetry type

$$G_{\mathrm{UPS}} = U(1)_{\mathrm{C}} \otimes P_{\mathrm{O}} \otimes SU(2)_{\mathrm{S}}, \tag{2}$$

  which is the outer direct product of unitary charge symmetry $U(1)_{\mathrm{C}}$ (particle conservation), the orbital point group P_{O}, and spin rotation symmetry $SU(2)_{\mathrm{S}}$ (conservation of total spin).

- `"doublegroup"` or `"D"`: Uses the symmetry type

$$G_{\mathrm{UOS}} = U(1)_{\mathrm{C}} \otimes D_{\mathrm{OS}}, \tag{3}$$

  which is the outer direct product of unitary charge symmetry $U(1)_{\mathrm{C}}$ (particle conservation) and the spin-orbital double group D_{OS}.

- `"totalangularmomentum"` or `"J"`: Uses the symmetry type

$$G_{\mathrm{UJ}} = U(1)_{\mathrm{C}} \otimes SU(2)_{\mathrm{OS}}, \tag{4}$$

  which is the outer direct product of unitary charge symmetry $U(1)_{\mathrm{C}}$ (particle conservation) and spin-orbital rotational invariance (conservation of total angular momentum).

4.2 Optional keyword arguments

4.2.1 `irrep::SF=""`

One-electron irrep for which to compute the multiplet states. The specification of the irrep changes depending on the value of `symmetry` (4.1.1):

- If `symmetry=="PS"` or `symmetry=="pointspin"`, then `irrep` must be a `String` and is the name of the name of the point group irrep as it appears in the Clebsch-Gordan coefficient files (see 3).
- If `symmetry=="D"` or `symmetry=="doublegroup"`, then `irrep` must be a `String` and is the name of the name of the double group irrep as it appears in the Clebsch-Gordan coefficient files (see 3).
- If `symmetry=="J"` or `symmetry="totalangularmomentum"`, then `irrep` must be a `Float` and is the value of the total angular momentum J of the electrons, where $\mathbf{J} = \mathbf{L} + \mathbf{S}$.

4.2.2 `clebschgordan_path::String=""`

Absolute or relative path of the directory where the Clebsch-Gordan coefficients are stored (see 3). It must be provided only if `symmetry=="D"`, `symmetry=="doublegroup"`, `symmetry=="PS"`, `symmetry=="pointspin"`, as the Clebsch-Gordan coefficients of $SU(2)$ are computed by the program.

4.2.3 `multiplets_path::String="multiplets"`

Absolute or relative path of the parent directory where the directories containing the multiplet states for the chosen irreps (see 4.2.1) will be stored. It must be the same for irreps intented to be used in the same NRG calculation. If it does not already exist, it is automatically created.

5 NRG calculation: `nrgfull`

The `nrgfull` function constructs the model from the input and the multiplet states computed by `compute_multiplets` (see 4), and it solves it using the NRG method. It can be used to calculate thermodynamic functions (see 5.2.2), spectral functions (see 5.2.8), and/or thermodynamic weights (see)

```
function nrgfull(
    symmetry::String ,
    label::String ,
    L::Float64 ,
    iterations::Int64 ,
    cutoff::IF ,
    shell_config::Dict{SF,Int64} ,
    tunneling::Dict{SF,Matrix{ComplexF64}} ;
```

```
    multiplets_dir::String="multiplets" ,
    calculation::String="IMP" ,
    impurity_config::Dict{SF,Int64}=Dict{SF,Int64}() ,
    onsite::Dict{SF,Vector{ComplexF64}}=Dict{SF,Vector{ComplexF64}}() ,
    interaction::{Tuple{String,Float64},Matrix{ComplexF64}}=Dict{ Tuple{String,Float64}
    spectrum::Dict{ClearIrrep,Vector{Float64}}=Dict{ClearIrrep,Vector{Float64}}() ,
    lehmann_iaj::Dict{ClearTripleG,Array{ComplexF64,4}}=Dict{ClearTripleG,Array{ComplexF
    # run only until a certain point for partial information
    # - 2-particle multiplets
    # - impurity spectrum
    # - impurity-shell spectrum
    until::String = "",
    cg_o_dir::String="" ,
    identityrep::String="" ,
    z::Float64=0.0 ,
    max_SJ2::Int64=10 ,
    channels_dos::Dict{ String , Vector{Function} }=Dict{ String , Vector{Function} }()
    mine::Float64=0.0 ,
    betabar::Float64=1.0 ,
    spectral::Bool=false ,
    broadening_distribution::String="loggaussian" ,
    spectral_broadening::Float64=0.5 ,
    K_factor::Float64=2.0 ,
    compute_impurity_projections::Bool=false ,
    band_width::Float64=1.0 ,
    print_spectrum_levels::Int64=0 ,
) where {SF<:Union{String,Float64},IF<:Union{Int64,Float64}}
```

5.1 Necessary arguments

5.1.1 `symmetry::String`

See 4.1.1.

5.1.2 `label::String`

Label or name given to the system. It will appear in the names of the output files: thermodynamic functions, spectral functions and impurity projections. Output files are ovewritten by subsequent calculations with the same value of `label`.

5.1.3 `L::Float64`

Discretization parameter Λ. See Ref. [1]. Lower values provide more accurate results, larger values result in convergence for lower cutoffs (see 5.1.5). Large values of Λ usually produce low resolution thermodynamic and spectral functions that can contain spurious oscillations. Oscillations can be removed in

many cases by averaging over even and odd step results, which is done automatically, resulting in improved spectral functions and often completely satisfactory thermodynamic functions. It is possible to go further by averaging over various discretizations (see 6). In general, it is recommended to use low values $\Lambda \in [2, 3]$ for the spectral functions, while for thermodynamic functions values as large as $\Lambda = 10$ often give satisfactory results.

5.1.4 `iterations::Int64`

Number of iterations in the NRG calculation. Lower temperatures (for thermodynamic function calculations) and lower energies (for spectral function calculations) are reached with more iterations.

5.1.5 `cutoff:IF`

Cutoff imposed on the multiplets: after each iterations, multiplets above the cutoff are discarded. The type of cutoff varies depending on the type of the input `cutoff`:

- If `cutoff` is an `Int`, *e.g.* `300`, then it specifies the number of multiplets kept at each iteration.
- If `cutoff` is a `Float`, *e.g.* `7.0`, then it specifies the cutoff energy: multiplets with energies larger than that are discarded.

In both cases, the program tries to avoid breaking accidental degeneracies by checking for a small energy window above the imposed cutoff and keeping also those states.

5.1.6 `shell_config::Dict{SF,Matrix{ComplexF64}}`

It specifies the configuration of the conduction channels. It has the structure

```
Dict(
    G1 => n1,
    G2 => n2,
    ...
)
```

where `Gi` are the irreps in the format described in 4.2.1 and `ni` specify the number of channels with the symmetry defined by the irrep.

5.1.7 `tunneling::Dict{SF,Matrix{ComplexF64}}`

It specifies the tunneling amplitudes $V(\Gamma_a)_{r_a r_b}$ as a dictionary with the format

```
Dict(
    G1 => amplitudes1,
    G2 => amplitudes2,
```

```
    ...
)
```

where `Gi` are the irreps Γ_a in the format described in 4.2.1 and `amplitudesi` are the amplitudes given as a `MatrixComplexF64` with indices r_a, r_b.

5.2 Optional keyword arguments

5.2.1 `multiplets_dir::String="multiplets"`

5.2.2 `calculation::String="IMP"`

5.2.3 `impurity_config::Dict{SF,Int64}`

5.2.4 `onsite::Dict{SF,Vector{ComplexF64}}=Dict{SF,Vector{ComplexF64}}()`

5.2.5 `interaction::Dict{SF,Matrix{ComplexF64}}=Dict{SF,Matrix{ComplexF64}}`

5.2.6 `spectrum::Dict{Tuple{Int64,String,Float64},Vector{Float64}}=Dict{...}()`

5.2.7 `lehmann_iaj`

5.2.8 **spectral::Bool=false**

5.2.9 **impurity_projections**

5.2.10 `multiplets_path::String="multiplets"`

6 Averaging over various discretizations