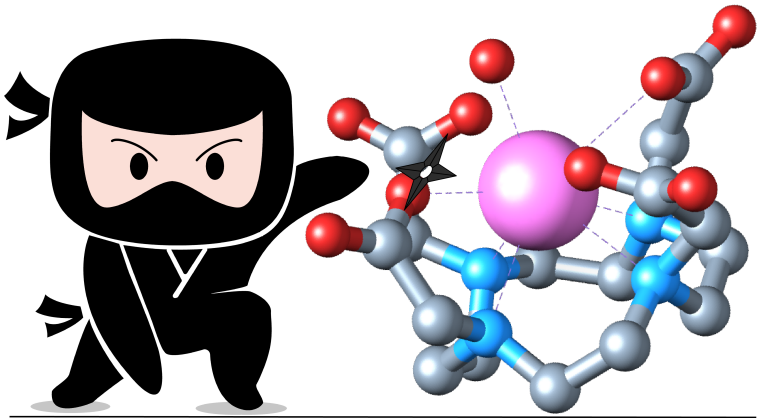

NJA-CFS: Not Just Another - Crystal Field Software

Letizia Fiorucci[1,2] and Enrico Ravera[1,2,3]

[1]CERM and Department of Chemistry "Ugo Schiff", University of Florence, Sesto Fiorentino, 50019, Italy
[2]Consorzio Interuniversitario Risonanze Magnetiche di Metalloproteine, Sesto Fiorentino, 50019, Italy
[3]Florence Data Science, University of Florence, Firenze, 50019, Italy

This documentation is currently under development and may be incomplete. Some sections or features may still be missing or subject to change. For more detailed examples and explanations, please refer to the related publication: https://doi.org/10.1002/jcc.70063

1 Introduction

The application of crystal-field (CF) or ligand-field (LF) effective Hamiltonians is a well-established practice in the study of the magnetic properties of inorganic complexes. The history of the crystal field formalism dates back to the pioneering work of Bethe [1]. Since then, various parametrization schemes have been developed to describe the splitting of the free ion terms of a metal center in the presence of surrounding ligands. As a result, many

CF/LF programs have been made available in the literature, such as BonnMag [2], SIMPRE [3, 4, 5], CONDON [6, 7], f_electron (available at link: https://github.com/octoYot/f_electron), and PyCrystalField [8].

In this context, we introduce NJA-CFS (Not Just Another-Crystal Field Software), with a nod to the fact that while it may appear to be just another CF software, its underlying purpose sets it apart. NJA-CFS was designed to meet the needs of both first-time users of crystal field theory and those who require a high degree of flexibility in the choice of crystal field parameters (CFPs) formalisms (*vide infra*). Its aim is to provide an intuitive, user-friendly structure that bridges accessibility and the rigor demanded by advanced users.

NJA-CFS serves as a comprehensive toolkit for manipulating crystal field parameters from various sources and exploring how these manipulations influence magnetic properties. Its Python-based foundation ensures excellent readability and visualization capabilities, while also allowing seamless integration with external routines and workflows.

This tutorial will guide you through the program's application, alongside a detailed theoretical explanation of its functions and references to the relevant literature.

In this guide, the sections indicated as:

```
# example code
```

represent code listings, while those reported as:

```
$ python main.py
Output
```

represent code printouts in the execution terminal.

2 Requirements

The code folder contains 4 Python scripts: `nja_cfs_vX.py`, which contains all the program functions organized in classes, `nja_cfs_red.py` which is a copy of the previous one with the difference that it can use `numba` algorithms to speed up the computation of some of the functions, `cryst_dat.py` is a collection of REPULSION angles (it is used for grid searches) and `test_nja.py` contains all the examples and test functions to prove the correct functioning of program and its applicability. All the files needed for the latter test file are saved in the `test/` folder. The `tables/` folder contains text files with coefficient tables used by the program.

NJA-CFS is a collection of functions, not an actual installable program (for the moment), therefore in order to use it you need to run the code directly from the NJA folder.

This program has been developed in `Python` (version 3.12) on a Linux machine (OS: Ubuntu 22.04 LTS).

The mandatory dependencies that you need to install in order to run NJA-CFS are: `numpy` (version 2.0.2)[9], `matplotlib` (version 3.9.2)[10] and `scipy` (version 1.14.1)[11]. The versions are mainly indicative, it could also work with different versions of the above-mentioned packages. Additional dependencies include: `numba` (version 0.60.0)[12] and `sympy` (version 1.13.3).

One way to set up the proper environment to run the codes is through the use of Miniconda, available at https://docs.anaconda.com/miniconda/. Miniconda is a free, lightweight

version of Anaconda that installs only Python and the conda package manager. It is a good option if you only need Python and do not need all the scientific packages that come with Anaconda. Once installed, it can be activated using:

```
$ conda activate
```

the confirmation of the correct activation is ensured by the presence of `(base)` in the terminal line:

```
(base) $
```

The mentioned packages can be installed with the command:

```
(base) $ conda install <package_name>
```

`numba` and `sympy` could be in conflict with the newest versions of Python and other packages. This is the reason why we kept them optional.

3 List of functions and program structure

The program is structured into 6 main `class`:

- `Wigner_coeff()`, it is a wrapper around the routines for the computation of 3j- and 6j-symbols and Wigner D matrices (in Euler angles and in quaternions),
- `CFP()`, it is responsible for the handling of cfps, i.e. it computes the almost-closed-shell version of the coefficients and selects the right coefficients for each matrix element,
- `RME()`, it computes the reduced matrix elements of $U^{(k)}$ and $V^{(1k)}$, using the cfps from `CFP()`,
- `Hamiltonian()`, it collects the functions for the computation of the effective Hamiltonian contributions,
- `calculation()`, it presents the routines for the construction of the Hamiltonian matrix, basis set reduction, and wavefunction optimization.
- `Magnetics()`, it collects the functions for the computation of the magnetic properties, i.e. effective g-matrix, susceptibility tensor, magnetization vector, and susceptibility and magnetization scalar fields.

Here follows a more details description of the above-mentioned `classes`, plus the description of all the functions included therein and some additional important routines.

Wigner_coeff()

This class is a wrapper around the functions to compute angular momentum coupling coefficients and Wigner D matrices for spherical harmonics rotations.

References:

1. Boca, R., "Theoretical Foundations of Molecular Magnetism" (1999) [13]

Wigner_coeff.threej_symbol() Compute the 3j-symbol using the Racah formula.

This function takes a 2D array as input, representing a 3j-symbol in the form ([j1, j2, j3],[j4, j5, j6]) and calculates the 3j-symbol using the Racah's formula (p. 52 Ch. 1 from [13]).

Parameters:

– matrix (numpy.ndarray): A 2D array representing a 6j-symbol.

Returns:

– result (float): The calculated 6j-symbol.

Wigner_coeff.sixj_symbol() Compute the 6j-symbol using the Racah formula.

This function takes a 2D array as input, representing a 6j-symbol in the form ([j1, j2, j3],[j4, j5, j6]) and calculates the 6j-symbol using the Racah's formula (p. 57 Ch. 1 from [13]).

Parameters:

– matrix (numpy.ndarray): A 2D array representing a 6j-symbol.

Returns:

– result (float): The calculated 6j-symbol.

Wigner_coeff.Wigner_Dmatrix() Compute an element of the Wigner D-matrix ($D^l_{m1,m}$).

This function calculates an element of the Wigner D-matrix defined in terms of Euler angles.

Parameters:

– l (int): The one-electron orbital quantum number.
– m1, m (int): Magnetic quantum numbers.
– α, β, γ (float): The three Euler angle for the ZYZ (active) rotation, in radians.

Returns:

– result (complex): The calculated element of the Wigner D-matrix.

Wigner_coeff.Wigner_Dmatrix_quat_complete() Compute the Wigner D-matrix using quaternions: R = R1 1 + Rx x + Ry y + Rz z.

This function calculates the Wigner D-matrix using quaternions, which are a more efficient way to represent rotations than Euler angles. The quaternion in input does not need to be normalized.

Parameters:

– l (int): The one-electron orbital quantum number.
– R (numpy.ndarray): The quaternion representing the rotation.
– bin (float, optional): The tolerance size for the calculation. Default is 1e-8.
– dict (dict, optional): Dictionary from tables. Default is None.
– coeff (numpy.ndarray, optional): The coefficients for the calculation from table. Default is None.

Returns:

– D (numpy.ndarray): The calculated Wigner D-matrix as a complex numpy array.

CFP()

Handles the coefficients of fractional parentage (cfp) for a given electron configuration.

References:

1. Nielson - Koster in "Spectroscopic Coefficients for the p^n, d^n and f^n Configurations" (1963) [14]

Attributes:

- dic_LS_inv_almost (dict): A dictionary of labels, with keys and values reversed with respect to dic_LS, hence: <state label>: '2S:L:2J:2M:sen:count', where <state label> comes from terms_labels().
- l (int): The one-electron orbital quantum number.
- n (int): The number of electrons in the almost-closed-shell configuration.
- N (int): The number of electrons in the configuration.
- closed (bool): A flag indicating whether the configuration is almost-closed-shell or not.

CFP.__init__() Initializes the CFP object with the given configuration, CFP dictionary, and optional LS-coupling scheme dictionary.

The configuration is represented by a string of the form 'l^x', where 'l' is orbital momentum quantum number (represented as 'd' for l=2 and 'f' for l=3), and 'x' is the number of electrons in the configuration.

Parameters:

- conf (str): The electron configuration.
- dic_cfp (dict): A dictionary containing the coefficients of fractional parentage (CFP) for the configuration. It is a dictionary with two keys, where the first one is the son state and the second indicates the parent state. The states are indicated using the free ion terms from terms_labels().
- dic_LS_inv_almost (dict, optional): A dictionary of labels, with keys and values reversed with respect to dic_LS, hence: <state label>: '2S:L:2J:2M:sen:count', where <state label> comes from terms_labels().

CFP.cfp() Returns the cfps for a given state, identified by its L, S quantum number and the name of the state.

Parameters:

- L (int): Orbital angular momentum quantum number
- S (float): Spin angular momentum quantum number
- name (str): The name of the state (from terms_labels()).

Returns:

- cfp_list (numpy.ndarray): The list of CFPs for the given state.

RME()

Docstring: This class is used to perform a RME calculation for a given electron configuration. The electron configuration is represented by a string of the form 'l^x', where 'l' is the azimuthal

quantum number (represented as 'd' for l=2 and 'f' for l=3), and 'x' is the number of electrons in the configuration.

References:

1. E. Konig "Ligand Field Energy Diagrams" (eq 2.85,2.87) [15]
2. R. Boca, "Theoretical foundations of molecular magnetism" (Ch 8, p 516) [13]

Attributes:

- v, L, S, v1, L1, S1 (float): The quantum numbers for the state.
- label1, label2 (str): The labels for the states.
- dic_LS (dict): A dictionary for the LS-coupling scheme, with elements of the type '2S:L:2J:2M:sen:count': <state label> as N. and K. (from terms_labels()).
- dic_LS_inv_almost (dict): A dictionary of labels, with keys and values reversed with respect to dic_LS, hence: <state label>: '2S:L:2J:2M:sen:count', where <state label> comes from terms_labels().
- s (float): The one-electron spin quantum number.
- l (int): The one-electron orbital quantum number.
- n (int): The number of electrons in the almost-closed-shell configuration.
- N (int): The number of electrons in the configuration.
- cfp (CFP): A CFP object for the configuration.
- closed (bool): A flag indicating whether the configuration is almost-closed-shell or not.

RME.__init__() Initializes the RME object with the given state, configuration, CFP dictionary, labels, and LS-coupling scheme dictionaries.

The state is represented by a list of the form [v, L, S, v1, L1, S1], where 'v', 'L', and 'S' are the quantum numbers for the initial state and 'v1', 'L1', and 'S1' are the quantum numbers for the final state. The configuration is represented by a string of the form 'nl^x', where 'n' is the principal quantum number, 'l' is the azimuthal quantum number (represented as 'd' for l=2 and 'f' for l=3), and 'x' is the number of electrons in the configuration.

Parameters:

- state (list): The quantum numbers for the state.
- conf (str): The electron configuration.
- dic_cfp (dict): A dictionary containing the coefficients of fractional parentage (CFP) for the configuration. It is a dictionary with two keys, where the first one is the son state and the second indicates the parent state. The states are indicated using the free ion terms from terms_labels().
- labels (list): The labels for the states from terms_labels().
- dic_LS (dict): A dictionary for the LS-coupling scheme, with elements of the type '2S:L:2J:2M:sen:count': <state label> as N. and K. (from terms_labels()).
- dic_LS_inv_almost (dict): A dictionary of labels, with keys and values reversed with respect to dic_LS, hence: <state label>: '2S:L:2J:2M:sen:count', where <state label> comes from terms_labels().

RME.Uk() Calculate the Uk coefficient for a given set of quantum numbers and labels.

This method calculates the Uk coefficient, which is used in the calculation of reduced matrix elements (RMEs) in atomic physics. The quantum numbers and labels for the initial and final states are either provided as arguments or taken from the RME object itself. The calculation involves summing over the coefficients of fractional parentage (CFP) for the initial and final states and the 6-j symbol of the associated angular momenta.

Parameters:

- k (int): The rank of the tensor operator.

Returns:

- Uk (float): The calculated Uk coefficient.

RME.V1k() Calculate the V1k coefficient for a given rank of the tensor operator.

This method calculates the V1k coefficient, which is used in the calculation of reduced matrix elements (RMEs) in atomic physics. The rank of the tensor operator is provided as an argument. The calculation involves summing over the coefficients of fractional parentage (CFP) for the initial and final states and the 6-j symbols of the associated angular momenta.

Parameters:

- k (int, optional): The rank of the tensor operator. Default is 1.

Returns:

- V1k (float): The calculated V1k coefficient.

Hamiltonian()

This class presents the routines for the computation of the Hamiltonian matrix element for the given integral.

References:

1. R. Boca, "A handbook of magnetochemical formulae". Elsevier, 2012.[16]
2. R. Boca, "theoretical foundations of molecular magnetism" 1999.[13]
3. E. Konig "Ligand Field Energy Diagrams" 2013.[17]
4. C. Goerller-Walrand, K. Binnemans, Handbook of Physics & Chemistry of Rare Earths, Vol 23, Ch 155, 1996.[18]

Attributes:

- v, L, S, v1, L1, S1 (float): The quantum numbers for the state.
- label1, label2 (str): The labels for the states.
- dic_LS (dict): A dictionary for the LS-coupling scheme, with elements of the type '2S:L:2J:2M:sen:count': <state label> as N. and K. (from terms_labels()).
- dic_LS_inv_almost (dict): A dictionary of labels, with keys and values reversed with respect to dic_LS, hence: <state label>: '2S:L:2J:2M:sen:count', where <state label> comes from terms_labels().
- s (float): The spin quantum number.
- l (int): The one-electron orbital quantum number.
- (int): The number of electrons in the almost-closed-shell configuration.

- N (int): The number of electrons in the configuration.
- cfp (CFP): A CFP object for the configuration. closed (bool): A flag indicating whether the configuration is almost-closed-shell or not.

Hamiltonian.__init__() Initializes the Hamiltonian object with the given state, configuration, CFP dictionary, labels, and LS-coupling scheme dictionaries.

The state is represented by a list of the form [v, L, S, v1, L1, S1, J, MJ, J1, MJ1], where 'v', 'L', 'S', 'J' and 'MJ' are the quantum numbers for the initial state and 'v1', 'L1', 'S1', 'J1' and 'MJ1' are the quantum numbers for the final state. The configuration is represented by a string of the form 'nl^x', where 'n' is the principal quantum number, 'l' is the azimuthal quantum number (represented as 'd' for l=2 and 'f' for l=3), and 'x' is the number of electrons in the configuration.

Parameters:

- state (list): The quantum numbers for the state.
- labels (list): The labels for the states.
- conf (str): The electron configuration.
- dic_cfp (dict): A dictionary containing the coefficients of fractional parentage for the configuration. It is a dictionary with two keys, where the first one is the son state and the second indicates the parent state. The states are indicated using the free ion terms from terms_labels().
- dic_LS (dict): A dictionary for the LS-coupling scheme, with elements of the type '2S:L:2J:2M:sen:count': <state label> as N. and K. (from terms_labels()).
- dic_LS_inv_almost (dict): A dictionary of labels, with keys and values reversed with respect to dic_LS, hence: <state label>: '2S:L:2J:2M:sen:count', where <state label> comes from terms_labels().

Hamiltonian.electrostatic_int() Computes electron repulsion integrals <self.label1 | Hee | self.label2> for the given basis set. Equations are taken from Boca 1999, "theoretical fundations of molecular magnetism" (Ch 8, p 518) (valid only for l2 conf) For the d^n configurations the equations are reported in Boca2012 (p 145 eq 4.66-4.69)

Parameters:

- basis (numpy.ndarray): The basis set for the calculation.
- F0, F2, F4, F6 (float, optional): The F0, F2, F4, and F6 are the Slater-Condon parameters in cm^{-1}. F0 is irrelevant.
- evaluation (bool, optional): A flag indicating whether the parameters are symbolic or numerical. Default is True.
- tab_ee (dict, optional): A dictionary containing the electron-electron integrals. Default is None.

Returns:

- integral (float): The calculated electron repulsion integral

Hamiltonian.SO_coupling() Computes SOC integrals <self.label1 | Hso | self.label2>. Equations are taken from Konig & Kremer (Ch 2, p 11, eq 2.82)

Parameters:

- zeta (float): The SOC parameter in cm^{-1}.
- k (int, optional): The orbital reduction factor. Default is 1.
- evaluation (bool, optional): A flag indicating whether the parameters are symbolic or numerical. Default is True.

Returns:

- integral (float): The calculated SOC integral.

Hamiltonian.LF_contribution() Computes the LF/CF contribution to the Hamiltonian matrix element <self.label1 | HCF/LF | self.label2> using Bkq in Wybourne formalism. Equations are taken from C. Goerller-Walrand, K. Binnemans, Handbook of Physics & Chemistry of Rare Earths, Vol 23, Ch 155, (1996).

Parameters:

- dic_kq (dict): A dictionary containing the Bkq coefficients in cm^{-1}, e.g. dic_bkq = '2':'-1':0.0 where k=2 and q=-1. The complete (complex) coefficient corresponds to Bkq + iBk-q.
- evaluation (bool, optional): A flag indicating whether the parameters are symbolic or numerical. Default is True.

Returns:

- result (complex): The calculated LF/CF contribution to the Hamiltonian matrix element.

Hamiltonian.Zeeman() Computes the Zeeman contribution to the Hamiltonian matrix element <self.label1 | HZeeman | self.label2> using the magnetic field. Equations are taken from Boca2012, "A handbook of magnetochemical formulae" (p 588).

Parameters:

- field (numpy.ndarray): The magnetic field vector in T.
- k (int, optional): The rank of the tensor operator. Default is 1.
- evaluation (bool, optional): A flag indicating whether the parameters are symbolic or numerical. Default is True.
- MM (bool, optional): A flag indicating whether to return the L1 and S1 contributions. Default is False.

Returns:

- int (complex): The calculated Zeeman contribution to the Hamiltonian matrix element.

calculation()

This is the main class of the code. Here the Hamiltonian matrix is computed and diagonalized. Additionally, in this class the basis set can be reduced and the wavefunctions optimized.

Attributes:

- conf (str): The electron configuration.
- l (int): The one-electron orbital quantum number.

- n (int): The number of electrons in the almost-closed-shell configuration.
- N (int): The number of electrons in the configuration.
- closed (bool): A flag indicating whether the configuration is almost-closed-shell or not.
- basis (numpy.ndarray): The complete basis set for the configuration.
- dic_LS (dict): A dictionary for the LS-coupling scheme, with elements of the type '2S:L:2J:2M:sen:count': <state label> as N. and K. (from terms_labels()).
- basis_l (numpy.ndarray): The labels for the states, 2S+1 L (J).
- basis_l_JM (numpy.ndarray): The labels for the states (with MJ), 2S+1 L (J) MJ.
- microst (int): The number of microstates.
- ground (bool): A flag indicating whether the basis set includes only the ground multiplet.
- dic_cfp (dict): A dictionary containing the coefficients of fractional parentage (CFP) for the configuration. It is a dictionary with two keys, where the first one is the son state and the second indicates the parent state. The states are indicated using the free ion terms from terms_labels().
- tables (dict): A dictionary containing the calculated RMEs.
- dic_LS_almost (dict): An inverse dictionary for the LS-coupling scheme for the almost-closed-shell configuration.
- dic_ee (dict): A dictionary containing the electron-electron integrals. It is a dictionary with two keys, where the first one is the son state and the second indicates the parent state. The states are indicated using the free ion terms from terms_labels().

calculation.__init__() Initializes the calculation object through the definition of the type of configuration (almost closed shell or not), the basis set and additional dictionaries. Additionally, it defines whether the RMEs are computed explicitly or read from tables (the computation of the RMEs will result in an increase of the computational time).

Parameters:

- conf (str): The electron configuration.
- ground_only (bool, optional): A flag indicating whether the basis set includes only the ground multiplet. Default is False.
- TAB (bool, optional): A flag indicating whether the RMEs are calculated or read from tables. Default is False.
- wordy (bool, optional): A flag indicating whether to print the output. Default is True.

calculation.ground_state_calc() Reduces the basis set to the ground multiplet.

Parameters:

- ground (str, optional): The ground multiplet. Default is None.

Returns:

- basis_red (numpy.ndarray): The reduced basis set for the ground multiplet.
- dic_LS_red (dict): A dictionary for the LS-coupling scheme for the ground multiplet.
- basis_l_red (numpy.ndarray): The labels for the states for the ground multiplet (hence, only one type).
- basis_l_JM_red (numpy.ndarray): The labels for the states (with MJ) for the ground multiplet.

calculation.Tables() Reads the RMEs from tables or computes them explicitly.

Parameters:

- TAB (bool): A flag indicating whether the RMEs are calculated or read from tables.
- stringa (str): The corresponding configuration.
- conf (str): The electron configuration.

Returns:

- dic_cfp (dict): A dictionary containing the coefficients of fractional parentage (CFP) for the configuration. It is a dictionary with two keys, where the first one is the son state and the second indicates the parent state. The states are indicated using the free ion terms from terms_labels().
- tables (dict): A dictionary containing the calculated RMEs.
- dic_LS_almost (dict): An inverse dictionary for the LS-coupling scheme for the almost-closed-shell configuration.
- dic_ee (dict): A dictionary containing the electron-electron integrals. It is a dictionary with two keys, where the first one is the son state and the second indicates the parent state. The states are indicated using the free ion terms from terms_labels().

calculation.reduce_basis() Reduces the basis set according to the spin multiplicity.
1. for dN configurations the Hund's rules are applied
2. for fN configurations the weak field/High Spin situation is considered.
The default parameters are taken from Ma, C. G., Brik, M. G., Li, Q. X., & Tian, Y. (2014) Journal of alloys and compounds, 599, 93-101 [19].

Parameters:

- conf (str): The electron configuration.
- roots (list, optional): A list containing the selected multiplicities as list of (sum(2*L+1), 2*S+1) pairs.
- dic (dict, optional): A dictionary containing the parameters for the free ion. Default is None.
- contributes (list, optional): A list containing the contributions to the matrix. Default is None.

calculation.MatrixH() This is the core of the calculation. This functions converts the different CFPs formalism in the Wybourne one, it optimizes the wavefunction composition, calls the function that builts the Hamiltonian and computes the wavefunctions composition.

Parameters:

- elem (list): The contributions to the Hamiltonian matrix among: 'Hee'=interelectronic repulsion, 'Hso'=spin-orbit coupling, 'Hcf'=crystal-field/ligand-field contribution, 'Hz'=Zeeman splitting.
- F0 (float, optional): The F0 Slater-Condon parameter in cm^{-1}. Default is 0.
- F2 (float, optional): The F2 Slater-Condon parameter in cm^{-1}. Default is 0.
- F4 (float, optional): The F4 Slater-Condon parameter in cm^{-1}. Default is 0.
- F6 (float, optional): The F6 Slater-Condon parameter in cm^{-1}. Default is 0.
- zeta (float, optional): The spin-orbit coupling parameter in cm^{-1}. Default is 0.

- k (float, optional): The orbital reduction factor. Default is 1.
- dic_V (dict, optional): A dictionary containing one-electron ligand field matrix elements in cm^{-1}. Default is None. dic_V = '11':0.0, '12':0, ..., with key='ij', with 1<=i<=2*l+1 and j<=i.
- dic_bkq (dict, optional): A dictionary containing the crystal field parameters in Wybourne formalism in cm^{-1}. Default is None. dic_bkq = 'k':'q':0.0, '-q':0.0, ..., with k=2,4,..2*l and q=-k,-k+1,...k-1,k.
- dic_AOM (dict, optional): A dictionary containing the crystal field parameters in AOM formalism in cm^{-1}. Default is None. dic_AOM = '<ligand name>:[e_σ, e_{π_c}, e_{π_s}, θ, ϕ, χ], where the angles are in degrees.
- PCM (list, optional): A list for the definition of the PCM for crystal field calculation. Default is None. The first element of the list is the array dtype=object where each row is: '<ligand name>', x, y, z, charge. The charge is expressed as fraction of electronic charge. The coordinates are in Angstrom. The metal center is assumed in (0,0,0). The second element in the list is a boolean that defines if the coordinates are expressed as Cartesian (False) or spherical (True). The third element is again a boolean and defines if the Sternheimer shielding parameters are used (True) or not (False).
- field (list, optional): The magnetic field vector in Tesla. Default is [0.,0.,0.].
- cfp_angles (list, optional): A list containing the Euler angles or quaternions for the rotation of the crystal field parameters. Default is None.
- wordy (bool, optional): A flag indicating whether to print the output. Default is False.
- Orth (bool, optional): A flag indicating whether to perform the orthogonality check. Default is False.
- Norm (bool, optional): A flag indicating whether to perform the normalization check. Default is False.
- eig_opt (bool, optional): A flag indicating whether to optimize the wavefunction. Default is False.
- ground_proj (bool, optional): A flag indicating whether to wavefunctions composition is computed or not. Default is False.
- return_proj (bool, optional): A flag indicating whether to return the wavefunctions composition. Default is False. If the calculation is perfomed in the complete basis set or on a reduced one (different from just considering the ground state only with ground_only=True) the projection is a list of two dictionaries. The first without, the second with, the MJ label.
- save_label (bool, optional): A flag indicating whether to save the labels of the states. Default is False.
- save_LF (bool, optional): A flag indicating whether to save the Hamiltonian matrix (without the Zeeman contribution). Default is False.
- save_matrix (bool, optional): A flag indicating whether to save the Hamiltonian matrix. Default is False.

Returns:

- result (numpy.ndarray): The eigenvalues and eigenvectors of the Hamiltonian matrix. The first row of the matrix are the eigenvalues. Each column of the matrix is an eigenvector.

– projected (dict): The wavefunctions composition. The states are labeled as progressive numbers.

calculation.build_matrix() Builds the Hamiltonian matrix.

Parameters:

– elem (list): The contributions to the Hamiltonian matrix among: 'Hee'=interelectronic repulsion, 'Hso'=spin-orbit coupling, 'Hcf'=crystal-field/ligand-field contribution, 'Hz'=Zeeman splitting.
– F0 (float, optional): The F0 Slater-Condon parameter in cm^{-1}. Default is 0.
– F2 (float, optional): The F2 Slater-Condon parameter in cm^{-1}. Default is 0.
– F4 (float, optional): The F4 Slater-Condon parameter in cm^{-1}. Default is 0.
– F6 (float, optional): The F6 Slater-Condon parameter in cm^{-1}. Default is 0.
– zeta (float, optional): The spin-orbit coupling parameter in cm^{-1}. Default is 0.
– k (float, optional): The orbital reduction factor. Default is 1.
– dic_bkq (dict, optional): A dictionary containing the crystal field parameters in Wybourne formalism in cm^{-1}. Default is None. dic_bkq = 'k':'q':0.0, '-q':0.0, ..., with k=2,4,..2*l and q=-k,-k+1,...k-1,k.
– field (list, optional): The magnetic field vector in Tesla. Default is [0.,0.,0.].
– save_label (bool, optional): A flag indicating whether to save the labels of the states. Default is False.
– save_LF (bool, optional): A flag indicating whether to save the Hamiltonian matrix (without the Zeeman contribution). Default is False.
– save_matrix (bool, optional): A flag indicating whether to save the Hamiltonian matrix. Default is False.

Returns:

– matrix (numpy.ndarray): The Hamiltonian matrix (complex).

calculation.opt_eigenfunction_minimization() Wavefunction optimization algorithm. It's based on the maximization of a single component of the ground state wavefunction using the dual anneling minimization algorithm implemented in scipy.

Parameters:

– calc (object): The instance of the class.
– dic_Bkq (dict): A dictionary containing the crystal field parameters in Wybourne Default is None. dic_bkq = 'k':'q':0.0, '-q':0.0, ..., with k=2,4,..2*l and q=-k,-k+1,...k-1,k.
– shape (int): The shape of the ground state wavefunction.

Returns:

– dic_rot (dict): A dictionary containing the rotated crystal field parameters.
– quat (list): A list containing the quaternion for the rotation of the crystal field parameters.

calculation.opt_eigenfunction_grid() Wavefunction optimization algorithm. It's based on a grid search, using REPULSION angles (evenly sampled on a sphere), read from crystdat.py.

Parameters:

- calc (object): The instance of the class.
- dic_Bkq (dict): A dictionary containing the crystal field parameters in Wybourne Default is None. dic_bkq = 'k':'q':0.0, '-q':0.0, ..., with k=2,4,..2*l and q=-k,-k+1,...k-1,k.
- shape (int): The shape of the ground state wavefunction.

Returns:

- dic_rot (dict): A dictionary containing the rotated crystal field parameters.
- quat (list): A list containing the quaternion for the rotation of the crystal field parameters.

Magnetics()

Class for magnetic properties calculations.

References:

1. R. Boca, "theoretical fundations of molecular magnetism" 1999.

Attributes:

- result (np.array): eigenvalue and eigenvectors of the Hamiltonian matrix.
- par_dict (dict): dictionary of parameters used in the calculation.
- calc (Calculation): instance of the Calculation class.
- basis (np.array): basis of the Hamiltonian matrix.
- Hterms (list): The contributions to the Hamiltonian matrix among: 'Hee'=interelectronic repulsion, 'Hso'=spin-orbit coupling, 'Hcf'=crystal-field/ligand-field contribution, 'Hz'=Zeeman splitting.
- kB (float): Boltzmann constant.
- mu0 (float): vacuum permeability.
- muB (float): Bohr magneton.

Magnetics.__init__() Initializes the Magnetics class, computing the 0 field results.

Parameters:

- calc (Calculation): instance of the Calculation class.
- contributes (list): The contributions to the Hamiltonian matrix among: 'Hee'=interelectronic repulsion, 'Hso'=spin-orbit coupling, 'Hcf'=crystal-field/ligand-field contribution, 'Hz'=Zeeman splitting.
- par (dict): dictionary of parameters used in the calculation.

Magnetics.mag_moment() Computes the magnetic moment matrix.

Parameters:

- k (int, optional): orbital reduction factor.
- evaluation (bool, optional): A flag indicating whether the parameters are symbolic or numerical. Default is True.

Returns:

- matrix (numpy.ndarray): the magnetic moment matrix (complex).

Magnetics.effGval() Computes the effective g-matrix for the Kramers doublets specified.

Parameters:

- levels (list): list of the Kramers doublet levels, e.g. [(1,2), (3,4)].
- v_matrix (numpy.ndarray, optional): eigenvectors of the Hamiltonian matrix. Default is None.

Returns:

- np.sqrt(2*w) (numpy.ndarray): eigenvalues of the g-matrix.
- v (numpy.ndarray): eigenvectors of the g-matrix.

Magnetics.susceptibility_field() Computes the scalar magnetization and magnetic susceptibility fields, for a set of field vectors (evenly sampled on a sphere) at a certain temperature.

Parameters:

- fields (numpy.ndarray): field vectors (shape = N x 3, in T).
- temp (float): temperature in K.
- delta (float, optional): differentiation step in T. Default is 0.001.
- wordy (bool, optional): A flag indicating whether to print the results. Default is False.

Returns:

- M_list (numpy.ndarray): scalar magnetization field (SI unit).
- susc_list (numpy.ndarray): scalar magnetic susceptibility field (SI unit).

Magnetics.susceptibility_B_copy() Computes the magnetic susceptibility tensor as the derivative of the magnetization vector in x,y,z, for a set of field vectors (evenly sampled on a sphere). The values are then averaged among the different directions of the magnetic field vector, hence it is generally enough to pass a single field vector as e.g. np.array([[0,0,1]]) (fields needs to be bidimensional).

Parameters:

- fields (numpy.ndarray): field vectors (shape = N x 3, in T).
- temp (float): temperature in K.
- delta (float, optional): differentiation step in T. Default is 0.001.
- wordy (bool, optional): A flag indicating whether to print the results. Default is False.

Returns:

- chi_tensor (numpy.ndarray): magnetic susceptibility tensor (SI unit).
- Mav (numpy.ndarray): magnetization vector (SI unit).

Magnetics.dfridr() Returns the derivative of the function at a point x by Ridders' method of polynomial extrapolation. The value h is input as an estimated initial stepsize. It has to be bigger than the one you would pass to a standard numerical differentiation method. The stepsize is decreased by CON at each iteeration. Max size of tableau is set by NTAB. See NJA-CFS documentation for references.

Parameters:

– func (function): function to differentiate.
– x (numpy.ndarray): differentiation variable.
– h (float): step size.
– idxi (int): index of the differentiation variable.
– shape (tuple): shape of the output.
– fargs (tuple): additional arguments for the function.

Returns:

– result (numpy.ndarray): result of the differentiation.
– err (float): error estimation of the differentiation.

Magnetics.susceptibility_B_ord1() Computes the magnetic susceptibility tensor as the derivative of the magnetization vector in x,y,z, for a set of field vectors (evenly sampled on a sphere). The values are then averaged among the different directions of the magnetic field vector, hence it is generally enough to pass a single field vector as e.g. np.array([[0,0,1]]) (fields needs to be bidimensional). The differentiation method is based on Ridders' method of polynomial extrapolation. The value h is input as an estimated initial stepsize (it has to be larger that the one you would pass to a standard differentiation procedure). An estimate of the error is also computed.

Parameters:

– fields (numpy.ndarray): field vectors (shape = N x 3, in T).
– temp (float): temperature in K.
– basis (numpy.ndarray): basis of the Hamiltonian matrix.
– LF_matrix (numpy.ndarray): 0 field Hamiltonian matrix.
– delta (float, optional): differentiation step in T. Default is 1.

Returns:

– chi_tensor (numpy.ndarray): magnetic susceptibility tensor (SI unit).
– err_tensor (numpy.ndarray): error estimation for the computation of the magnetic susceptibility tensor (SI unit).

Additional functions

Full_basis() Produces the complete basis set for a given configuration, saved in different format:
1. basis –> array: n. microstates x [2S, L, 2J, 2M, sen (,count)]
2. dic_LS –> dict: '[2S, L, 2J, 2M, sen (,count)]': label as N. and K.
3. basis_l –> list: 2S+1 L (J)
4. basis_l_JM –> list: 2S+1 L (J) MJ

Parameters:

– conf (str): The electron configuration.

Returns:

– basis (numpy.ndarray): The complete basis set for the configuration.

Table 1: The energy level values (in cm^{-1}) relative to the ground state multiplet for [Dy-DOTA(H_2O)]$^-$ (C_4) complex obtained from the complete approach (top row) and the ground-only calculation, including only the $^6H_{15/2}$ (bottom row). All the states are doubly degenerate.

complete basis:	7.181	63.027	95.133	145.614	209.124	288.136	386.291
$^6H_{15/2}$ basis:	7.501	63.598	96.586	148.052	212.781	293.036	392.296

- dic_LS (dict): A dictionary for the LS-coupling scheme.
- basis_l (numpy.ndarray): The labels for the states.
- basis_l_JM (numpy.ndarray): The labels for the states (with MJ).

4 Example scripts

4.1 Energy levels calculation

The simulation starts with the definition of a *calculation* object, that takes as input the configuration of choice. The calculations can be performed on transition metal complex, for configurations from d^1 to d^9 (from 10 to 252 microstates in the complete basis respectively), and on lanthanoid complexes, for configurations from f^1 to f^{13} (from 14 to 3432 microstates in the complete basis respectively). At this stage, the basis set is generated. Given the considerations on the basis set choice, I included in NJA-CFS the possibility of choosing specific subsets of microstates, to face the inevitable increase of computational time that computation on f^n configuration can cause (it can be applied also to d^n configurations). The basis set reduction can be performed at multiple levels. The easy way is, in case of f^n configurations (for n from 2 to 12), to use only the ground multiplet, by setting the *ground_only* flag as true (see table 1). Alternatively, it is possible to select a specific number of free ion terms based on their spin multiplicity, to mimic what one would do in a CAS calculation in ORCA[20]. This is done by adding a dedicated line to the code. At this level, one can also specify whether to compute the RMEs from cfps or to read the values from the tables. The computation of such integrals, even if does not make a sensible difference simulation-wise, I believe it has a non-negligible didactic value since it is not trivial both to find the necessary tables in a computationally usable form and to find the equations for their computation.

The energy levels and eigenfunctions composition can be performed using the *MatrixH* function. In this function, the type of contributions for the effective Hamiltonian has to be specified, together with the necessary parameters. The electron-electron interaction needs the Slater-Condon parameters: F^2 and F^4 for d^n configurations and F^2, F^4 and F^6 for f^n configurations. For spin-orbit coupling, the SO coupling coefficient (ζ) and the orbital reduction factor (κ) have to be defined. Concerning the crystal-field contribution, different coefficients formalism can be adopted. The convention that is used to compute the CF matrix elements, and in which all the others are converted, is Wybourne's definition of CFPs (see equation 122), in which B_q^k and $B_q'^k$ are the (real-valued) real and imaginary part of the Wybourne's CFPs. Lastly, the Zeeman term requires the definition of a magnetic field vector and the orbital reduction factor κ.

An example of code for the computation of energy levels and wavefunctions for a nickel(II) complex (d^8) using parameters from a CASSCF(8,5) calculation performed with ORCA soft-

ware.

```
import nja-cfs as nja

conf = "d8"
contr = ["Hee","Hso","Hcf"]
calc = nja.calculation(conf, ground_only=False)
dic_orca = nja.read_AILFT_orca6("test/calcsuscenisalfix.out", conf,
                                        method="CASSCF",
                                        return_V=False,
                                        rotangle_V=False,
                                        print_orcamatrix=False)
result = calc.MatrixH(contr, **dic_orca,
                                    evaluation=True,
                                    wordy=True,
                                    ground_proj=False,
                                    return_proj=False)
```

In this example, the complete basis and parameters coming from the output file of the QC calculation are used. The *result* matrix stores the energy levels on the first row and the wavefunctions on the corresponding column.

Functions for the rotation of Wybourne's CFPs are available, which can be used in the Euler angle formalism or in quaternions (see appendix 5.3). The rotation can be manually defined, or, in case of a point charge model for the ligands, an automatic fitting procedure can be applied, and implemented in a dedicated package (*vide infra*).

Once the calculation of energy levels and wavefunctions terminates, the energy levels splitting diagram (interactively open in a *matplotlib* interface) (see figure 1) and the state composition in terms of $|LSJ\rangle$ and/or $|LSJM_J\rangle$ can be obtained using dedicated functions.

4.2 Magnetic properties

The computation of magnetic properties starts with a calculation of the energy levels (performed as described in the preceding paragraph). Subsequently, the *Magnetics* object can be defined, and subsequently used for the computation of the different magnetic properties:

- effective g matrix for a Kramer doublet of states $|i\rangle$ and $|j\rangle$, using the following definition:

$$g_{kl} = \langle i|\mu_k|i\rangle\langle i|\mu_l|i\rangle + 2\langle i|\mu_k|j\rangle\langle j|\mu_l|i\rangle + \langle j|\mu_k|j\rangle\langle j|\mu_l|j\rangle \tag{1}$$

  with: $\mu_k = \kappa L_k + g_e S_k$ $(k,l = x,y,z)$ and the square root of the double of the eigenvalues of this matrix are the actual effective g values for the x, y and z directions,
- magnetization and magnetic susceptibility scalar fields, computed as

$$M(\mathbf{B}) = \mu_B \frac{\sum_k \sum_i \mu_k e^{-\frac{E(\mathbf{B})_i}{k_B T}}}{\sum_i e^{-\frac{E(\mathbf{B})_i}{k_B T}}} \tag{2}$$

$$\chi(\mathbf{B}) = \mu_0 \mu_B \frac{M(\mathbf{B}+\delta\mathbf{B}) - M(\mathbf{B}-\delta\mathbf{B})}{2\delta\mathbf{B}} \tag{3}$$

  where $E(\mathbf{B})_i$ is the energy value for the level i computed for the magnetic field vector $\mathbf{B}$, k_B is the Boltzmann constant, T is the temperature and $k = x,y,z$. The calculation is performed for each magnetization vector $\mathbf{B}$ defined on the surface of a sphere according

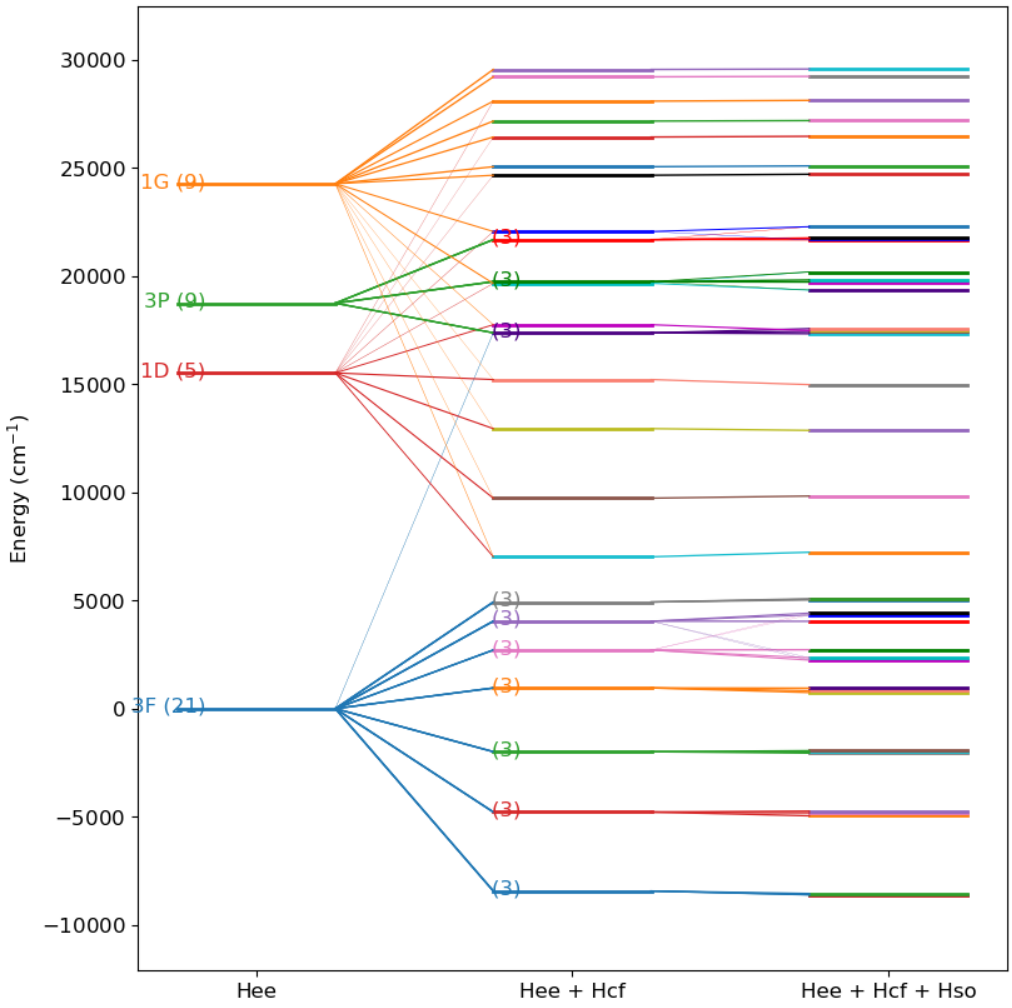

Figure 1: Energy levels splitting diagram for the $3d^8$ NiSAL-HDPT complex[21]. The parameters for the CF effective Hamiltonian come from a CASSCF(8,5) calculation performed with ORCA software. The free ion term state labels are also indicated, together with the $|LSJ\rangle$ degeneracy. The thickness of the lines connecting different levels is proportional to their relative contribution to the state composition. The energy of the ground-level free ion term is set to zero.

to a set of angles determined either with the REPULSION algorithm[22] or with the golden spiral method. Such fields can be plotted with a dedicated function.

- magnetization vector and susceptibility tensor, defined, for a certain magnetic field vector $\mathbf{B} = [B_x, B_y, B_z]$, as:

$$M_k = \mu_B \frac{\sum_i (\mu_k)_i e^{-\frac{E_i}{k_B T}}}{\sum_i e^{-\frac{E_i}{k_B T}}} \tag{4}$$

$$\chi_{kl} = \mu_B \mu_0 \frac{\partial M_k}{\partial B_l}. \tag{5}$$

The numerical derivative is computed as:

$$\chi_{kl} = \mu_B \mu_0 \frac{M_l(B_k + \delta B) - M_l(B_k - \delta B)}{2\delta B} \tag{6}$$

An example of the application of such functions for the calculation of effective g values and susceptibility tensor for the DyDOTA complex, in the reduced basis set, is presented below:

```
import nja-cfs as nja
import numpy as np

conf = "f9"
calc = nja.calculation(conf, ground_only=True)
contr = ["Hcf","Hz"]

# Stevens CFP from J. Am. Chem. Soc. 2021, 143, 8108-8115
cfp_list = np.loadtxt("test/CFP_DyDOTA.txt")
dic_Akqrk = {}
count = 0
for k in range(2,7,2):
    dic_Akqrk[f'{k}'] = {}
    for q in range(k,-k-1,-1):
        dic_Akqrk[f'{k}'][f'{q}'] = cfp_list[count]/nja.Stev_coeff(str(k), conf)
        count += 1

# Conversion from Stevens' CFPs to Wybourne's CFPs
dic_Bkq = nja.from_Akqrk_to_Bkq(dic_Akqrk)
dic = {}
dic['dic_bkq'] = dic_Bkq
dic['field'] = np.array([0.0,0.0,0.0])

Magn = nja.Magnetics(calc, contr, dic)
# Computation of the magnetic susceptibility matrix
chi_B = Magn.susceptibility_B_copy(fields=np.array([[0.0,0.0,0.0]]),
                                   temp=298.,
                                   delta=0.01)
# computation of the effective g values (gw) and reference frame (gv)
gw, gv = Magn.effGval_copy((1,2))
```

The Stevens CFPs for DyDOTA[23] are converted to the Wybourne's ones before the calculation and the g matrix is computed for the ground Kramer's doublet.

The computation of the magnetic susceptibility tensor, and in general the multiple derivatives of the magnetization vector with respect to the magnetic field vector, can alternatively be computed with an alternative algorithm, based on Ridders' method of polynomial extrapolation[24]. The number of evaluations of the function is typically from 6 to 12, but can even be larger, therefore the computation can be too slow. For this reason, I also implemented an

alternative version of the function, based on the use of the *numba* package for Python's functions implementation. To use this algorithm specific data has to be stored from a preceding calculation of the energy levels.

```
import nja-cfs as nja

conf = "f13"
contr = ["Hee","Hso","Hcf"]

dic = nja.read_AILFT_orca6("../run_YbDOTA.out", conf)

# energy levels calculation (with save_label=True and save_LF=True)
calc = nja.calculation(conf, ground_only=True, TAB=True, wordy=True)
result, projected = calc.MatrixH(contr, **dic,
                                 wordy=True,
                                 ground_proj=True,
                                 return_proj=True,
                                 save_label=True,
                                 save_LF=True)
# loading of the data previously computed
basis = np.loadtxt("matrix_label.txt")
LF_matrix = np.load("matrix_LF.npy", allow_pickle=True, fix_imports=False)

# use of the loaded data for susceptibility tensor computation with Ridders'
    method
chi, err = nja.susceptibility_B_ord1(np.array([[0.0,0.0,0.0]]),
                                     298.,
                                     basis,
                                     LF_matrix,
                                     delta=1.0)
```

In this listing, the computation of the susceptibility tensor is performed with the Ridders' method for the YbDOTA complex. Besides the saving of computational time thanks to the *numba* engine, this procedure allows also an estimation of the numerical error from the derivative computation.

5 Theoretical background

The main contributions included in an effective Hamiltonian to describe the magnetic and electronic properties of small inorganic complexes (e.g. single molecule magnets) are: the electron-electron interaction ($\hat{\mathcal{H}}_{\text{e-e}}$), the spin-orbit coupling ($\hat{\mathcal{H}}_{\text{s-o}}$), the crystal field splitting ($\hat{\mathcal{H}}_{\text{cf}}$) and - if the system is subjected to the action of an external field - the term describing the interaction, e.g. the Zeeman splitting caused by the system exposure to external magnetic fields ($\hat{\mathcal{H}}_{\text{Z}}$):

$$\hat{\mathcal{H}} = \hat{\mathcal{H}}_{\text{e-e}} + \hat{\mathcal{H}}_{\text{s-o}} + \hat{\mathcal{H}}_{\text{cf}} + \hat{\mathcal{H}}_{\text{Z}} \tag{7}$$

The latter term is usually the smaller one.

The extension of the CF effect depends on the type of considered metal ion. For transition metals, the splitting contribution introduced by the ligands is generally bigger than the spin-orbit coupling contribution, while the reverse holds for lanthanoid ions. This will influence the number and classification scheme of the microstates (basis set) included in a hypothetical CF calculation. For example, in the case of f^n configurations the basis set can be reduced to include just the ground state multiplet, which will be almost independent of the coordination environment, like it is done in many known CF programs, e.g. SIMPRE. This will lead

to a considerable reduction of the computational burden, especially for the calculation on lanthanoid complexes with $4 \leq n \leq 10$. However, the inclusion of all possible microstates ensures intermediate coupling and J-mixing, therefore this simplification should be evaluated case by case.

Once the Hamiltonian and the basis set classification scheme are defined, the other main point in a CF program is the formalism for the resolution of the integrals. In quantum-mechanical (QM) calculations, Hartree-Fock being the most simple example, the wavefunctions are generally represented as determinantal product states of the type:

$$K_1, K_2, ...K_n = (n!)^{-\frac{1}{2}} \sum_P (-1)^p \psi_1(K_1)\psi_2(K_2)...\psi_n(K_n) \tag{8}$$

where p is the number of permutations, n is the number of electrons in the considered configuration, ψ_i are the wavefunction basis set, generally composed by the product of an orbital part and a spin part, and K_i (with $i = 1, ..., n$) are the set of quantum numbers used to unambiguously describe the electron in the specific orbital. Given the computational burden that the resolution of integrals in first quantization with such wavefunctions could bring, an alternative way for Hamiltonian matrix elements calculation has been developed thanks to the work of Condon and Shortley [25], Racah[26], and others, on the application of group theory to the interpretation of atomic spectra.

In this approach the Hamiltonian integrals calculations are performed according to the theory of angular momenta in Racah's tensor operator formalism[27, 28, 29, 30, 31, 15] through the well-known Wigner-Eckart theorem, expressed for a generic tensor operator of rank k and order q as

$$\langle j'm'|T_q^{(k)}|jm\rangle = \langle jmkq|j'm'\rangle \langle j'\|T^{(k)}\|j\rangle \tag{9}$$

where the factors on the r.h.s. are respectively: Clebsch-Gordan coefficients $\langle jmkq|j'm'\rangle$ and the so-called Reduced Matrix Elements (RME) $\langle j'\|T^{(k)}\|j\rangle$ (*vide infra*).

Within the theoretical framework outlined above, equation 7 is written in terms of Racah's tensor operators[16], with the crystal field expressed distribution of discrete charges, as

$$\begin{aligned}\hat{\mathcal{H}} = \left(\frac{e^2}{4\pi\varepsilon_0}\right) \sum_{i=1}^{n}\sum_{j>i}^{n}\sum_{k=0}^{\infty}[r_>^{-(k+1)} \cdot r_<^k] \sum_{q=-k}^{+k}(-1)^q C_{-q}^{(k)}(i) C_q^{(k)}(j) + \\ +\hbar^{-2}\zeta\kappa\{\mathbf{L}^{(1)} \otimes \mathbf{S}^{(1)}\}^0 + \\ + \left(\frac{e^2}{4\pi\varepsilon_0}\right) \sum_{i=1}^{n}\sum_{L} Z_L \sum_{k=0}^{\infty}[r_>^{-(k+1)} \cdot r_<^k] \sum_{q=-k}^{+k}(-1)^q C_{-q}^{(k)}(L) C_q^{(k)}(i) + \\ +\mu_B\hbar^{-1} \sum_{q=-1}^{+1}(-1)^q B_{-q}^{(1)}(\kappa L_q^{(1)} + g_e S_q^{(1)}).\end{aligned} \tag{10}$$

where e is the electron charge, ζ is the spin-orbit coupling constant ($\lambda = \pm\zeta/(2S)$), κ is the orbital reduction factor, Z_L is the charge of the L-ligand defined as fraction of electronic charge and, $r_>$ and $r_<$ are the farthest and closest positions of electron/ligand charges respectively, with respect to the center of the reference frame, assumed coincidient with the position of the metal center. $C_q^{(k)}$ are the tensor operators representation of the complex spherical harmonics,

also called Racah normalized spherical harmonics [28]:

$$C_q^{(k)} = \sqrt{\frac{4\pi}{2k+1}} Y_k^q, \tag{11}$$

$L^{(1)}$ and $S^{(1)}$ are the spherical tensor representations of orbital angular momentum and spin angular momentum operators respectively and $B_q^{(1)}$ is the spherical tensor representation of the magnetic field vector.

To reduce the computational burden further, instead of using the determinantal product of states, the basis wave-vectors are combined to obtain a proper wavefunction through coefficients called: fractional parentage coefficients (cfp) $G_{n,K}^{n-1,K'}$

$$|\psi(l^n, K)\rangle = \sum_{K'} |\psi(l^{n-1}, K')\rangle G_{n,K}^{n-1,K'}. \tag{12}$$

In this equation (analogous to equation 15 of appendix **??**), the sum runs over the basis functions of the $n-1$ states. In this case, K follows the Koster-Nielson classification scheme [14], where the atomic states are constructed in the Russel-Sunders coupling scheme, in which, besides the canonical $|L, S, J, M_J\rangle$, additional quantum numbers, not of physical meaning, are introduced based on the properties of certain mathematical groups. Specifically, these groups are: R_5 in the case of d^n configurations and R_7 and G_2 for f^n configurations. Additionally, the seniority number is also used (more detailed information on cfps are available in the appendices 5.1 and 5.4).

The Clebsh-Gordan coefficients and RME are generally tabulated. Alternatively, recursive relations are available, based on the use of 3j and 6j-symbols for the first and of cfps for the latter (see appendix 5.1.1). Also for cfps, a closed form has been defined by Redmond [32], however, it does not ensure the reproduction of the same coefficients obtained by group-theoretical considerations. A more complete approach, based on the use of Casimir operators, was defined by Nielson in his doctoral thesis[33] (described in appendix 5.1). All these relations are available in NJA-CFS.

5.1 Fractional parentage coefficients

The following discussion will be limited to f^n configurations (unless otherwise specified) since analogous considerations and the complete set of coefficients for d^n configurations can be easily found in [29].

The determinantal product of states is the most common wavefunction structure used in QM calculations, however, their construction for systems with more than two electrons can be quite cumbersome. To avoid that, the tensor operator representation of Hamiltonian contributions can be exploited. This enables the application of the Wigner-Eckart theorem, whose most general form is the following:

$$\langle \alpha K M_K | T_q^{(k)} | \alpha' K' M'_K \rangle = (-1)^{K-M_K} \begin{pmatrix} K & k & K' \\ -M_K & q & M'_K \end{pmatrix} \langle \alpha K \| T^{(k)} \| \alpha' K' \rangle \tag{13}$$

where K and M_K are the angular momentum quantum numbers used to discriminate different states. The product of the first two terms on the r.h.s. is a vector coupling coefficient,

while the remaining term is called reduced matrix element (RME).

This method can be applied as it is only for configurations comprising two equivalent electrons. For l^n with $n > 2$ the possibility of the occurrence of more than one term with a given K, e.g. with the same L and S, raises the problem of defining a state. So, a classification like the following should be used:

$$|l^n \alpha S L M_S M_L\rangle \tag{14}$$

where α would be the additional quantum number used for terms' differentiation.

Considering that S, L, M_L, M_S are group-theoretical in origin, since e.g. the $2L+1$ components of a multiplet (with given S and M_S) transform under rotation according to the irreducible representation $\mathfrak{D}_L$ of R_3, the same principle could be used for other continuous groups that include the operations of R_3. The method for the individuation of such groups and the application of tensor operators to the theory of continuous groups are due to Racah[34].

Racah found that the group that could serve for such scope are (in order): $R_3 \subset G_2 \subset R_7 \subset U_7$. The quantum numbers associated to each groups are labels for their irreducible representations. G_2 is represented by $U = (u_1, u_2)$ and R_7 is represented by $W = (w_1, w_2, w_3)$. The representation of R_3 (and its subgroups) are the usual orbital angular momentum quantum numbers and the use of irreducible representations of U_7 (represented as $[\lambda_1, \lambda_2, ..., \lambda_n]$) is equivalent to the specification of the spin quantum number S. The decomposition schemes for the reductions $U_7 \to R_7$, $R_7 \to G_2$ and $G_2 \to R_3$ are reported in tables 5-1, 5-2 and 5-3 of [31] respectively.

However, for few cases, also these classification scheme is not enough, and Racah introduced the seniority quantum number[30], that is (for a certain S, W pair) represented by an integer ν such that f^ν is the first configuration in which that S, W appears.

The immediate advantage of the group theoretical classification of states is that many states can be uniquely without going to the lengths of calculating the particular linear combination of determinantal product state to which they correspond. In order to sfruttare le potenzialità di questo approccio it is necessary to find a different way for the evaluation of the matrix elements, not requiring the construction of such linear combinations. Racah[29], through the elaboration of an idea of Bacher and Goudsmit[35], developed an elegant solution to this problem.

Assuming that Ω refers to the set of quantum numbers describing the state of l^n configuration, $\bar{\Omega}$ the set of quantum numbers for the state of l^{n-1} and ω the quantum numbers for the one-electron state. Since every determinantal product state for an n-electron system is a sum of terms[31]:

$$|\Omega\rangle = \sum_{\bar{\Omega},\omega} \langle \bar{\Omega}; \omega | \Omega \rangle | \Omega \rangle | \omega_n \rangle \tag{15}$$

where $\langle \bar{\Omega}; \omega | \Omega \rangle$ are the coefficients of our interest and the subscript n indicates that the last state in the equation refers to the nth electron.

Using this equation, the matrix elements for a generic single-particle operator $F = \sum_i f_i$, between two states of the l^n configuration, can be expressed as:

$$\langle \Omega | F | \Omega' \rangle = n \sum_{\bar{\Omega},\omega,\omega'} \langle \Omega | \bar{\Omega}; \omega \rangle \langle \omega_n | f_n | \omega_n \rangle \langle \bar{\Omega}; \omega' | \Omega' \rangle \tag{16}$$

while, for a two-particles operator $G = \sum_{i>j} g_{ij}$:

$$\langle \Omega | G | \Omega' \rangle = [n/(n-2)] \sum_{\bar{\Omega},\bar{\Omega}',\omega} \langle \Omega | \bar{\Omega}; \omega \rangle \langle \bar{\Omega} | \sum_{j<i\neq n} g_{ij} | \bar{\Omega}' \rangle \langle \bar{\Omega}'; \omega | \Omega' \rangle \tag{17}$$

The procedure used for the computation of the cfp is taken from the Nielson PhD thesis[33].

According to the group-theoretical classification of states:

- the single-electron wavefunction, with quantum numbers: $S = 1/2$, $W = (100)$, $U = (10)$, $L = 3$, m_l, m_s, correspond to the irreducible representation Γ_f,
- the antisymmetric states of $n-1$ electrons, with quantum numbers: S', W', U', α', L', M'_L, M'_S, constitute the basis of the irreducible representation $\Gamma_A^{(n-1)}$,
- the antisymmetric states of n electrons, with quantum numbers: S, W, U, α, L, M_L, M_S, constitute the basis of the irreducible representation $\Gamma_A^{(n)}$.

The objective is to find the reduction coefficients of the Kronecker product $\Gamma_A^{(n-1)} \times \Gamma_f$ relative to the antisymmetric part $\Gamma_A^{(n)}$, indicated in Nielson's notation as:

$$b_n(S, W, U, \alpha, L, M_L, M_S; S', W', U', \alpha', L', M'_L, M'_S, m_l, m_s) \tag{18}$$

These coefficients can be factorized according to the Racah's theorem as follows:

$$\begin{aligned} b_n(S, W, U, \alpha, L, M_L, M_S; S', W', U', \alpha', L', M'_L, M'_S, m_l, m_s) = \\ C^{S'\ 1/2\ S}_{n,W'\ (100)\ W} C^{W'\ (100)\ W}_{U'\ (10)\ U} C^{U'\ (10)\ U}_{\alpha' L'\ 3\ \alpha L} C^{L'\ 3\ L}_{M'_L\ m_l\ M_L} C^{S'\ 1/2\ S}_{M'_S\ m_s\ M_S} \end{aligned} \tag{19}$$

where the last two coefficients are Clebsch-Gordon coefficients and the product of the other three compose the cfp:

$$a_n(S, W, U, \alpha, L; S', W', U', \alpha', L') = C^{S'\ 1/2\ S}_{n,W'\ (100)\ W} C^{W'\ (100)\ W}_{U'\ (10)\ U} C^{U'\ (10)\ U}_{\alpha' L'\ 3\ \alpha L} \tag{20}$$

In Racah's notation (used from now on):

$$(l^{n-1}(\alpha' \nu' S' L') l S L | \} l^n \alpha \nu S L) \equiv a_n(S, W, U, \alpha, L; S', W', U', \alpha', L') \tag{21}$$

$$(f^{n-1} \nu' S' + f | \} f^n \nu S) \equiv C^{S'\ 1/2\ S}_{n,W'\ (100)\ W} \tag{22}$$

$$(W' U' + f | W U) \equiv C^{W'\ (100)\ W}_{U'\ (10)\ U} \tag{23}$$

$$(U' \alpha' L' + f | U \alpha L) \equiv C^{U'\ (10)\ U}_{\alpha' L'\ 3\ \alpha L} \tag{24}$$

For the coefficients $(f^{n-1} \nu' S' + f | \} f^n \nu S)$ a closed form is available (equations 52a-52b-56 of [30]). The other coefficients $(W' U' + f | W U)$ and $(U' \alpha' L' + f | U \alpha L)$, which represent the reduction coefficients of the Kronecker product of groups R_7 and G_2 respectively, can be computed via the diagonalization of the corresponding Casimir operators (G) (equivalent of a diagonalization of L^2 operator for the reduction of a representation of the group R_3). The

Racah's expression of Casimir operators are:

$$G(R_7) = \frac{3}{5}(U^1)^2 + \frac{7}{5}(U^3)^2 + \frac{11}{5}(U^5)^2 \tag{25}$$

$$G(G_2) = \frac{3}{4}(U^1)^2 + \frac{11}{4}(U^5)^2 \tag{26}$$

where U^1, U^3 and U^5 are unit tensor operators, with known eigenvalues:

$$\lambda(W) = w_1(w_1+5)/2 + w_2(w_2+3)/2 + w_3(w_3+1)/2 \tag{27}$$

$$\lambda(U) = (u_1^2 + u_1u_2 + u_2^2 + 5u_1 + 4u_2)/12 \tag{28}$$

The matrix elements of $G(G_2)$ between the states arising from the coupling of a single f electron to parent states of quantum numbers U', L', M'_L to give the final states of angular momentum L, M_L, can be expressed as:

$$\begin{aligned}
&\langle U'\alpha' L' + f \to LM_L | G(G_2) | U'\alpha'' L'' + f \to LM_L \rangle = \\
&= (-1)^{L+3+L''} \frac{11}{2} \left\{ \begin{array}{ccc} L & 3 & L' \\ 5 & L'' & 3 \end{array} \right\} (U'\alpha' L' \| U^5_{n-1} \| U' L''\alpha'') + \\
&+ \delta_{\alpha'\alpha''}\delta_{L'L''} \left[\lambda(U') + \lambda(10) + \frac{1}{112}(L(L+1) - L'(L'+1) - 3(3+1)) \right]
\end{aligned} \tag{29}$$

Once these matrix elements are evaluated for a chosen L and U', its diagonalization produces eigenvalues for U containing L and known to occur in the reduction $(U') \times (10)$. Each (normalized) eigenvector is a whole set of $(U'\alpha' L' + f | U\alpha L)$ with L' varying along the row.

The RME of U^5 are computed iteratively, from the previous cycle of calculation. In this case the labels n, $n-1$, $n-2$ are used only to represent the hierarchy of computation, these coefficients are general and do not depend on the particular number of electrons considered. The necessary relations are the following:

$$\begin{aligned}
&(U'''L''' + f \to L' \| U^5_{n-1} \| U'''L'^v + f \to L'') = \\
&= (2L'+1)^{1/2}(2L''+1)^{1/2} \left[(-1)^{L'''+L''} (U'''L''' \| U^5_{n-2} \| U'''L'^v) \right. \\
&\left. \times \left\{ \begin{array}{ccc} L''' & L' & 3 \\ L'' & L'^v & 5 \end{array} \right\} + (-1)^{L'^v+L'} \delta_{L'''L'^v} \delta_{\alpha'''\alpha'^v} \left\{ \begin{array}{ccc} 3 & L' & L''' \\ L'' & 3 & 5 \end{array} \right\} \right]
\end{aligned} \tag{30}$$

and

$$\begin{aligned}
&(U'\alpha' L' \| U^5_{n-1} \| U'\alpha'' L'') = \\
&= \sum_{\alpha''', L''', \alpha'^v, L'^v} (U'''\alpha'''L''' + f | U'\alpha' L')(U'^v\alpha'^v L'^v + f | U'\alpha' L') \\
&\times (U'''L''' + f \to L' \| U^5_{n-1} \| U'''L'^v + f \to L'')
\end{aligned} \tag{31}$$

An analogous procedure can be applied (subsequently) for the computation of $(W'U' +$

$f|WU)$. The matrix elements of $G(R_7)$ are expressed as:

$$\begin{aligned}
&\langle W'U' + f \to ULM_L|G(R_7)|W'U'' + f \to ULM_L\rangle = \\
&= \sum_{L',L''} \Big[(U'L' + f|UL)(U''L'' + f|UL)(-1)^{L''+3+L} \\
&\times \frac{14}{5} \begin{Bmatrix} L & 3 & L' \\ 3 & L'' & 3 \end{Bmatrix} (W'U'L'\|U^3_{n-1}\|W'U''L'')\Big] + \\
&+\delta_{U'U''} \left[\frac{4}{5}\lambda(U) + \lambda(W') - \frac{4}{5}\lambda(U') + \lambda(100) - \frac{4}{5}\lambda(10)\right]
\end{aligned} \tag{32}$$

In this case, the W that have to be considered are those that occur in the reduction $W' \times (100)$. The iterative procedure for the computation of the RME for U^3 is similar to the one described for U^5, with equations:

$$\begin{aligned}
&(W'U'L'\|U^3_{n-2}\|W'U''L') = \\
&= \sum_{U''',U'^v} (W'''U''' + f|W'U')(W'''U'^v + f|W'U'') \\
&\times \Bigg[\sum_{L''',L'^v} (U'''L''' + f|U'L')(U'^vL'^v + f|U''L'') \\
&\times (W'''U'''L''' + f \to L'\|U^3_{n-1}\|W'''U'^vL'^v + f \to L'')\Big]
\end{aligned} \tag{33}$$

and:

$$\begin{aligned}
&(W'''U''' + f \to L'''\|U^3_{n-1}\|W'''U'^v + f \to L'') = \\
&= (-1)^{L'''+3+L''+3}(2L'+1)^{1/2}(2L''+1)^{1/2} \\
&\times \begin{Bmatrix} L''' & L' & 3 \\ L'' & L'^v & 3 \end{Bmatrix} (W'''U'''L'''\|U^3_{n-2}\|W'''U'^vL'^v) + \\
&+(-1)^{L'''+3+L'+3}\delta_{L'''L'^v}\delta_{U'''U'^v}(2L'+1)^{1/2}(2L''+1)^{1/2} \begin{Bmatrix} 3 & L' & L''' \\ L'' & 3 & 3 \end{Bmatrix}
\end{aligned} \tag{34}$$

5.1.1 Reduced matrix elements

The RMEs for the different tensor operators used in the integrals computation for the physical operators (vide infra) can be computed using the following equations, using the calculated cfps, for a generic rank k and configuration l^n:

$$\begin{aligned}
&(\alpha LS\|U^{(k)}\|\alpha'L'S') = \delta_{SS'}n[(2L+1)(2L'+1)]^{1/2} \\
&\times \sum_{\alpha'',L'',S''} (l^{n-1}(\alpha''L''S'')lSL|\}l^n\alpha LS)(l^{n-1}(\alpha''L''S'')lSL|\}l^n\alpha'L'S') \\
&\times (-1)^{K+L+l+L''} \begin{Bmatrix} l & L & L'' \\ L' & l & k \end{Bmatrix}
\end{aligned} \tag{35}$$

and:

$$(\alpha LS\|V^{(1k)}\|\alpha' L'S') =$$
$$= \frac{3}{2}[(2L+1)(2L'+1)(2S+1)(2S'+1)]^{1/2}$$
$$\times \sum_{\alpha'',L'',S''} (l^{n-1}(\alpha''L''S'')lSL|\}l^n\alpha LS)(l^{n-1}(\alpha''L''S'')lSL|\}l^n\alpha' L'S')$$
$$\times(-1)^{k+L+l+L''} \left\{ \begin{array}{ccc} l & L & L'' \\ L' & l & k \end{array} \right\} (-1)^{1+S+1/2+S''} \left\{ \begin{array}{ccc} 1/2 & S & S'' \\ S' & 1/2 & 1 \end{array} \right\} \quad (36)$$

In both equations, the 6-j symbol $\left\{ \begin{array}{ccc} a & b & c \\ A & B & C \end{array} \right\}$ represents the coupling of three angular momenta, and can be solved using the Racah's equation[28]:

$$\left\{ \begin{array}{ccc} a & b & c \\ A & B & C \end{array} \right\} = (-1)^{a+b+A+B} f(abc)f(ABc)f(AbC)f(aBC)$$
$$\sum_{n=n_{\min}}^{n_{\max}} (-1)^n (a+b+A+B+1-n)!/\left[n!(a+b-c-n)!\right.$$
$$\times(A+B-c-n)!(a+B-C-n)!(A+b-C-n)!$$
$$\left.\times(-a-A+c+C+n)!(-b-B+c+C+n)!\right] \quad (37)$$

with:

$$f(abc) = \left[\frac{(a+b-c)!(a-b+c)!(-a+b+c)!}{(a+b+c+1)!}\right]^{1/2}$$

and:

$$n_{\min} = \max\{0; a+A-c-C; b+B-c-C\}$$
$$n_{\max} = \min\{a+b+A+B+1; a+b-c; A+B-c; a+B-C; A+b-C\}$$

5.2 Crystal field coefficients formalism

The perturbation of the electron cloud of a metal ion, subjected to the influence of a coordination environment, can be effectively described by a crystal-field Hamiltonian of the type:

$$\hat{\mathcal{H}}_{cf} = -e\sum_{i=1}^{n} V(r_i) \quad (38)$$

where e is the elementary charge, the summation i runs over the n electrons in the open-shell orbitals and $V(r_i)$ is the potential felt by the electron at the position r_i. The crystal field potential, in presence of a field independent charge distribution $\rho(R)$, can be expressed as:

$$V(r_i) = \int \frac{\rho(R)}{|R-r_i|} d\tau \quad (39)$$

where R indicates a general point in the ligand framework and $d\tau$ is a volume element. Alternatively, considering a discrete distribution of charges represented in a point charge model

(PCM), the following expression is valid:

$$V(r_i) = \sum_L \frac{(-Ze)_L}{|R_L - r_i|} \tag{40}$$

where Z is the fraction of negative charge relative to the Lth ligand.

The potential in equation 39 can be expanded as follows:

$$\frac{1}{|R - r_i|} = \sum_{k=0}^{\infty} \frac{r_<^k}{r_>^{k+1}} P_k(\cos\omega) \tag{41}$$

where $r_<$ and $r_>$ are the shortest and longest distances respectively and $P_k(\cos\omega)$ are the Legendre polynomials relative to the angle ω between R and r_i. Generally, the ligands are more distant than the electron of the metal ion itself, therefore equation 41 becomes:

$$\frac{1}{|R - r_i|} = \sum_{k=0}^{\infty} \frac{r_i^k}{R^{k+1}} P_k(\cos\omega) \tag{42}$$

$P_k(\cos\omega)$ can be further expanded according to the spherical harmonics addition theorem [36]. According to that, ω can be decomposed in the polar angles: (θ, ϕ), which describe the distribution of ligand charges, and (θ_i, ϕ_i), which describe the position of the ith-electron:

$$P_k(\cos\omega) = \sqrt{\frac{4\pi}{2k+1}} \sum_{q=-k}^{k} Y_k^{q*}(\theta, \phi) Y_k^q(\theta_i, \phi_i) \tag{43}$$

leading to the final expression:

$$V(r_i) = \sum_{k=0}^{\infty} \sqrt{\frac{4\pi}{2k+1}} \left[\sum_{q=-k}^{k} Y_k^q(\theta_i, \phi_i) \int \rho(R) \frac{r_i^k}{R^{k+1}} Y_k^{q*}(\theta, \phi) d\tau \right] \tag{44}$$

or in the PCM:

$$V(r_i) = \sum_{k=0}^{\infty} \sqrt{\frac{4\pi}{2k+1}} \left[\sum_{q=-k}^{k} Y_k^q(\theta_i, \phi_i) \sum_L (-Ze)_L \frac{r^k}{R_L^{k+1}} Y_k^{q*}(\theta_L, \phi_L) \right] \tag{45}$$

The radial part of this expression and the angular part describing the ligand distribution are usually factorized in the so-called crystal field parameters (CFPs), with general expression [37]:

$$V(r_i) = \sum_{kq} f_{kq}(R) Y_k^q(\theta_i, \phi_i) \tag{46}$$

This factorization can be performed in many ways, giving rise to the multitudes of formalism available in literature (vide infra). I implemented some of them - and their conversion factors - in NJA-CFS.

Throughout the whole discussion on CFPs I will use the representation formalism employed by Ryabov and Rudowicz in their numerous reviews on the topic (referenced below).

As previously described, the (electric) crystal field of an open-shell ion in a crystalline environment can be described by a potential function which satisfies the Laplace equation.

The solution of this equation can be expressed in a series of spherical harmonics. Stevens (1952) was the first one to show that the matrix elements of these spherical harmonics, within the same angular momentum manifold, could be calculated by replacing them by operator equivalents having the similar transformation properties. Since then, tensor operators (TOs) of angular momenta [28] have been widely used in the construction of crystal field Hamiltonians [38] and (surely too) many conventions and formalism have been presented since then[39]. Two classes of TOs can generally be identified: the spherical tensor operators (STOs) and the tesseral tensor operators (TTOs), which are the operator equivalents to the spherical and tesseral harmonics respectively. The conventional Stevens operators (CSOs) [40] and the extended Stevens operators (ESOs)[39] belongs to the TTO class. Both of them are generally indicated as B_k^q, where $q \geq 0$ for the CSOs and $-k \leq q \leq k$ for the ESOs. The first are not used anymore since they do not transform consistently under a general rotation of coordinates for some crystal symmetries, as first pointed out by Buckmaster (1962) [41]. Nowadays, the most used operator equivalents are the Stevens ESOs one[42]. According to this approach, the crystal field hamiltonian in a single $J(f^n)$-multiplet or $L(d^n)$-multiplet can be expressed as:

$$\hat{\mathcal{H}}_{\text{cf}}^{\text{Stevens}} = \sum_{k,q} B_k^q O_k^q (J \text{ or } L) \tag{47}$$

where O_k^q are the ESOs, either defined in terms of total angular momentum J or orbital angular momentum L. The B_k^q defined herein should not be confused with the B_k^q from the $\hat{\mathcal{H}}_{\text{ZFS}}$, where the O_k^q are defined in terms of the spin angular momentum S. The CFPs defined in this way are real-valued. Another widespread notation used to express the crystal field hamiltonian, generally used for f^n multiplets, in terms of ESOs is the following:

$$\hat{\mathcal{H}}_{\text{cf}}^{\text{Stevens}} = \sum_{k,q} A_k^q \langle r^k \rangle \theta_k O_k^q \tag{48}$$

where A_k^q are the CFPs (like the B_k^q), $\langle r^k \rangle$ are the expectation value of r^k (listed in table 2 for f^n, analogous values are available also for d^n configurations) and the θ_k are the multiplicative Stevens factors, indicated as α, β and γ in the original Stevens paper from '52 [40] for the rank $k = 2, 4, 6$ respectively. The properties of TTOs differ from the STOs. An example of the latter is represented by the Wybourne operators.

According to Wybourne (1965)[44, 18, 5], the crystal field hamiltonian can be defined as:

$$\hat{\mathcal{H}}_{\text{cf}}^{\text{Wybourne}} = \sum_{k,q,i} B_q^k C_q^{(k)}(\theta_i, \phi_i) \tag{49}$$

where the sum over i runs on the electrons in the chosen (open-shell) configuration and $-k \leq q \leq k$ [18, 45]. $C_q^{(k)}$ (also called Racah's harmonics[28]) are Wybourne's spherical tensor operators $T_q^{(k)}$ equivalent to the renormalized spherical harmonics:

$$C_q^{(k)} = \sqrt{\frac{4\pi}{2k+1}} Y_k^q \tag{50}$$

Table 2: Values of radial integrals $\langle r^k \rangle$ (for $k = 2, 4, 6$) computed by Edvardsson and Klintenberg [43].

configuration	2	4	6
f^1	1.456	5.437	42.26
f^2	1.327	4.537	32.65
f^3	1.222	3.875	26.12
f^4	1.135	3.366	21.46
f^5	1.061	2.964	17.99
f^6	0.997	2.638	15.34
f^7	0.942	2.381	13.36
f^8	0.893	2.163	11.75
f^9	0.849	1.977	10.44
f^{10}	0.810	1.816	9.345
f^{11}	0.773	1.677	8.431
f^{12}	0.740	1.555	7.659
f^{13}	0.710	1.448	7.003

Table 3: Stevens coefficients α, β and γ for f^n configurations[40].

configuration	α	β	γ
f^1	$-\frac{2}{35}$	$\frac{2}{7\cdot 45}$	0
f^2	$-\frac{52}{11\cdot 15^2}$	$-\frac{4}{55\cdot 33\cdot 3}$	$\frac{17\cdot 16}{7\cdot 11^2\cdot 13\cdot 5\cdot 3^4}$
f^3	$-\frac{7}{33^2}$	$-\frac{8\cdot 17}{11^2\cdot 13\cdot 297}$	$-\frac{17\cdot 19\cdot 5}{13^2\cdot 11^3\cdot 3^3\cdot 7}$
f^4	$\frac{14}{11^2\cdot 15}$	$\frac{952}{13\cdot 3^3\cdot 11^3\cdot 5}$	$\frac{2584}{11^2\cdot 13^2\cdot 3\cdot 63}$
f^5	$\frac{13}{7\cdot 45}$	$\frac{26}{33\cdot 7\cdot 45}$	0
f^6	0	0	0
f^7	0	0	0
f^8	$-\frac{1}{99}$	$\frac{2}{11\cdot 1485}$	$-\frac{1}{13\cdot 33\cdot 2079}$
f^9	$-\frac{2}{9\cdot 35}$	$-\frac{8}{11\cdot 45\cdot 273}$	$\frac{4}{11^2\cdot 13^2\cdot 3^3\cdot 7}$
f^{10}	$-\frac{1}{30\cdot 15}$	$-\frac{1}{11\cdot 2730}$	$-\frac{5}{13\cdot 33\cdot 9009}$
f^{11}	$\frac{4}{45\cdot 35}$	$\frac{2}{11\cdot 15\cdot 273}$	$\frac{8}{13^2\cdot 11^2\cdot 3^3\cdot 7}$
f^{12}	$\frac{1}{99}$	$\frac{8}{3\cdot 11\cdot 1485}$	$-\frac{5}{13\cdot 33\cdot 2079}$
f^{13}	$\frac{2}{63}$	$-\frac{2}{77\cdot 15}$	$\frac{4}{13\cdot 33\cdot 63}$

Alternatively, by rewriting equation 43 as:

$$P_k(\cos\omega) = \frac{4\pi}{2k+1}\left[Y_k^0 Y_k^0(i) + \sum_{q=1}^{k}(Y_k^{-q}Y_k^q(i) + Y_k^q Y_k^{-q}(i))\right] \tag{51}$$

where the spherical harmonics with (i) refers to ith-electron position, while the others refer to the ligands position. The Hamiltonian can be expressed as:

$$\hat{\mathcal{H}}_{\text{cf}}^{\text{Wybourne}} = -e\sum_{k,i}\Big[B_0^k C_0^{(k)}(i) + \sum_{q=1}^{k}(B_q^k(C_{-q}^{(k)}(i) + (-1)^q C_q^{(k)}(i)) + B'^k_q \mathrm{i}(C_{-q}^{(k)}(i) - (-1)^q C_q^{(k)}(i)))\Big] \tag{52}$$

where k has values 2 and 4 for d^n and 2, 4 and 6 for f^n configurations. The B_q^k and B'^k_q are the so-called crystal field coefficients (CFCs), that can be converted in the corresponding crystal field parameters B_q^k and B'^k_q, when solving the integral, by bringing them outside the matrix element, together with the radial parts of the wavefunctions $R_{nl}(r)$. Therefore, the CFPs contains the electron charge $-e$, a radial and an angular part. In this way, the tensor operators are left acting solely on the angular part. The use of the same notation for both coefficients and parameters is confusing but, unfortunately, it is what is done in literature.

Due to the operators' properties, the Stevens CFPs B_k^q (or A_k^q) in equations 47 and 48 are real-valued, whereas the Wybourne CFPs B_q^k are in general complex. All operators of the STO class are simply related to each other by specific numerical coefficients arising from the Wigner–Eckhart theorem while the same is not true for the operators of the TTO class [45]. Furthermore, while equations 47 and 48 are valide only in J(or L)-constants manifolds, expression 49 is applicable to the complete set of microstates.

In order to derive the relationship between the two set of operators we can express the O_k^q in terms of $T_q^{(k)}$:

$$O_k^0 = \lambda_{k0} T_0^{(k)} \tag{53}$$

$$O_k^q = \lambda_{kq}(T_{-q}^{(k)} + (-1)^q T_q^{(k)}) \tag{54}$$

$$O_k^{-q} = \mathrm{i}\lambda_{kq}(T_{-q}^{(k)} - (-1)^q T_q^{(k)}) \tag{55}$$

The conversion coefficients ($\lambda_{kq} = B_q^k / A_k^q \langle r^k \rangle$) are listed in table 4.

The formalism actually used in the integrals computation is the Wybourne one. The coefficients can either be fed into the function *MatrixH* (that performs the computation of energy levels and eigenfunctions) directly, through the dictionary *dic_bkq*, or using a PCM model, via

Table 4: The conversion factors λ_{kq} between the Stevens CFPs and the Wybourne CFPs, for $k = 2, 4, 6$[5].

q	2	4	6
0	2	8	16
1	$1/\sqrt{6}$	$2/\sqrt{5}$	$8/\sqrt{42}$
2	$2/\sqrt{6}$	$4/\sqrt{10}$	$16/\sqrt{105}$
3		$2/\sqrt{35}$	$8/\sqrt{105}$
4		$8/\sqrt{70}$	$16/3\sqrt{14}$
5			$8/3\sqrt{77}$
6			$16/\sqrt{231}$

the PCM item, applying the following equations[5]:

$$B_0^k = \sqrt{\frac{4\pi}{2k+1}}\langle r^k\rangle \sum_L \frac{Z_L e^2}{R_L^{k+1}} Y_k^0(\theta_L, \phi_L) \tag{56}$$

$$B_q^k = \sqrt{\frac{4\pi}{2k+1}}(-1)^q\langle r^k\rangle \sum_L \frac{Z_L e^2}{R_L^{k+1}} \mathrm{Re}Y_k^q(\theta_L, \phi_L) \tag{57}$$

$$B'^k_q = \sqrt{\frac{4\pi}{2k+1}}(-1)^q\langle r^k\rangle \sum_L \frac{Z_L e^2}{R_L^{k+1}} \mathrm{Im}Y_k^q(\theta_L, \phi_L) \tag{58}$$

Alternatively, the B_q^k can be computed from $A_k^q\langle r^k\rangle$ using the λ_{kq} factors, using the *from_Akqrk_to_Bkq*. A function for the calculation of the Stevens coefficients is also implemented in *calc_Akqrk* from PCM:

$$A_q^0 = \frac{4\pi}{2k+1} \sum_L \frac{Z_L e^2}{R_L^{k+1}} Z_{k0}(\theta_L, \phi_L) p_{k0}, \tag{59}$$

$$A_k^q = \frac{4\pi}{2k+1} \sum_L \frac{Z_L e^2}{R_L^{k+1}} Z_{kq}^{\mathrm{c}}(\theta_L, \phi_L) p_{kq}, \tag{60}$$

$$A_k^{-q} = \frac{4\pi}{2k+1} \sum_L \frac{Z_L e^2}{R_L^{k+1}} Z_{kq}^{\mathrm{s}}(\theta_L, \phi_L) p_{kq}, \tag{61}$$

where p_{kq} are tesseral hamronics' numerical prefactors (listed in table 5) and:

$$Z_{k0} = Y_k^0 \tag{62}$$

$$Z_{kq}^{\mathrm{c}} = \frac{1}{\sqrt{2}}\mathrm{Re}(Y_k^{-q} + (-1)^q Y_k^q) \tag{63}$$

$$Z_{kq}^{\mathrm{s}} = -\frac{1}{\sqrt{2}}\mathrm{Im}(Y_k^{-q} - (-1)^q Y_k^q) \tag{64}$$

All the equation presented so far have proven that it is possible to express the ligand-field potential as linear combination of spherical harmonics multiplied by coefficients, bearing information on radial part of the electronic wavefunction and ligand's wavefunction. This is true also at the one-electron level:

$$\langle l, m_l|V|l, m_l'\rangle = \sum_j c_{kq}^j \langle l, m_l|Y_k^q|l, m_l'\rangle \tag{65}$$

Table 5: Tesseral harmonics' prefactors p_{kq}, for $k = 2, 4, 6$[5].

q	2	4	6
0	$\frac{1}{4}\sqrt{\frac{5}{\pi}}$	$\frac{3}{16}\sqrt{\frac{1}{\pi}}$	$\frac{1}{32}\sqrt{\frac{13}{\pi}}$
1	$\frac{1}{2}\sqrt{\frac{15}{\pi}}$	$\frac{3}{4}\sqrt{\frac{5}{2\pi}}$	$\frac{1}{8}\sqrt{\frac{273}{4\pi}}$
2	$\frac{1}{4}\sqrt{\frac{15}{\pi}}$	$\frac{3}{8}\sqrt{\frac{5}{\pi}}$	$\frac{1}{64}\sqrt{\frac{2730}{\pi}}$
3		$\frac{3}{8}\sqrt{\frac{70}{\pi}}$	$\frac{1}{32}\sqrt{\frac{2730}{\pi}}$
4		$\frac{3}{16}\sqrt{\frac{35}{\pi}}$	$\frac{21}{32}\sqrt{\frac{13}{7\pi}}$
5			$\sqrt{\frac{9009}{512\pi}}$
6			$\frac{231}{64}\sqrt{\frac{26}{231\pi}}$

where j runs over the ligand indices and $\langle l, m_l|V|l, m_l'\rangle$ are one-electron ligand field integrals in the complex basis, elements of the one-electron ligand field matrix (V^{LF}). Since the $\langle l, m_l|Y_k^q|l, m_l'\rangle$ integrals can be easily solved using the tensor operator formalism (vide infra), what we get is a series of equations that express the c_{kq}^j coefficients in terms of $\langle l, m_l|V|l, m_l'\rangle$. This can be expressed in matrix form as:

$$\mathbf{c} = \mathbf{CM} \tag{66}$$

where $\mathbf{c}$ and $\mathbf{M}$ are vectors containing the $(l+1)(2l+1)$ independent coefficients and V^{LF} independent matrix elements respectively, and $\mathbf{C}$ is the $(l+1)(2l+1) \times (l+1)(2l+1)$ coefficient matrix. This implementation allows easily also the reverse process, i.e. compute the ligand field matrix from the coefficients via $\mathbf{M} = \mathbf{C}^{-1}\mathbf{c}$. In order for the procedure to properly work, the elements in $\mathbf{c}$, $\mathbf{C}$ and $\mathbf{M}$ have to be coherently ordered.

Since, for NJA-CFS, we are interested in real-valued CF coefficients, I implemented this procedure (in the *from_Vint_to_Bkq_2* function) using the ligand-field matrix elements in the real basis and the $\mathbf{C}$ matrix obtained exploiting the relations between the real basis and the complex basis:

$$|0\rangle = |\sigma\rangle \tag{67}$$

$$|1\rangle = -\frac{\mathrm{i}}{\sqrt{2}}|\pi_s\rangle - \frac{1}{\sqrt{2}}|\pi_c\rangle \tag{68}$$

$$|-1\rangle = -\frac{\mathrm{i}}{\sqrt{2}}|\pi_s\rangle + \frac{1}{\sqrt{2}}|\pi_c\rangle \tag{69}$$

$$|2\rangle = \frac{\mathrm{i}}{\sqrt{2}}|\delta_s\rangle + \frac{1}{\sqrt{2}}|\delta_c\rangle \tag{70}$$

$$|-2\rangle = -\frac{\mathrm{i}}{\sqrt{2}}|\delta_s\rangle + \frac{1}{\sqrt{2}}|\delta_c\rangle \tag{71}$$

$$|3\rangle = -\frac{\mathrm{i}}{\sqrt{2}}|\phi_s\rangle - \frac{1}{\sqrt{2}}|\phi_c\rangle \tag{72}$$

$$|-3\rangle = -\frac{\mathrm{i}}{\sqrt{2}}|\phi_s\rangle + \frac{1}{\sqrt{2}}|\phi_c\rangle \tag{73}$$

where the complex basis is indicated as $|l, m_l\rangle = |m_l\rangle$, while the real basis is indicated with

$|l,u\rangle = |u\rangle$ (with $u = \sigma, \pi_s, \pi_c, \delta_s, \delta_c, \phi_s, \phi_c$). I chose this convention for the real orbitals since it is the one used in the papers on the subject. The expression are the same as the one reported in table 2 of [46] for d^n configurations and table 3 of [47] for f^n configurations.

The V^{LF} matrix elements can be computed ab initio through what is called: ab initio ligand field theory (AILFT). The one-electron ligand field matrices of a AILFT from an ORCA CASSCF calculation are organized as follows: for d^n configurations ($l = 2$), the elements are displayed with $m_l = [0, 1, -1, 2, -2]$, or, using the ORCA's orbital labels, as $[d_{z^2}, d_{xz}, d_{yz}, d_{x^2-y^2}, d_{xy}]$, while for f^n configurations ($l = 3$), the elements are displayed with $m_l = [0, 1, -1, 2, -2, 3, -3]$, or, using the ORCA's orbital labels, as $[f_0, f_{+1}, f_{-1}, f_{+2}, f_{-2}, f_{+3}, f_{-3}]$. In the notation used in equations 67, the orbitals are ordered as: $[\sigma, \pi_c, \pi_s, \delta_c, \delta_s, \phi_c, \phi_s]$. An important note: the phases of f_{+3} and f_{-3} orbitals in ORCA are different from the ones generally used, so in NJA-CFS a sign change is performed for the mixed integrals involving those orbitals. Those matrix are obtained using the *actorb* flag in the ORCA *%casscf* block (see the ORCA manual for further details).

Another alternative to the (electrostatically-based) crystal field models, is the angular overlap model (AOM). Analogously to the other ligand-field treatment, also this one is based on one-electron operators. It assumes that the antibonding-orbital energies (E^*) are determined by covalent perturbation weak enough to be proportional to the square of appropriate overlap integrals:

$$E^* = (A_t^d)^2 e_t \tag{74}$$

where d indicates the metal d orbital and t the symmetry of the ligand orbital (i.e. σ, π...). A_t^d is an angular factor expressing the fact that the chosen global coordinate frame of the metal atom is not generally coincident with the local frame to which the symmetry of the ligand functions are referred. e_t is the parameter enclosing the information on the overlap integral between the metal and ligand orbitals and it is different for each ligand type and for each bonding mode. Contributes arising from the different ligands to this antibonding energy are considered additive. A general expression for the one-electron matrix elements within the AOM model is the following:

$$\langle l,u|V|l,v\rangle = \sum_{j}^{\text{ligands}} \sum_{t}^{\text{modes}} D_{ut}^{l}(j) D_{vt}^{l}(j) e_t(j) \tag{75}$$

where D_{ut}^{l} are the matrix elements of the transformation of the real metal orbitals into each ligand coordinate frame. In direction cosines, the D matrix elements for the modes e_σ, e_{π_c} and e_{π_s} is the following, for $l = 2$ (equation 9 of [46]):

$$\begin{bmatrix} (1/2)(3\gamma_3^2 - 1) & \sqrt{3}\alpha_3\gamma_3 & \sqrt{3}\beta_3\gamma_3 \\ \sqrt{3}\gamma_1\gamma_3 & \alpha_1\gamma_3 + \alpha_3\gamma_1 & \beta_1\gamma_3 + \beta_3\gamma_1 \\ \sqrt{3}\gamma_2\gamma_3 & \alpha_2\gamma_3 + \alpha_3\gamma_2 & \beta_2\gamma_3 + \beta_3\gamma_2 \\ \sqrt{3}\gamma_1\gamma_2 & \alpha_1\gamma_2 + \alpha_2\gamma_1 & \beta_1\gamma_2 + \beta_2\gamma 1 \\ (1/2)\sqrt{3}(\gamma_1^2 - \gamma_2^2) & \alpha_1\gamma_1 - \alpha_2\gamma_2 & \beta_1\gamma_1 - \beta_2\gamma_2 \end{bmatrix} \tag{76}$$

and for $l = 3$ (table 1 of [47]):

$$\begin{bmatrix}
(1/2)\gamma_3(5\gamma_3^2-3) & (1/2)\sqrt{3/2}\beta_3(5\gamma_3^2-1) & (1/2)\sqrt{3/2}\alpha_3(5\gamma_3^2-1) \\
(1/4)\sqrt{6}\gamma_2(5\gamma_3^2-1) & (1/4)\beta_2(5\gamma_3^2-1)+(5/2)\gamma_2\gamma_3\beta_3 & (1/4)\alpha_2(5\gamma_3^2-1)+(5/2)\gamma_2\gamma_3\alpha_3 \\
(1/4)\sqrt{6}\gamma_1(5\gamma_3^2-1) & (1/4)\beta_1(5\gamma_3^2-1)+(5/2)\gamma_1\gamma_3\beta_3 & (1/4)\alpha_1(5\gamma_3^2-1)+(5/2)\gamma_1\gamma_3\alpha_3 \\
\sqrt{15}\gamma_1\gamma_2\gamma_3 & \sqrt{(5/2)}(\beta_1\gamma_2\gamma_3+\beta_2\gamma_1\gamma_3+\beta_3\gamma_1\gamma_2) & \sqrt{(5/2)}(\alpha_1\gamma_2\gamma_3+\alpha_2\gamma_1\gamma_3+\alpha_3\gamma_1\gamma_2) \\
(1/2)\sqrt{15}\gamma_3(\gamma_1^2-\gamma_2^2) & \sqrt{5/8}(\gamma_1^2\beta_3-\gamma_2^2\beta_3+2\beta_1\gamma_1\gamma_3-2\beta_2\gamma_2\gamma_3) & \sqrt{5/8}(\gamma_1^2\alpha_3-\gamma_2^2\alpha_3+2\alpha_1\gamma_1\gamma_3-2\alpha_2\gamma_2\gamma_3) \\
(1/4)\sqrt{10}\gamma_2(3\gamma_1^2-\gamma_2^2) & (1/4)\sqrt{15}\beta_2(\gamma_1^2-\gamma_2^2)+(1/2)\sqrt{15}\gamma_1\gamma_2\beta_1 & (1/4)\sqrt{15}\alpha_2(\gamma_1^2-\gamma_2^2)+(1/2)\sqrt{15}\gamma_1\gamma_2\alpha_1 \\
(1/4)\sqrt{10}\gamma_1(\gamma_1^2-3\gamma_2^2) & (1/4)\sqrt{15}\beta_1(\gamma_1^2-\gamma_2^2)+(1/2)\sqrt{15}\gamma_1\gamma_2\beta_2 & (1/4)\sqrt{15}\alpha_1(\gamma_1^2-\gamma_2^2)+(1/2)\sqrt{15}\gamma_1\gamma_2\alpha_2
\end{bmatrix} \tag{77}$$

and θ, ϕ and χ are the ligand angles. Based on this model, we can derive an additional way to compute the CFPs. Using equation 75 one can compute the V^{LF} matrix (implmented in *from_AOM_to_Vint*), and from there one can proceed with equation 66.

5.3 Crystal field rotation

Since the (complex) crystal field coefficients in the Wybourne convention can be expressed in terms of raal or complex spherical harmonics, they maintain the same transformation properties. Consequently, an arbitrary rotation of the CFPs in NJA-CFS is performed using the Wigner D-matrix (**D**), either in Euler angles (with *rota_LF* function) or in quaternions (with *rota_LF_quat* function), by applying the general formula for the spherical harmonics Y_m^l subjected to an arbitrary rotation $\mathcal{R}$:

$$Y_m^l = \sum_{m'=-l}^{+l} D_{m'm}^l(\mathcal{R}) Y_{m'}^l \tag{78}$$

For a 3-dimensional rotation defined by the Euler angles α, β and γ, in the z-y-z convention and active interpretation, the elements of the unitary square matrix of dimension $2k+1$ are defined as:

$$D_{q'q}^k(\alpha,\beta,\gamma) = e^{-\mathrm{i}q'\alpha} d_{q'q}^k(\beta) e^{-\mathrm{i}q\gamma} \tag{79}$$

where q runs from $-k$ to k and $d_{q'q}^k(\beta)$ is the element of the Wigner's (small) d-matrix, defined as:

$$\begin{aligned} d_{q'q}^k(\beta) &= [(k+q')!(k-q')!(k+q)!(k-q)!]^{1/2} \\ \times \sum_{s=s_{\min}}^{s_{\max}} &\left[\frac{(-1)^{q'+q+s} \left(\cos\frac{\beta}{2}\right)^{2k+q-q'-2s} \left(\sin\frac{\beta}{2}\right)^{q'-q+2s}}{(k+q-s)!s!(q'-q+s)!(k-q'-s)!} \right] \end{aligned} \tag{80}$$

where the sum over s runs over such values that the factorials are nonnegative, i.e. $s_{\min} = \max(0, q-q')$ and $s_{\max} = \min(k+q, k-q')$.

The analogous of this equations, in quaternions, are computed as described by Lynden-Bell and Stone in [48]. In brief, the derivation starts from the definition of the simplest Wigner D-matrix, for $k = 1/2$. Expressing $D^{1/2}$ as:

$$\mathbf{D}^{1/2} = \begin{pmatrix} A & B \\ -B^* & A^* \end{pmatrix} \tag{81}$$

where the elements are complex functions of α, β and γ, which satisfy:

$$AA^* + BB^* = 1 \tag{82}$$

The connection among these matrix elements and the quaternion description can be found by expanding the matrix as:

$$\mathbf{D}^{1/2} = A_{\mathrm{Re}}\mathbf{I} - B_{\mathrm{Im}}\mathbf{i} - B_{\mathrm{Re}}\mathbf{j} - A_{\mathrm{Im}}\mathbf{k} \tag{83}$$

where $\mathbf{I}$ is the 2×2 unit matrix, A_{Re} and A_{Im} (B_{Re} and B_{Im}) are the real and imaginary parts

of A (B) respectively, and the 2×2 matrices $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ are:

$$\mathbf{i} = \begin{pmatrix} 0 & -\mathrm{i} \\ -\mathrm{i} & 0 \end{pmatrix} \tag{84}$$

$$\mathbf{j} = \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix} \tag{85}$$

$$\mathbf{k} = \begin{pmatrix} -\mathrm{i} & 0 \\ 0 & \mathrm{i} \end{pmatrix} \tag{86}$$

which satisfy the multiplication rules:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -\mathbf{I} \tag{87}$$

The parameters of a normalized quaternion $(q_0, \mathbf{q})$ are related to A and B as:

$$A = q_0 - \mathrm{i}q_3 \tag{88}$$

$$B = -q_2 - \mathrm{i}q_1 \tag{89}$$

Given that any rotation matrix can be built from $\mathbf{D}^{1/2}$. Since an eigenfunction of angular momentum with $l = m = L$ (where $L = k$ in this case) can be constructed from $2L$ spin $= 1/2$ functions, the top left element of the Wigner rotation matrix in the customary arrangement (labeling the rows and columns in the order k, $k-1$, ..., $-k$) is given by:

$$D^k_{kk} = A^{2k} \tag{90}$$

and the rest of the matrix can be constructed using the operators $\hat{\mathfrak{R}}$ and $\hat{\mathfrak{B}}$ for the generation of the matrix elements on the right and beneath respectively:

$$\hat{\mathfrak{R}} D^k_{q'q} = (k(k+1) - q(q+1))^{1/2} D^k_{q'q-1} \tag{91}$$

$$\hat{\mathfrak{B}} D^k_{q'q} = (k(k+1) - q'(q'+1))^{1/2} D^k_{q'-1q} \tag{92}$$

The effect of $\hat{\mathfrak{R}}$ and $\hat{\mathfrak{B}}$ on A, A^*, B and B^* can be deduced from the matrix in 81. Additionally, since both operators are differential operators, they obey to the usual rules for differentiation of products. The following relation was also used in the matrices construction:

$$D^k_{-q'-q} = (-1)^{q'-q} (D^k_{q'q})^* \tag{93}$$

The matrices for $k = 2$ and $k = 3$ are reported in tables 6 and 7, while matrices for $k = 4$ and $k = 6$ are saved in dedicated files, i.e. *tab_wignerDquat.txt* and *tab_wignerDquat_coeff_t.txt*, in the NJA-CFS's *tables* folder.

In order to apply this matrices to the different orders of CFPs, the real-valued Wigner coefficients are organized in a complex array ($\mathbf{B}^k$):

$$\mathbf{B}^k = [\mathcal{B}^k_k, \mathcal{B}^k_{k-1}, ..., \mathcal{B}^k_{-k+1}, \mathcal{B}^k_{-k}] \tag{94}$$

where, the complex-valued Wigner coefficients are defined as:

$$\mathcal{B}_q^k = \begin{cases} (B_{|q|}^k + \mathrm{i}B'^k_{|q|}) & \text{for} \quad q < 0 \\ B_0^k & \text{for} \quad q = 0 \\ (-1)^{|q|}(B_{|q|}^k - \mathrm{i}B'^k_{|q|}) & \text{for} \quad q > 0 \end{cases} \tag{95}$$

The rotation matrix is applied (according to equation 78) as follows:

$$\mathbf{B}^{\mathrm{rot},k} = \mathbf{D}^k \mathbf{B}^k \tag{96}$$

where the elements of $\mathbf{D}^k$ are organized according to the elements of $\mathbf{B}^k$. The real-valued rotated CFPs B_q^k and B'^k_q are defined as real and imaginary part respectively of the $q = -k, ..., 0$ elements of $\mathbf{B}^{\mathrm{rot},k}$.

Alternatively, the rotation can be performed on the $(2k+1) \times (2k+1)$ one-electron ligand field matrix, i.e. the AILFT matrix from an ORCA CASSCF calculation, with the *rotate_dicV* function. Also in this case the Wigner D-matrices are used. The elements of V^{LF} are real-valued, while the elements of $\mathbf{D}^k$ are complex. Therefore, for the application of a rotation to V^{LF}, $\mathbf{D}^k$ are unitary transformed, using:

$$\mathbf{U} = \begin{pmatrix} 1/\sqrt{2} & 0 & 0 & 0 & -\mathrm{i}/\sqrt{2} \\ 0 & -1/\sqrt{2} & 0 & \mathrm{i}/\sqrt{2} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & \mathrm{i}/\sqrt{2} & 0 \\ 1/\sqrt{2} & 0 & 0 & 0 & \mathrm{i}/\sqrt{2} \end{pmatrix} \tag{97}$$

for $k = 2$ (for d^n configurations), and:

$$\mathbf{U} = \begin{pmatrix} -1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & \mathrm{i}/\sqrt{2} \\ 0 & 1/\sqrt{2} & 0 & 0 & 0 & -\mathrm{i}/\sqrt{2} & 0 \\ 0 & 0 & -1/\sqrt{2} & 0 & \mathrm{i}/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & 0 & \mathrm{i}/\sqrt{2} & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & 0 & 0 & \mathrm{i}/\sqrt{2} & 0 \\ 1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & \mathrm{i}/\sqrt{2} \end{pmatrix} \tag{98}$$

for $k = 3$ (for f^n configurations), as:

$$\mathcal{D}^k = \mathbf{U}^\dagger \mathbf{D}^k \mathbf{U} \tag{99}$$

At this point, each element of V^{LF} can be converted, for an arbitrary rotation $\mathcal{R}$ either defined in terms of Euler angles or quaternions, as follows:

$$V_{MM'}^{\mathrm{LF,rot},k} = \sum_{m'} \sum_{m''} [\mathcal{D}_{Mm'}^k(\mathcal{R})][\mathcal{D}_{M'm''}^k(\mathcal{R})^*] V_{m'm''}^{\mathrm{LF},k} \tag{100}$$

for M, m', M' and m'' in range $-k, ..., k$, derived from the application of equation 78.

Table 6: Wigner D matrix in quaternions, for $k = 2$. The missing elements can be derived with equation 93. $Z = q_0^2 - q_1^2 - q_2^2 + q_3^2 = AA^* - BB^*$.

q'	q	matrix element
2	2	A^4
2	1	$2A^3B$
2	0	$\sqrt{6}A^2B^2$
1	2	$-2A^3B^*$
1	1	$A^2(2Z-1)$
1	0	$\sqrt{6}ABZ$
0	2	$\sqrt{6}A^2B^{*2}$
0	1	$-\sqrt{6}AB^*Z$
0	0	$1/2(3Z^2-1)$
-1	2	$-2AB^{*3}$
-1	1	$B^{*2}(2Z+1)$
-2	2	B^{*4}
-2	1	$-2A^*B^{*3}$

Table 7: Wigner D matrix in quaternions, for $k = 3$. The missing elements can be derived with equation 93. $Z = q_0^2 - q_1^2 - q_2^2 + q_3^2 = AA^* - BB^*$.

q'	q	matrix element
3	3	A^6
3	2	$\sqrt{6}BA^5$
3	1	$\sqrt{15}B^2A^4$
3	2	$2\sqrt{5}B^3A^3$
2	3	$-\sqrt{6}A^5B^*$
2	2	$A^4(3Z-2)$
2	1	$(1/2)\sqrt{10}BA^3(3Z-1)$
2	0	$\sqrt{30}B^2A^2Z$
1	3	$\sqrt{15}A^4B^{*2}$
1	2	$(1/2)\sqrt{10}A^3B^*(1-3Z)$
1	1	$(1/4)A^2(15Z^2-10Z-1)$
1	0	$(1/2)\sqrt{3}AB(5Z^2-1)$
0	3	$-2\sqrt{5}A^3B^{*3}$
0	2	$\sqrt{30}A^2B^{*2}Z$
0	1	$(1/2)\sqrt{3}AB^*(1-5Z^2)$
0	0	$(1/2)(5Z^3-3Z)$
-1	3	$\sqrt{15}A^2B^{*4}$
-1	2	$-(1/2)\sqrt{10}AB^{*3}(1+3Z)$
-1	1	$(1/4)B^{*2}(15Z^2+10Z-1)$
-2	3	$-\sqrt{6}AB^{*5}$
-2	2	$B^{*4}(3Z+2)$
-2	1	$-(1/2)\sqrt{10}A^*B^{*3}(3Z+1)$
-3	3	B^{*6}
-3	2	$-\sqrt{6}A^*B^{*5}$
-3	1	$\sqrt{15}B^{*4}A^{*2}$

5.4 Integrals computation

5.4.1 Electron-electron interaction

The matrix elements are J and M_J independent[16]:

$$\langle l^n \nu LSJM_J|\hat{\mathcal{H}}_{e-e}|l^n \nu' L'S'J'M'_J\rangle = \langle l^n \nu LS\|\hat{\mathcal{H}}_{e-e}\|l^n \nu' L'S'\rangle \delta_{J,J'}\delta_{M'_J,M'_J} \tag{101}$$

where the seniority quantum number ν is made explicit among the possible α and:

$$\begin{aligned}\hat{\mathcal{H}}_{e-e} &= \left(\frac{e^2}{4\pi\varepsilon_0}\right)\sum_i^n\sum_{j>i}^n \frac{1}{r_{ij}} \\ &= \left(\frac{e^2}{4\pi\varepsilon_0}\right)\sum_i^n\sum_{j>i}^n\sum_{k=0}^\infty [r_>^{-(k+1)}\cdot r_<^k](C_1^{(k)}\cdot C_2^{(k)})\end{aligned} \tag{102}$$

with $k = 0, 2, 4$ for d^n configurations and $k = 0, 2, 4, 6$ for f^n configurations.

For a two-electron system the equation is simplified in:

$$\langle l^2 \nu LS \| \hat{\mathcal{H}}_{e-e} \| l^2 \nu' L' S' \rangle = \sum_{k=0}^{\infty} F_{ll}^k [\langle l \| C^{(k)} \| l \rangle]^2 (-1)^L \left\{ \begin{array}{ccc} l & l & k \\ l & l & L \end{array} \right\} \tag{103}$$

where

$$F_{ll}^k = \int \int \frac{r_<^k}{r_>^{k+1}} R_l^2(r_1) R_l^2(r_2) r_1^2 r_2^2 dr_1 dr_2 \tag{104}$$

and the reduced matrix elements of the Racah operator are defined as:

$$\langle l \| C^{(k)} \| l \rangle = (-1)^l [(2l+1)(1l'+1)]^{1/2} \left(\begin{array}{ccc} l & k & l' \\ 0 & 0 & 0 \end{array} \right). \tag{105}$$

The 3-j symbol $\left(\begin{array}{ccc} a & b & c \\ A & B & C \end{array} \right)$ represent the coupling of two angular momenta and can be computed using the Racah's closed form[28]:

$$\begin{aligned} \left(\begin{array}{ccc} a & b & c \\ A & B & C \end{array} \right) &= (-1)^{a-b-C} \left[\frac{(a+b-c)!(b+c-a)!(c+a-b)!}{(a+b+c+1)!} \right] \\ &\times [(a+A)!(a-A)!(b+B)!(b-B)!(c+C)!(c-C)!]^{1/2} \\ &\times \sum_{n=n_{\min}}^{n_{\max}} (-1)^n [n!(c-b+n+A)!(c-a+n-B)!(a+b-c-n)! \\ &\times (a-n-A)!(b-n+B)!]^{-1} \end{aligned} \tag{106}$$

with:

$$\begin{aligned} n_{\min} &= \max\{0; -c+b-A; -c+a+B\} \\ n_{\max} &= \min\{a+b-c; b+B; a-A\} \end{aligned}$$

For $n > 2$ and d^n configurations, the equation is the following:

$$\langle l^2 \nu LS \| \hat{\mathcal{H}}_{e-e} \| l^2 \nu' L' S' \rangle = \delta_{L,L'} \delta_{S,S'} \sum_{k=0,2,4} F_{ll}^k \cdot c^k(l^n \nu \nu' LS) \tag{107}$$

with the angular coefficients, for $k > 0$:

$$\begin{aligned} c^k(l^n \nu \nu' LS) &= \frac{1}{2} \langle l \| C^{(k)} \| l \rangle^2 \\ &\times \left\{ \frac{1}{2L+1} \sum_{\nu'' L''} \langle l^n \nu LS \| U^{(k)} \| l^n \nu'' L'' S \rangle \langle l^n \nu' LS \| U^{(k)} \| l^n \nu'' L'' S \rangle - \frac{n}{2l+1} \delta_{\nu,\nu'} \right\} \end{aligned} \tag{108}$$

and for $k = 0$:

$$c^0(l^n \nu \nu' LS) = \frac{n(n-1)}{2} \delta_{\nu,\nu'} \tag{109}$$

The parameters that have to be passed to the program are the Slater-Condon parameters

F^k. Care must be taken for the superscript k, since:

$$\begin{aligned} F^0 &= F_0 \\ F^2 &= 49F_2 \\ F^4 &= 441F_4 \end{aligned} \tag{110}$$

for d^n and:

$$\begin{aligned} F^0 &= F_0 \\ F^2 &= 225F_2 \\ F^4 &= 1089F_4 \\ F^6 &= 184041F_6/25 \end{aligned} \tag{111}$$

for f^n.

The integrals computation for $n > 2$ and f^n configurations is more complex[30, 44]. Equation 107 is usually rewritten as:

$$\langle l^2 \nu LS \| \hat{\mathcal{H}}_{e-e} \| l^2 \nu' LS \rangle = \sum_{k=0,2,4,6} f_k F^k = \sum_{k=0,2,4,6} f^k F_k \tag{112}$$

with the F^k defined as 111, and $f^k = D_k f_k$, with D_k are those that relate F^k to F_k. Although the operator $(C_1^{(k)} \cdot C_2^{(k)})$ in equation 102 is scalar with respect to R_3, it does not have the transformation properties of the group R_7 and G_2 (used for the states classification, see appendix 5.1) and therefore we have to use linear combinations of such operators. The matrix elements are re-defined as:

$$\hat{\mathcal{H}}_{e-e} = \sum_{k=0,1,2,3} e_k E^k \tag{113}$$

where e_k are related to the f^k by the expressions:

$$\begin{aligned} e_0 &= f^0 = n(n-1) \\ e_1 &= \frac{9}{7} f^0 + \frac{1}{42} f^2 + \frac{1}{77} f^4 + \frac{1}{462} f^6 \\ e_2 &= \frac{143}{42} f^2 - \frac{130}{77} f^4 + \frac{35}{462} f^6 \\ e_3 &= \frac{11}{42} f^2 + \frac{4}{77} f^4 - \frac{7}{462} f^6 \end{aligned} \tag{114}$$

whereas the E^ks are linear combinations of the F_ks:

$$\begin{aligned} E^0 &= F_0 - 10F_2 - 33F_4 - 286F_6 \\ E^1 &= \frac{70F_2 + 231F_4 + 2002F_6}{9} \\ E^2 &= \frac{F_2 - 3F_4 + 7F_6}{9} \\ E^3 &= \frac{5F_2 + 6F_4 - 91F_6}{3} \end{aligned} \tag{115}$$

The e_is can be computed with the following procedure:

$$e_0 = n(n-1)/2 \tag{116}$$

$$e_1(f^n USL) = 9(n-v)/2 + v(v+1)/4 - S(S+1) \tag{117}$$

$$e_2(f^n USLU'SL) = \pm e_2(WUL, WU'L) = \pm \sum_\gamma x_\gamma(W, UU')(U|\chi_\gamma|U') \tag{118}$$

$$e_3(f^n vUSLv'U'SL) = y(f^n, vSU, v'SU')(U|\phi(L)|U') - \Omega\delta_{UU',vv'} \tag{119}$$

where all the factors (and additional closed forms) necessary for the computation of e_2 and e_3 for all f^n configurations are available in [30]. NJA reads these factors from a table, however the routine for their calculation is available in the software.

The parameters that have to be passed to the program for the computation of this contributions are the Slater-Condon parameters F^k. The parameter F^0 can be omitted since we are interested in the splitting of the terms, not in their absolute position on the energy scale.

5.4.2 Spin-orbit coupling

The spin-orbit coupling contribution can be simply computed as:

$$\langle l^n \nu LSJM_J|\hat{\mathcal{H}}_{so}|l^n \nu' L'S'J'M'_J\rangle = \kappa\langle l^n \nu LS\|\hat{\mathcal{H}}_{so}\|l^n \nu' L'S'\rangle\delta_{J,J'}\delta_{M'_J,M'_J} \tag{120}$$

This reduced matrix element can be computed as:

$$\begin{aligned}\langle l^n \nu LS\|\hat{\mathcal{H}}_{so}\|l^n \nu' L'S'\rangle = \zeta(-1)^{J+L+S'}[l(l+1)(2l+1)]^{1/2} \\ \times \begin{Bmatrix} L & L' & 1 \\ S' & S & J \end{Bmatrix} \langle l^n \nu LS\|V^{(11)}\|l^n \nu' L'S'\rangle\end{aligned} \tag{121}$$

where the RMEs $\langle l^n \nu LS\|V^{(11)}\|l^n \nu' L'S'\rangle$ are computed according to equation 36.

In this case the parameters to be defined are the spin-orbit coupling coefficient ζ and the orbital reduction factor κ.

5.4.3 Crystal-Field splitting

In this case, even if different kind of parameters can be fed into the program (see appendix 5.2), the formalism used for the integral resolution is the Wybourne one[44]. The crystal-field integrals in the Racah's tensor operator formalism, according to equation 52, can be expressed as:

$$\begin{aligned}\langle \alpha SLJM_J|\hat{\mathcal{H}}_{cf}|\alpha' S'L'J'M'_J\rangle = \langle \alpha SLJM_J| - e\sum_{i,k}\Big[B^k_0 C^{(k)}_0(i) + \\ + \sum_{q=1}^{k}(B^k_q(C^{(k)}_{-q}(i) + (-1)^q C^{(k)}_q(i)) + B'^k_q \mathrm{i}(C^{(k)}_{-q}(i) - (-1)^q C^{(k)}_q(i)))\Big]|\alpha' S'L'J'M'_J\rangle\end{aligned} \tag{122}$$

where α represent the additional set of quantum numbers used and k has values 2 and 4 for d^n and 2, 4 and 6 for f^n configurations. The B^k_q and B'^k_q are the so-called crystal field coefficients (CFCs), that can be converted in the corresponding crystal field parameters B^k_q and B'^k_q, when

solving the integral, by bringing them outside the matrix element, together with the radial parts of the wavefunctions $R_{nl}(r)$. In this way, the tensor operators are left acting solely on the angular part. The use of the same notation for both coefficients and parameters is confusing but, unfortunately, it is what is done in literature.

The CFPs are defined as:

$$B_0^k = \int_{r=0}^{\infty} R_{nl}^2(r) r^k dr \sqrt{\frac{4\pi}{2k+1}} \sum_L Z_{k0}^c(\theta_L) \frac{Z_L e^2}{R_L^{k+1}} \tag{123}$$

$$B_q^k = \int_{r=0}^{\infty} R_{nl}^2(r) r^k dr \sqrt{\frac{4\pi}{2k+1}} \frac{1}{\sqrt{2}} \sum_L Z_{kq}^c(\theta_L, \phi_L) \frac{Z_L e^2}{R_L^{k+1}} \tag{124}$$

$$B'^k_q = \int_{r=0}^{\infty} R_{nl}^2(r) r^k dr \sqrt{\frac{4\pi}{2k+1}} \frac{1}{\sqrt{2}} \sum_L Z_{kq}^s(\theta_L, \phi_L) \frac{Z_L e^2}{R_L^{k+1}} \tag{125}$$

where the sum over L runs over the ligands, represented in the point charge model. The $Z_{kq}(\theta_L, \phi_L)$ are the tesseral harmonics of degree k and order q, defined at the polar (colatitudinal) angle θ_L and azimutal (longitudinal) angle ϕ_L at the L-th ligand position. The very same equations can be expressed in term of the (complex) spherical harmonics Y_k^q. Z_L and R_L are the point charge and the distance from the metal ion of the L-th ligand respectively. For a continuous distribution of charges, an integral over the charge density can be used. The integrals over the radial part, dedicated parametrizations are available, generally represented as $\langle r^k \rangle$.

The angular parts (containing the electron coordinates) of the matrix elements expressed in equation 122 can be calculated using the tensor operators properties, by rewriting the the summation over i in terms of unit tensor U_q^k:

$$\langle \alpha SLJM_J | \sum_i C_q^{(k)}(i) | \alpha' S'L'J'M'_J \rangle = \langle \alpha SLJM_J | U_q^{(k)} | \alpha' S'L'J'M'_J \rangle \langle l \| C^{(k)} \| l' \rangle \tag{126}$$

The U_q^k matrix elements are diagonal in S.

Applying the Wigner-Eckart theorem, and solving the RME of $C^{(k)}$, equation 126 becomes:

$$\begin{gathered} \langle \alpha SLJM_J | \sum_i C_q^{(k)}(i) | \alpha' SL'J'M'_J \rangle = \\ (-1)^{2J-M_J+S+L'+k+l} [(2J+1)(2J'+1)]^{1/2} [(2l+1)(2l'+1)]^{1/2} \\ \times \begin{pmatrix} l & k & l' \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} J & k & J' \\ -M_J & q & M'_J \end{pmatrix} \begin{Bmatrix} J & J' & k \\ L' & L & S \end{Bmatrix} \langle \alpha SL \| U^{(k)} \| \alpha' SL' \rangle \end{gathered} \tag{127}$$

The RMEs $\langle \alpha SL \| U^{(k)} \| \alpha' SL' \rangle$ can be solved using the cfps and the equations in appendix 5.1.1.

5.4.4 Zeeman interaction

The contribution from the Zeeman interaction with the magnetic field vector is expressed as:

$$\langle l^n \nu LSJM_J|\hat{\mathcal{H}}_Z|l^n \nu' L'S'J'M'_J\rangle = \mu_B \hbar^{-1} \sum_{q=-1}^{1} (-1)^q B^{(1)}_{-q}$$
$$\times (-1)^{J-M} \begin{pmatrix} J & 1 & J' \\ -M_J & q & M'_J \end{pmatrix} \langle l^n \nu LSJ \| (\kappa L^{(1)} + g_e S^{(1)} \| l^n \nu' L'S'J'\rangle \tag{128}$$

where the RMEs are computed as:

$$\langle l^n \nu LSJ \| (\kappa L^{(1)} + g_e S^{(1)} \| l^n \nu' L'S'J'\rangle = \delta_{L,L'}\delta_{S,S'}\hbar[(2J+1)(2J'+1)]^{1/2}$$
$$\times \left\{ k[L(L+1)(2L+1)]^{1/2}(-1)^{L+S+J+1} \begin{Bmatrix} J' & J & 1 \\ L & L & S \end{Bmatrix} + g_e[S(S+1)(2S+1)]^{1/2} \right.$$
$$\left. \times (-1)^{L+L'+2S'+J+J'}(-1)^{L+S+J+1} \begin{Bmatrix} J' & J & 1 \\ S & S & L \end{Bmatrix} \right\} \tag{129}$$

with:

$$B^{(1)}_{+1} = -\frac{1}{\sqrt{2}}(B_x + \mathrm{i}B_y)$$
$$B^{(1)}_{-1} = \frac{1}{\sqrt{2}}(B_x - \mathrm{i}B_y)$$
$$B^{(1)}_{0} = B_z$$

In this case the parameters needed are the magnetic field vector $\mathbf{B} = [B_x, B_y, B_z]$ and the orbital reduction factor κ.

6 Fractional Parentage Coefficients

The procedure used for computing the cfp is taken from the Nielson PhD thesis[33].

According to the group-theoretical classification of states:

- the single-electron wavefunction, with quantum numbers: $S = 1/2$, $W = (100)$, $U = (10)$, $L = 3$, m_l, m_s, correspond to the irreducible representation Γ_f,
- the antisymmetric states of $n-1$ electrons, with quantum numbers: S', W', U', α', L', M'_L, M'_S, constitute the basis of the irreducible representation $\Gamma^{(n-1)}_A$,
- the antisymmetric states of n electrons, with quantum numbers: S, W, U, α, L, M_L, M_S, constitute the basis of the irreducible representation $\Gamma^{(n)}_A$.

The objective is to find the reduction coefficients of the Kronecker product $\Gamma^{(n-1)}_A \times \Gamma_f$ relative to the antisymmetric part $\Gamma^{(n)}_A$, indicated in Nielson's notation as:

$$b_n(S, W, U, \alpha, L, M_L, M_S; S', W', U', \alpha', L', M'_L, M'_S, m_l, m_s) \tag{130}$$

These coefficients can be factorized according to the Racah's theorem as follows:

$$b_n(S,W,U,\alpha,L,M_L,M_S;S',W',U',\alpha',L',M'_L,M'_S,m_l,m_s) = \\ C^{S'\ 1/2\ S}_{n,W'\ (100)\ W} C^{W'\ (100)\ W}_{U'\ (10)\ U} C^{U'\ (10)\ U}_{\alpha' L'\ 3\ \alpha L} C^{L'\ 3\ L}_{M'_L\ m_l\ M_L} C^{S'\ 1/2\ S}_{M'_S\ m_s\ M_S} \tag{131}$$

where the last two coefficients are Clebsch-Gordan coefficients and the product of the other three compose the cfp:

$$a_n(S,W,U,\alpha,L;S',W',U',\alpha',L') = C^{S'\ 1/2\ S}_{n,W'\ (100)\ W} C^{W'\ (100)\ W}_{U'\ (10)\ U} C^{U'\ (10)\ U}_{\alpha' L'\ 3\ \alpha L} \tag{132}$$

In Racah's notation (used from now on):

$$(l^{n-1}(\alpha' \nu' S' L') l S L | \} l^n \alpha \nu S L) \equiv a_n(S,W,U,\alpha,L;S',W',U',\alpha',L') \tag{133}$$

$$(f^{n-1}\nu' S' + f | \} f^n \nu S) \equiv C^{S'\ 1/2\ S}_{n,W'\ (100)\ W} \tag{134}$$

$$(W'U' + f | WU) \equiv C^{W'\ (100)\ W}_{U'\ (10)\ U} \tag{135}$$

$$(U'\alpha' L' + f | U \alpha L) \equiv C^{U'\ (10)\ U}_{\alpha' L'\ 3\ \alpha L} \tag{136}$$

For the coefficients $(f^{n-1}\nu' S' + f | \} f^n \nu S)$ a closed form is available (equations 52a-52b-56 of [30]). The other coefficients $(W'U' + f | WU)$ and $(U'\alpha' L' + f | U\alpha L)$, which represent the reduction coefficients of the Kronecker product of groups R_7 and G_2 respectively, can be computed via the diagonalization of the corresponding Casimir operators (G) (equivalent of a diagonalization of L^2 operator for the reduction of a representation of the group R_3). The Racah's expression of Casimir operators are:

$$G(R_7) = \frac{3}{5}(U^1)^2 + \frac{7}{5}(U^3)^2 + \frac{11}{5}(U^5)^2 \tag{137}$$

$$G(G_2) = \frac{3}{4}(U^1)^2 + \frac{11}{4}(U^5)^2 \tag{138}$$

where U^1, U^3 and U^5 are unit tensor operators, with known eigenvalues:

$$\lambda(W) = w_1(w_1+5)/2 + w_2(w_2+3)/2 + w_3(w_3+1)/2 \tag{139}$$

$$\lambda(U) = (u_1^2 + u_1 u_2 + u_2^2 + 5u_1 + 4u_2)/12 \tag{140}$$

The matrix elements of $G(G_2)$ between the states arising from the coupling of a single f electron to parent states of quantum numbers U', L', M'_L to give the final states of angular momentum L, M_L, can be expressed as:

$$\langle U'\alpha' L' + f \to L M_L | G(G_2) | U'\alpha'' L'' + f \to L M_L \rangle = \\ = (-1)^{L+3+L''} \frac{11}{2} \begin{Bmatrix} L & 3 & L' \\ 5 & L'' & 3 \end{Bmatrix} (U'\alpha' L' \| U^5_{n-1} \| U' L'' \alpha'') + \\ + \delta_{\alpha'\alpha''}\delta_{L'L''} \left[\lambda(U') + \lambda(10) + \frac{1}{112}(L(L+1) - L'(L'+1) - 3(3+1)) \right] \tag{141}$$

Once these matrix elements are evaluated for a chosen L and U', its diagonalization produces eigenvalues for U containing L and known to occur in the reduction $(U') \times (10)$. Each (nor-

malized) eigenvector is a whole set of $(U'\alpha'L' + f|U\alpha L)$ with L' varying along the row.

The RME of U^5 are computed iteratively, from the previous cycle of calculation. In this case the labels n, $n-1$, $n-2$ are used only to represent the hierarchy of computation, these coefficients are general and do not depend on the particular number of electrons considered. The necessary relations are the following:

$$\begin{aligned}(U'''L''' + f \to L'\|U^5_{n-1}\|U'''L'^v + f \to L'') = \\ = (2L'+1)^{1/2}(2L''+1)^{1/2}\left[(-1)^{L'''+L''}(U'''L'''\|U^5_{n-2}\|U'''L'^v)\right. \\ \left.\times \begin{Bmatrix} L''' & L' & 3 \\ L'' & L'^v & 5 \end{Bmatrix} + (-1)^{L'^v+L'}\delta_{L'''L'^v}\delta_{\alpha'''\alpha'^v} \begin{Bmatrix} 3 & L' & L''' \\ L'' & 3 & 5 \end{Bmatrix}\right] \end{aligned} \tag{142}$$

and

$$\begin{aligned}(U'\alpha'L'\|U^5_{n-1}\|U'\alpha''L'') = \\ = \sum_{\alpha''',L''',\alpha'^v,L'^v} (U'''\alpha'''L''' + f|U'\alpha'L')(U'^v\alpha'^vL'^v + f|U'\alpha'L') \\ \times (U'''L''' + f \to L'\|U^5_{n-1}\|U'''L'^v + f \to L'') \end{aligned} \tag{143}$$

An analogous procedure can be applied (subsequently) for the computation of $(W'U' + f|WU)$. The matrix elements of $G(R_7)$ are expressed as:

$$\begin{aligned}\langle W'U' + f \to ULM_L|G(R_7)|W'U'' + f \to ULM_L\rangle = \\ = \sum_{L',L''}\left[(U'L' + f|UL)(U''L'' + f|UL)(-1)^{L''+3+L}\right. \\ \left.\times\frac{14}{5}\begin{Bmatrix} L & 3 & L' \\ 3 & L'' & 3 \end{Bmatrix}(W'U'L'\|U^3_{n-1}\|W'U''L'')\right] + \\ +\delta_{U'U''}\left[\frac{4}{5}\lambda(U) + \lambda(W') - \frac{4}{5}\lambda(U') + \lambda(100) - \frac{4}{5}\lambda(10)\right] \end{aligned} \tag{144}$$

In this case, the W that have to be considered are those that occur in the reduction $W' \times (100)$. The iterative procedure for the computation of the RME for U^3 is similar to the one described for U^5, with equations:

$$\begin{aligned}(W'U'L'\|U^3_{n-2}\|W'U''L') = \\ = \sum_{U''',U'^v}(W'''U''' + f|W'U')(W'''U'^v + f|W'U'') \\ \times\left[\sum_{L''',L'^v}(U'''L''' + f|U'L')(U'^vL'^v + f|U''L'')\right. \\ \left.\times(W'''U'''L''' + f \to L'\|U^3_{n-1}\|W'''U'^vL'^v + f \to L'')\right] \end{aligned} \tag{145}$$

and:

$$
\begin{aligned}
&(W'''U''' + f \to L'''\|U^3_{n-1}\|W'''U'^v + f \to L'') = \\
&= (-1)^{L'''+3+L''+3}(2L'+1)^{1/2}(2L''+1)^{1/2} \\
&\times \left\{ \begin{array}{ccc} L''' & L' & 3 \\ L'' & L'^v & 3 \end{array} \right\} (W'''U'''L'''\|U^3_{n-2}\|W'''U'^vL'^v) + \\
&+(-1)^{L'''+3+L'+3}\delta_{L'''L'^v}\delta_{U'''U'^v}(2L'+1)^{1/2}(2L''+1)^{1/2} \left\{ \begin{array}{ccc} 3 & L' & L''' \\ L'' & 3 & 3 \end{array} \right\} \qquad (146)
\end{aligned}
$$

References

[1] Hans A Bethe. "Splitting of terms in crystals". In: *Selected Works Of Hans A Bethe: (With Commentary)*. World Scientific, 1997, pp. 1–72.

[2] Anna Bronova et al. *BonnMag: Computer program for ligand-field analysis of f n systems within the angular overlap model*. 2018.

[3] José J Baldoví et al. *SIMPRE: A software package to calculate crystal field parameters, energy levels, and magnetic properties on mononuclear lanthanoid complexes based on charge distributions*. 2013.

[4] José J Baldoví et al. "An updated version of the computational package SIMPRE that uses the standard conventions for S tevens crystal field parameters". In: *Journal of Computational Chemistry* 35.26 (2014), pp. 1930–1934.

[5] Mirosław Karbowiak and Czesław Rudowicz. "Software package SIMPRE—revisited". In: *Journal of Computational Chemistry* 35.26 (2014), pp. 1935–1941.

[6] Jan van Leusen et al. "Comprehensive insight into molecular magnetism via CONDON: Full vs. effective models". In: *Coordination Chemistry Reviews* 289 (2015), pp. 137–148.

[7] Manfred Speldrich, Jan van Leusen, and Paul Kögerler. *CONDON 3.0: An Updated Software Package for Magnetochemical Analysis-All the Way to Polynuclear Actinide Complexes*. 2018.

[8] Allen Scheie. "PyCrystalField: software for calculation, analysis and fitting of crystal electric field Hamiltonians". In: *Journal of Applied Crystallography* 54.1 (2021), pp. 356–362.

[9] Travis E Oliphant et al. *Guide to numpy*. Vol. 1. Trelgol Publishing USA, 2006.

[10] Sandro Tosi. *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[11] Pauli Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nature methods* 17.3 (2020), pp. 261–272.

[12] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. "Numba: A llvm-based python jit compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. 2015, pp. 1–6.

[13] Roman Boca. *Theoretical foundations of molecular magnetism*. Elsevier, 1999.

[14] CW Nielson and George F Koster. *Spectroscopic Coefficients for the p, d, and f Configurations*. MIT Press, 1963.

[15] Edgar König and Stefan Kremer. *Magnetism Diagrams for Transition Metal Ions*. Springer Science & Business Media, 2012.

[16] Roman Boca. *A handbook of magnetochemical formulae*. Elsevier, 2012.

[17] Ekkehard Konig. *Ligand field: energy diagrams*. Springer Science & Business Media, 2013.

[18] Christiane Görller-Walrand and Koen Binnemans. "Rationalization of crystal-field parametrization". In: *Handbook on the physics and chemistry of rare earths* 23 (1996), pp. 121–283.

[19] C-G Ma et al. "Systematic analysis of spectroscopic characteristics of the lanthanide and actinide ions with the 4fN and 5fN (N= 1… 14) electronic configurations in a free state". In: *Journal of alloys and compounds* 599 (2014), pp. 93–101.

[20] Frank Neese et al. “The ORCA quantum chemistry program package”. In: *The Journal of chemical physics* 152.22 (2020).

[21] Enrico Ravera et al. “A quantum chemistry view on two archetypical paramagnetic pentacoordinate nickel (II) complexes offers a fresh look on their NMR spectra”. In: *Inorganic Chemistry* 60.3 (2021), pp. 2068–2075.

[22] Mads Bak and Niels Chr Nielsen. “REPULSION, a novel approach to efficient powder averaging in solid-state NMR”. In: *Journal of Magnetic Resonance* 125.1 (1997), pp. 132–139.

[23] Matteo Briganti et al. “Magnetic anisotropy trends along a full 4f-series: the fn+ 7 effect”. In: *Journal of the American Chemical Society* 143.21 (2021), pp. 8108–8115.

[24] William H Press et al. *Numerical recipes*. Cambridge University Press, London, England, 1988.

[25] Edward Uhler Condon and George Hiram Shortley. *The theory of atomic spectra*. Cambridge University Press, 1935.

[26] Giulio Racah. “Group theory and spectroscopy”. In: *Springer Tracts in Modern Physics, Volume 37* (2006), pp. 28–84.

[27] Giulio Racah. “Theory of complex spectra. I”. In: *Physical Review* 61.3-4 (1942), p. 186.

[28] Giulio Racah. “Theory of complex spectra. II”. In: *Physical Review* 62.9-10 (1942), p. 438.

[29] Giulio Racah. “Theory of complex spectra. III”. In: *Physical Review* 63.9-10 (1943), p. 367.

[30] Giulio Racah. “Theory of complex spectra. IV”. In: *Physical Review* 76.9 (1949), p. 1352.

[31] Brian R Judd. *Operator techniques in atomic spectroscopy*. Vol. 35. Princeton University Press, 2014.

[32] PJ Redmond. “An explicit formula for the calculation of fractional parentage coefficients”. In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 222.1148 (1954), pp. 84–93.

[33] Clair Worley Nielson. “Fractional parentage coefficients for f-shell electrons”. PhD thesis. Massachusetts Institute of Technology, 1962.

[34] Giulio Racah. *Group theory and spectroscopy*. Institute for Advanced Study, 1951.

[35] ROBERT F Bacher and SAMUEL Goudsmit. “Atomic energy relations. I”. In: *Physical Review* 46.11 (1934), p. 948.

[36] John Stanley Griffith. *The theory of transition-metal ions*. Cambridge university press, 1961.

[37] Octave Duros et al. “General expressions for Stevens and Racah operator equivalents”. In: *arXiv preprint arXiv:2405.08978* (2024).

[38] Harvey A Buckmaster, Ramananda Chatterjee, and YH Shing. “The application of tensor operators in the analysis of EPR and ENDOR spectra”. In: *physica status solidi (a)* 13.1 (1972), pp. 9–50.

[39] Czeslaw Rudowicz. “Transformation relations for the conventional Okq and normalised O’kq Stevens operator equivalents with k= 1 to 6 and-k q k”. In: *Journal of Physics C: Solid State Physics* 18.7 (1985), p. 1415.

[40] KWH Stevens. “Matrix elements and operator equivalents connected with the magnetic properties of rare earth ions”. In: *Proceedings of the Physical Society. Section A* 65.3 (1952), p. 209.

[41] HA Buckmaster. “Tables of matrix elements for the operators”. In: *Canadian Journal of Physics* 40.11 (1962), pp. 1670–1677.

[42] C Rudowicz and CY Chung. “The generalization of the extended Stevens operators to higher ranks and spins, and a systematic review of the tables of the tensor operators and their matrix elements”. In: *Journal of Physics: Condensed Matter* 16.32 (2004), p. 5825.

[43] Sverker Edvardsson and Mattias Klintenberg. “Role of the electrostatic model in calculating rare-earth crystal-field parameters”. In: *Journal of alloys and compounds* 275 (1998), pp. 230–233.

[44] Brian G Wybourne and William F Meggers. *Spectroscopic properties of rare earths*. American Institute of Physics, 1965.

[45] Ivan D Ryabov. “On the operator equivalents and the crystal-field and spin Hamiltonian parameters”. In: *Applied Magnetic Resonance* 35 (2009), pp. 481–494.

[46] Malcolm Gerloch and Robert F McMeeking. “Paramagnetic properties of unsymmetrical transition-metal complexes”. In: *Journal of the Chemical Society, Dalton Transactions* 22 (1975), pp. 2443–2451.

[47] W Urland. “On the ligand-field potential for f electrons in the angular overlap model”. In: *Chemical Physics* 14.3 (1976), pp. 393–401.

[48] RM Lynden-Bell and AJ Stone. “Reorientational correlation functions, quaternions and Wigner rotation matrices”. In: *Molecular Simulation* 3.5-6 (1989), pp. 271–281.