openLOWDIN

Release 0.1

openLOWDIN

May 27, 2025

CONTENTS:

openLOWDIN is a computational program that implements the Any Particle Molecular Orbital (APMO) method to study systems containing any type and number of quantum species, such as electrons, positrons, quantum nuclei, muons, or drude oscillators.

This manual is still in early construction stage! Thanks for your patience.

CHAPTER

ONE

ABOUT

openLOWDIN is FORTRAN quantum chemistry code that implements the Any Particle Molecular Orbital (APMO) method to study systems containing any type and number of quantum species, such as electrons, positrons, quantum nuclei, muons, or drude oscillators. It's parallelized with OMP paradigm.

At present, openLOWDIN code is publicly available at https://github.com/efposadac/openLOWDIN

1.1 Capabilities

What can we do in Lowdin? The current version of the code encompasses the following quantum chemistry methods extended for any quantum species:

- HF
- DFT
- MP2
- CI (CIS, CISD, CIST, CISDTQ, FCI, CIPSI)
- PT (PT2, PT3, PP3, RENPP3, OVGF)
- NOCI

Check more details in the *Code* and *Tutorials*

1.2 Documentation

The online manual of openLOWDIN is available at https://github.com/openLOWDIN/openLOWDIN_manual. A compiled pdf version of the manual can be found here `openlowdin.pdf`

1.3 How to cite:

Please cite the code as:

R.i Flores-Moreno, E. Posada, F. Moncada, J. Romero, J. Charry, M. Díaz-Tinoco, S.A. González, N.F. Aguirre, A. Reyes, LOWDIN: The any particle molecular orbital code. Int. J. Quantum Chem.. 114. (1), 50–56 (2014).

1.4 Acknowledgements:

This code results from an evolution of the APMO and LOWDIN software packages, both mainly developed at the Universidad Nacional de Colombia. Currently, the code is under an open-source initiative thanks to the support of some initial developers, who are presently spread around the world!

CHAPTER

TWO

DEVELOPERS AND CONTRIBUTORS

2.1 Active developers (alphabetical order)

- Andrés Reyes
- Edwin Posada
- Jorge Charry
- Félix Moncada

2.2 Contributors (alphabetic order)

- Jhonathan Romero (APMO/Propagators)
- Sergio Gonzalez (APMO: Older versions)
- Nestor Aguirre (APMO: Older versions)
- Danilo González Forero (APMO/COSMO)
- Jose Mauricio Rodas Rodriguez (APMO/(QM/MM))
- Carlos Ortiz-Mahecha (APMO/CC)
- Alejandro Peña Torres (APMO/CC)
- Laura Pedraza-González (APMO/core)
- Manuel Diaz (APMO/Propagators)
- Teresa Tamayo-Mendoza (Development of higher order propagator methods)
- Roberto Flores-Moreno (ADFT/ADPT/Propagators)

CHAPTER

THREE

INSTALLATION

3.1 Dependencies

openLOWDIN requires the following standard packages:

```
wget git build-essential liblapack-dev libblas-dev libgsl0-dev autotools-dev automake␣
↪libtool gfortran python3 gawk libeigen3-dev libgmp-dev libboost-all-dev
```

Additionally, it requires some quantum chemistry libraries:

```
libinit - Molecular integrals in gaussian type orbital basis
libxc - DFT functionals
```

The following opensource libraries are distributed within the openLOWDIN code

```
aduw - Four-index integrals transformation
erkale - Orbital localization
gepol - COSMO
jadamilu - large sparse matrix diagonalization
molden2aim - molden to AIM wave function converter
```

3.2 Getting the code

The source code is available at https://github.com/efposadac/openLOWDIN

```
git clone https://github.com/efposadac/openLOWDIN
```

3.3 Basic installation

Once all the dependencies are installed, the code is compile with the following steps. First, run the interactive configuration script in openLOWDIN root directory. Be sure that you have permissions to write in the installation directory and have properly exported the *$PATH* environment.

```
./configure
```

This script will ask a set of questions, please provide the option that satisfies your needs.

```
INFO: Interactive configuration options
Fortran Compiler command? gfortran(default) or ifort/ifx [gfortran]

```

(continues on next page)

(continued from previous page)

```
Compiler Options: (1) regular,  (2) backtrace and debug,  (3) static (for intel fortran␣
↪compiler only), (4) Full debug, (5) Highest optimization level [1]

Speed up on GPUs? (you need to have already installed CUDA and Magma libraries): yes/no␣
↪[no]

Executable name? default=openlowdin [openlowdin]

Installation directory? default=/usr/local [/usr/local]
```

Compile the code with

```
make
```

you can add the parallel flag to compile in paralle, e.g. with 4 threads as *-j 4*.

Finally, install as

```
make install
```

To uninstall the binaries from the selected installation folder

```
make uninstall
```

To clean the project

```
make clean
make distclean
```

3.4 Step-by-step installation

Here you can find a step-by-step workflow to install on ubuntu-latest linux distribution.

```
### Step-by-step  installation example: (replace apt-get with your preferred package␣
↪manager) ###

        sudo apt-get update
        sudo apt-get -y install wget git build-essential liblapack-dev libblas-dev␣
↪libgsl0-dev autotools-dev automake libtool gfortran python3 gawk libeigen3-dev libgmp-
↪dev libboost-all-dev
        # Define ENV Variables
        export WORKDIR=$PWD/dependencies
        export PATH=$PATH:$WORKDIR/bin
        export C_INCLUDE_PATH=$C_INCLUDE_PATH:$WORKDIR/include:$WORKDIR/include/libint2:/
↪usr/include/eigen3
        export CPLUS_INCLUDE_PATH=$CPLUS_INCLUDE_PATH:$WORKDIR/include:$WORKDIR/include/
↪libint2:/usr/include/eigen3
        export LIBRARY_PATH=$LIBRARY_PATH:$WORKDIR/lib
        export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$WORKDIR/lib
        # Create work directories
        mkdir $WORKDIR
        mkdir $WORKDIR/bin
```

(continues on next page)

(continued from previous page)

```
      mkdir $WORKDIR/lib
      cd $WORKDIR

    # Libint2
      # If you have Ubuntu, you can get this precompiled Libint2 library
      wget https://www.dropbox.com/s/d3d44j238lkfwcr/libint-master-SEP052019.tgz
      tar xzvf libint-master-SEP052019.tgz

    # Otherwise, download and compile with minimal (default am), G12, fPIC options␣
↪(libint2 commit 668b10c4bdca5876984058742d4212675eb93f3f)
    # git clone https://github.com/evaleev/libint.git
    # cd libint
    # git checkout 668b10c4bdca5876984058742d4212675eb93f3f
    # ./autogen.sh
    # mkdir ../build
      # cd ../build
      # ../libint/configure --prefix=$WORKDIR --with-max-am=6 --enable-g12=4 --with-
↪g12-max-am=4 --with-cxxgen-optflags
      # make -j 4
      # make install
    # ../libint/configure --prefix=$WORKDIR

      cd -

      # Libint1
      git clone https://github.com/evaleev/libint.git
      cd libint
      git checkout v1
      aclocal -I lib/autoconf
      autoconf
      ./configure --prefix=$WORKDIR
      make -j 4
      make install
      make clean
      make distclean
      cd -

    # Libxc
      cd $WORKDIR
      # If you have Ubuntu, you can get this precompiled Libxc library
      wget https://www.dropbox.com/s/6cja3zzhl1cq46i/libxc-master-MAY242023.tgz
      tar xzvf libxc-master-MAY242023.tgz
    # Otherwise, download and compile with default options (libxc commit␣
↪4bd0e1e36347c6d0a4e378a2c8d891ae43f8c951)
    # git clone https://gitlab.com/libxc/libxc.git
    # cd libxc
    # git checkout 4bd0e1e36347c6d0a4e378a2c8d891ae43f8c951
    # autoreconf -i
    # ./configure --enable-shared --prefix=$WORKDIR
    # make -j 4
    # make install
```

(continues on next page)

(continued from previous page)

```
cd ..

# Configure Lowdin
./configure -p $WORKDIR/bin -s /tmp -l "-lblas -llapack"
# Build Lowdin
make -j 4
# Install Lowdin
make install
# Run Tests
make test
```

CHAPTER

FOUR

GETTING STARTED

Let's get ready to run openLOWDIN. Here you can find the basic information about the input and how to run the code. A more exhaustive description of all code keywords and files can be found in section *Code*

4.1 Input file

The code requires a plain text `input` file with extension `.lowdin`. Here is an example of a minimum `input` file for computing muonic water with propagator theory

```
GEOMETRY
e-[O]        6-31G                0.0000         0.0000         0.1173 multiplicity=2␣
↪addParticles=-1
e-[H]        6-31G                0.0000         0.7572        -0.4692
e-[H]        6-31G                0.0000        -0.7572        -0.4692
u-        13S.ET.O.u.TF         0.0000         0.0000         0.1173
O        dirac                 0.0000         0.0000         0.1173
H-a_1        DZSPDN                0.0000         0.7572        -0.4692 m = 1836.15
H-b_2        DZSPDn                0.0000        -0.7572         0.4692 m = 1836.15
END GEOMETRY

TASKS
        method = "UHF"
        propagatorTheoryCorrection = 2
END TASKS

CONTROL
        iterationScheme = 3
        convergenceMethod = 1
        readCoefficients = F
        hartreeProductGuess = T
END CONTROL
```

The minimum required blocks to run a calculation are `GEOMETRY`, `TASKS`, and `CONTROL`.

The `GEOMETRY` block provides the information needed to build the molecular system. The first column declares the type of the quantum species. As shown in the above example, `e-[H]` and `e-[O]` define the electrons of a Hydrogen and a Oxygen atom respectively; `U-` defines a negative muon, `O dirac`, `H\_1` and `H\_2` define a ^{16}O, ^{1}H and ^{2}H nuclei respectively.

The second column declares the basis sets. When the `dirac` basis is chosen, the particle is treated as a classical point charge. The third, fourth and fifth columns declare the x,y,z coordinates of the particle basis set center.

The sixth column provides additional information via keywords `addParticles` and `multiplicity`. These keywords are used to change the default values. `addParticles` is used to modify the number of particles of a quantum species. As shown in the provided example, one electron is removed from the system. `multiplicity` defines the multiplicity for open shell calculations. In the example, an electronic multiplicity of 2 was chosen.

4.2 How to run

To run openLOWDIN with 4 OMP threads simply run

```
openlowdin -i inputname.lowdin -n 4
```

This will generate a plain text output file called `inputname.out`. These are the full command line options:

```
$ openlowdin -i file.lowdin [-t [all] [list] [file]] [-n number] [-v number] [-p] [-s] [-
↪h]

-i file.lowdin
   This is the input file name
-n number
   This will set the number of OMP threads
-t all
   This will run all the test files located on the test database.
-t list
   This will list all the test files located on the test database.
-t file
   This will run a specific test file which is located on test database.
-v number
   This is the lowdin version that will be used
-p
   This will print the output file to the standard output on the fly
-w
   This will save the LOWDIN .wfn file
-k
   This will keep the temporary files in the scratch directory after running the␣
↪calculation
-s
   This activate the singleton mode
-h
   This will print this same message
```

4.3 Scratch

openLOWDIN creates a folder to save temporary files located at `$LOWDIN_SCRATCH/$nameFile` where `$nameFile` is the input name given after the command line `-i file.lowdin` without the `.lowdin` extension. The `$LOWDIN_SCRATCH` environmental variable is set in the bash script `.openlowdin/lowdinvars.sh` which is located in the installation folder during configuration.

> **Warning**
>
> Please notice that the scratch folder is completely removed after the calculation terminates.

CHAPTER

FIVE

CODE

This sections summarizes the multicomponent version of the common quantum chemistry methods and capabilities implemented in openLOWDIN, as well as of all available input code keywords.

CHAPTER

SIX

INPUT

6.1 General structure

openLOWDIN input file consists on a plain text file with extension `.lowdin` which will be internally processed by a bash script to generate a Fortran namelist, creating the true input file for the code. The input file consists of different blocks enclosed by the lines `BLOCK` and `END BLOCK`, where "BLOCK" is one of the following sections:

- `GEOMETRY`: define the position of the basis set centers, position of point charges, number and properties of quantum species.
- `TASKS` : select the type of calculation to be performed by the code.
- `CONTROL`: contains all general input parameters to control the behavior of the program, such as thresholds, maximum number of SCF cycles. See *CONTROL* for a full of keywords.
- `OUTPUT`: request the calculation of other molecular property to visualize the molecular wave function or density. See *Outputs*
- `BASIS` : declare a user defined basis. This can be alternatively defined in an external file. See *Basis*
- `INTERPOTENTIAL` define an internal potential between pairs of quantum species. See *Potentials*
- `EXTERPOTENTIAL` define an external potential for a quantum species. See *Potentials*
- `POTENTIAL` declare a user defined potential built on gaussian type orbitals. See *Potential Basis*
- `INPUT_CI` define the active space for each quantum species for CI calculations. See *Configuration Interaction, CI*

6.2 GEOMETRY block

The `GEOMETRY` block provides the information needed to build the molecular system as:

```
GEOMETRY
SPECIES_SYMBOL    BASIS_NAME  X_COORD Y_COORD Z_COORD
END GEOMETRY.
```

each line define either a new particle (quantum or point charge), or a new basis center (quantum particles only).

- `SPECIES_SYMBOL` *[character]* symbol of the quantum particle as defined in the `lib/dataBases/elementalParticles.lib` and `lib/dataBases/constantsOfCoupling.lib` files, or the symbol of point charges defined in `lib/dataBases/atomicElements.lib`. See *DataBases* to define or redefine particles.
- `BASIS_NAME` *[character]* for quantum particles, this corresponds to the name of the basis set, the code will look in the `BASIS` block or in the folder `lib/basis`. For point charge simply use `dirac`
- `X_COORD`, `Y_COORD`, `Z_COORD` *[real]* coordinates of the basis set center origin or position of the point charge.

Besides the above mandatory elements, the following flags can be added to modify the properties of the particles declared within the same line.

- `addParticles`= *[integer]* modify the number of particles of a quantum species. Positive values increase number of particles, negative values decrease the number of particles. To not be confused with total charge. *Default* `0`
- `multiplicity`= *[integer]* declare the multiplicity of the quantum species as $2S + 1$, where S is the total spin within the quantum species. *Default* `1`
- `q`= *[real]* charge of the particle, quantum or point charge. *Default* defined in `lib/dataBases/elementalParticles.lib`
- `m`= *[real]* mass of the particle, quantum only. *Default* defined in `lib/dataBases/elementalParticles.lib`
- `omega`= *[real]* frequency of the harmonic potential of the particle, quantum only. *Default* `0.0` See *Potentials*
- `qdoCenterOf`= *[character]* declare this particle as the point charge center of the quantum species in *[character]* within the QDO formalism. *Default* `"NONE"` See *Quantum Drude Oscillators, QDOs*

Note that a duplicate of one of the above keywords will overwrite the previous values for particles defined with the same name.

6.3 TASKS block

This block control the method to be computed

- `method`= *[character]* Type of Hartree-Fock method See *Hartree-Fock, HF*

`"RHF"`	Restricted Hartree-Fock
`"UHF"`	Unrestricted Hartree-Fock
`"NONE"`	No Hartree-Fock. *Default*

- `optimizeGeometry`= *[logical]* Activates geometry optimization. *Default* `False`
- `mollerPlessetCorrection`= *[integer]* Order of Möller-Plesset Perturbation correction. Values: `2`. *Default* `0` See *Many-Body Perturbation Theory, MBPT*
- `epsteinNesbetCorrection`= *[integer]* Order of Epstein-Nesbet correction. Values: `2`, `3`. *Default* `0` See *Many-Body Perturbation Theory, MBPT*
- `propagatorTheoryCorrection`= *[integer]* Order of propagator theory correction. Values: `2`, `3`. *Default* `0` See *Propagathor Theory, PT*
- `nonOrthogonalConfigurationInteraction`= *[logical]* Performs non-Orthogonal Configuration Interaction calculation. *Default* `False` See *Non-Orthogonal Configuration Interaction, NOCI*
- `configurationInteractionLevel`= *[character]* Select Configuration Interaction level. See *Configuration Interaction, CI*

`"CIS"`	Singles
`"CISD"`	Singles and doubles
`"CISD-"`	Singles, doubles interspecies only
`"CISD+"`	Singles, doubles, triples interpecies only
`"CISDT"`	Singles, doubles, triples
`"CISDTQ"`	Singles, doubles, triples, quadruples
`"SCI"`	Selected CI
`"FCI"`	Full CI
`"NONE"`	*Default*

- TDHF= *[logical] Default* `False`
- cosmo= *[logical]* Performs an implicit solvent job using COSMO model. *Default* `False`
- subsystemEmbedding= *[logical] Default* `False`

CHAPTER

SEVEN

LIB

openLOWDIN employs external files to obtain information regarding particles properties, basis sets, and potential basis. Those files are located in the folder `openLOWDIN/lib` under the following tree structure: .. code

```
├── basis
├── dataBases
├── libjadamilu.a
├── libmolden2aim.a
├── libtransform.a
├── lowdincore.a
└── potentials
```

Notice that all *.a files corresponds to static external libraries compiled during openLOWDIN installation.

7.1 DataBases

Information regarding particle properties are stored in three differents files, while `uffParameters.lib` contains the Universal Force Field (UFF) parameters

```
├── atomicElements.lib
├── constantsOfCoupling.lib
├── elementalParticles.lib
└── uffParameters.lib
```

7.1.1 atomicElements.lib

This file stores some physicochemical for all elements in the periodic table in a Fortran namelist format. This information is mostly used to get the charge of classical particles and number of electrons from the given atomic element, as well as the mass of quantum nuclei

```
&ELEMENT
      NAME = "HYDROGEN"
      SYMBOL = "H"
      ATOMICNUMBER = 1
      MASS = 1.00794
      MELTINGPOINT = 13.81
      BOILINGPOINT = 20.28
      DENSITY = 0.084
      ELECTRONAFFINITY = -73
      IONIZATIONENERGY1 = 1312
      ELECTRONEGATIVITY = 2.1
```

(continues on next page)

(continued from previous page)

```
      COVALENT = 0.30
      ATOMIC = 0.25
  KLAMT = 0.00
      VANDERWAALS = 1.2
      ISOTOPES = 1, 1.0078250321, 99.9885, 0.5,
               2, 2.0141017780, 0.0115, 1,
               3, 3.0160492675, 0.0, 0.5,
               4, 4.02783, , -2.0,
               5, 5.03954, , ,
               6, 6.04494, , ,
               7, 3.1289311806, , 1,
/
```

7.1.2 constantsOfCoupling.lib

`lambda`, `eta`, and `particlesFraction` are related to the occupation of quantum species (number of particles) per orbital, this variable is used in post-HF methods and nuclear gradients. `kappa` is used to change the sign of the exchange integrals. See *Hartree-Fock, HF*

```
&SPECIE
      NAME = "ELECTRON"
      SYMBOL = "E-"
      KAPPA = -1.0
      ETA = 2.0
      LAMBDA = 2.0
      PARTICLESFRACTION = 0.5
/
```

7.1.3 elementalParticles.lib

this namelist defines the elemental properties of quantum particles: charge, mass, and spin

```
&PARTICLE
      NAME = "ELECTRON"
      SYMBOL = "E-"
      CATEGORY = "LEPTON"
      CHARGE = -1
      MASS = 1.0
      SPIN = 0.5
/
```

Note

To create a new unique (distinguishable) quantum species is necessary to add a new entry into both namelist on constantsOfCoupling.lib and elementalParticles.lib which an unique SYMBOL not used by another other quantum species

Note

openLOWDIN has three exceptions for quantum species. Electrons, defined by `"E-"`, in which the number of particles and basis set are defined by adding the atomic symbol in parenthesis, e.g. `E-(N)` will declare to add 7 electrons to the quantum species `"E-"` and to add a set of Gaussian type functions identifed by the name `"NITROGEN"` from the basis set requested. The quantum species `E-ALPHA` and `E-BETA` are automatically created when requesting and `UHF` calculation.

Note

The mass of quantum nuclei can be selected by adding to the species symbol the isotope number followed by "_" e.g. `H_3` will define a Tritium nucleus.

7.2 Basis

openLOWDIN constructs the spatial part of spin-molecular orbitals, χ as a linear combination of Gaussian type functions (GTFs), φ_μ^α:

$$\chi_i^\alpha(\mathbf{r}_i) = \sum_\mu^{N_{bas}^\alpha} C_\mu^\alpha \varphi_\mu^\alpha(\mathbf{r}_i; \mathbf{R}_\mu) \tag{7.1}$$

where C_μ^α is a combination coefficient for species α, and the atomic orbital $\varphi_\mu^\alpha(\mathbf{r}_i; \mathbf{R}_\mu)$ is built as a sum of primitive functions forming a contracted orbital

$$\varphi_\mu^\alpha(\mathbf{r}_i; \mathbf{R}_\mu) = \sum_s b_{s\mu}^\alpha N_{s\mu}^\alpha (x_i - X_\mu)^{l_\mu^\alpha} (y_i - Y_\mu)^{m_\mu^\alpha} (z_i - Z_\mu)^{n_\mu^\alpha} \times \exp[-a_s^\alpha (\mathbf{r}_i - \mathbf{R}_\mu)^2] \tag{7.2}$$

here, $b_{s\mu}^\alpha$ are the primitive coefficients, $N_{s\mu}^\alpha$ is a normalization constant, $\{x_i, y_i, z_i\}$ are the spatial coordinates of the position vector $\mathbf{r}$ for the i particle of the quantum species α, the set $\{l, m.n\}$ are the angular momemtum components, a_s is the exponent, and $\mathbf{R}$ is the GTF center, which is independent of nuclei positions.

openLOWDIN has a collection of built-in basis stored in `lib/basis/`. Please note that all filenames should be written in capital letters. The code employs the same basis set format of deMon2k, That is to say

```
O-ELEMENT_NAME ELEMENT_SYMBOL_OR_SPECIES_SYMBOL (BASIS_NAME) BASIS TYPE: 1
#
NUMBER_OF_CONTRACTED_ORBITALS
ID  ANGULAR_MOMEMTUM  NUMBER_OF_PRIMITIVES
EXPONENT  COEFFICIENT
```

For example, for a Hydrogen atom center with aug-cc-pVDZ basis

```
O-HYDROGEN H (AUG-CC-PVDZ) BASIS TYPE: 1
#
5
1 0 3
13.01000000 0.01968500
1.96200000 0.13797700
0.44460000 0.47814800
2 0 1
0.12200000 1.00000000
3 0 1
0.02974000 1.00000000
4 1 1
```

(continues on next page)

(continued from previous page)

```
0.72700000 1.00000000
5 1 1
0.14100000 1.00000000
```

Alternatively, it's possible to define the basis set of a quantum species within the `.lowdin` input file by creating a `BASIS BASIS_NAME - END BASIS` block section, where `BASIS_NAME` should be a basis name defined within the `GEOMETRY` block. For example

```
GEOMETRY
        e-[H]  cc-pvtz     0.0000  0.0000  0.00000
        e-[H]  CUSTOM_1    0.0000  0.0000  0.74144
        H_1    CUSTOM_2    0.0000  0.0000  0.00000 m=1836.1527
        U-     CUSTOM_3    0.0000  0.0000  0.74144 m=206.7683
        He_4   CUSTOM_3    0.0000  0.0000  0.74144 m=7349.6727
END GEOMETRY

BASIS CUSTOM_1
O-HYDROGEN H (CC-PVTZ+LOCAL) BASIS TYPE: 1
#
9
1 0 1
103.8700000 1.00000000
2 0 1
33.8700000 1.00000000
3 0 1
5.09500000 1.00000000
4 0 1
1.15900000 1.00000000
5 0 1
0.32580000 1.00000000
6 0 1
0.10270000 1.00000000
7 1 1
1.40700000 1.00000000
8 1 1
0.38800000 1.00000000
9 2 1
1.05700000 1.00000000
END BASIS
```

If the basis set is not provided within the input file, then the code will look in the `lib/basis` folder for a basis set file under the name defined in the `GEOMETRY` block, e.g. a file `lib/basis/CUSTOM_1` in the above example.

A more extensive collection of updated basis set can be found https://www.basissetexchange.org/ or https://github.com/MolSSI-BSE/basis_set_exchange

7.3 Potential Basis

openLOWDIN has the capabilities of computing additional one-body external potential, as well as replacing the standard two-particles Coulomb potential by a general two-body intraspecies and interspecies potentials based on sum Gaussian-Type functions (GTFs).

7.3.1 External potential

This potential is built as a sum of uncontracted and unnormalized GTFs

$$V_1^{\alpha}(\mathbf{r}_i) = \sum_{\tau}^{N_{bas}^{\alpha}} C_{\tau}^{\alpha}(x_i - X_{\tau})^{l_{\tau}^{\alpha}}(y_i - Y_{\tau})^{m_{\tau}^{\alpha}}(z_i - Z_{\tau})^{n_{\tau}^{\alpha}} \times \exp[-a_{\tau}^{\alpha}(\mathbf{r}_i - \mathbf{R}_{\tau})^2] \tag{7.3}$$

where all parameters are defined in a similar fashion that the one defined for basis sets. The format of this potential basis corresponds to

```
O-SPECIES_SYMBOL
#
NUMBER_OF_FUNCTIONS
ID  ANGULAR_MOMEMTUM
EXPONENT  COEFFICIENT
ORIGIN_X ORIGIN_Y ORIGIN_Z
```

For example, for a 4*s* potential felt by the species `HEA3`

```
O-HEA3
#
25
1 0
0.00100000 -4.85267703e-04
0.0 0.0 0.0
2 0
0.00200000 2.44420303e-03
0.0 0.0 0.0
3 0
0.00400000 -7.54493346e-03
0.0 0.0 0.0
4 0
0.00800000 1.57739046e-02
0.0 0.0 0.0
```

In the input file, this potential is invoked by adding a new line in `EXTERPOTENTIAL` - `END EXTERPOTENTIAL` block section as

```
EXTERPOTENTIAL
       SPECIES_SYMBOL  POT_NAME
END EXTERPOTENTIAL
```

where `POT_NAME` is the name of the potential file described above for the given species `SPECIES_SYMBOL`. For example

```
EXTERPOTENTIAL
       HEA3   HE2C60-IH-1P
       HEB3   HE2C60-IH-1P
END EXTERPOTENTIAL
```

> **Note**
>
> Notice that it's possible to define a potential for different quantum species within the same file.

7.3.2 Internal potential

This potential is also built as a sum of uncontracted and unnormalized geminal GTFs, but it is limited to s-type orbitals

$$V_2^{\alpha,\beta}(\mathbf{r}_i^\alpha, \mathbf{r}_j^\beta) = \sum_{\tau}^{N_{bas}^{\alpha\beta}} C_\tau^{\alpha\beta} \exp[-a_\tau^{\alpha\beta}(\mathbf{r}_i^\alpha - \mathbf{r}_j^\beta)^2] \tag{7.4}$$

And the potential format is given by

```
O-SPECIES_SYMBOL_ALPHASPECIES_SYMBOL_BETA
#
NUMBER_OF_FUNCTIONS
ID  ANGULAR_MOMEMTUM
EXPONENT  COEFFICIENT
ORIGIN_X ORIGIN_Y ORIGIN_Z
```

For a 4*s* potential between two quantum species `HEA3` and `HEB3`

```
O-HEA3HEB3
4
0 0
     10.000000000000      16.533492358000
0.0 0.0 0.0
1 0
      7.498942093300      -6.924850494800
0.0 0.0 0.0
2 0
      5.623413251900       5.780268985500
0.0 0.0 0.0
3 0
      4.216965034300       1.957403910100
0.0 0.0 0.0
4 0
      3.162277660200      -5.166130478700
```

In the input file, this potential is invoked by adding a new line in `INTERPOTENTIAL - END INTERPOTENTIAL` block section as

```
INTERPOTENTIAL
       SPECIES_SYMBOL_ALPHASPECIES_SYMBOL_BETA  POT_NAME
END INTERPOTENTIAL
```

where `POT_NAME` is the name of the potential file described above between the species `SPECIES_SYMBOL_ALPHA` and `SPECIES_SYMBOL_BETA`. For example

```
INTERPOTENTIAL
        HEA3 HEA3       HE2C60-IH-2P
        HEA3 HEB3       HE2C60-IH-2P
        HEB3 HEB3       HE2C60-IH-2P
END INTERPOTENTIAL
```

> **Note**
>
> Notice that it's possible to define a potential for different quantum species within the same file.

Note

Notice that interspecies potentials can be simply declared by setting $\alpha = \beta$, in other words, by setting the same quantum species symbol twice.

Warning

For simplicity, openLOWDIN requires to read an angular momentum and origin for the two-body potentials, despite these are not used in the potential definition.

CHAPTER

EIGHT

CONTROL

8.1 Dummy variables

For debugging purposes

- `dummyReal(:)`= *[float]* Dummy real array(10). *Default* `0.0_8`
- `dummyInteger(:)`= *[integer]* Dummy integer array(10). *Default* `0`
- `dummyLogical(:)`= *[logical]* Dummy logical array(10). *Default* `.false.`
- `dummyCharacter(:)`= *[character]* Dummy character array(10). *Default* `""`

8.2 Geometry optimization

Currenly unsupported

- `numericalDerivativeDelta`= *[float] Default* `1.0E-3`
- `minimizationInitialStepSize`= *[float] Default* `0.5_8`
- `minimizationLineTolerance`= *[float] Default* `0.001_8`
- `minimizationToleranceGradient`= *[float] Default* `0.00001_8`
- `minimizationMaxIteration`= *[integer] Default* `200`
- `minimizationMethod`= *[integer] Default* `4`
- `minimizationLibrary`= *[character] Default* `"GENERIC"`
- `coordinates`= *[character] Default* `"CARTESIAN"`
- `energyCalculator`= *[character] Default* `"INTERNAL"`
- `analyticGradient`= *[logical] Default* `.true.`
- `minimizationWithSinglePoint`= *[logical] Default* `.true.`
- `useSymmetryInMatrices`= *[logical] Default* `.false.`
- `restartOptimization`= *[logical] Default* `.false.`
- `firstStep`= *[logical] Default* `.true.`
- `lastStep`= *[logical] Default* `.true.`
- `optimizeWithCpCorrection`= *[logical] Default* `.false.`
- `cpCorrection`= *[logical] Default* `.false.`
- `TDHF`= *[logical] Default* `.false.`

- optimize= *[logical] Default* .false.
- optimizeGeometryWithMP= *[logical] Default* .false.
- projectHessiane= *[logical] Default* .true.

8.3 Atomic connectivity

Currenly unsupported

- bondDistanceFactor= *[float] Default* 1.3_8
- bondAngleThreshold= *[float] Default* 170.0_8
- dihedralAngleThreshold= *[float] Default* 170.0_8

8.4 COSMO

Currenly unsupported

- cosmo= *[logical] Default* .false.
- cosmo_solvent_dielectric= *[character] Default* 78.3553d+00
- cosmo_scaling= *[character] Default* 0.0d+00

8.5 Info and units

- formatNumberOfColumns= *[integer] Default* 5
- unitForOutputFile= *[integer] Default* 6
- unitForMolecularOrbitalsFile= *[integer] Default* 8
- unitForMP2IntegralsFile= *[integer] Default* 7
- printLevel= *[integer]*

0	No output
1	Normal. *Default*
5	Method
6	Method and wave function
7	Method, wave function, global
8	Method, wave function, global, SCF

- units= *[character] Default* "ANGS"
- doubleZeroThreshold= *[float] Default* 1.0E-12

8.6 General

- method= *[character] Default* "NONE"
- transformToCenterOfMass= *[logical] Default* .false.
- areThereDummyAtoms= *[logical] Default* .false.
- areThereQDOPotentials= *[logical] Default* .false.

- setQDOEnergyZero= *[logical] Default* .false.
- isThereExternalPotential= *[logical] Default* .false.
- isThereInterparticlePotential= *[logical] Default* .false.
- isThereOutput= *[logical] Default* .false.
- isThereFrozenParticle= *[logical] Default* .false.
- dimensionality= *[integer] Default* 3

8.7 Molecular Mechanics

Currenly unsupported

- forceField= *[character] Default* "UFF"
- electrostaticMM= *[logical] Default* .false.
- chargesMM= *[logical] Default* .false.
- printMM= *[logical] Default* .false.

8.8 Miscelaneous options

- MOFractionOccupation= *[float] Default* 1.0_8
- ionizeMO= *[integer] Default* 0
- ionizeSpecies= *[character] Default* "NONE"
- exciteSpecies= *[character] Default* "NONE"

8.9 Integrals transformation

- integralsTransformationMethod= *[character] Default* "C"
- ITBuffersize= *[integer] Default* 1024

8.10 Libraries

- uffParametersDataBase= *[character] Default* "/dataBases/uffParameters.lib"
- atomicElementsDataBase= *[character] Default* "/dataBases/atomicElements.lib"
- basisSetDataBase= *[character] Default* "/basis/"
- potentialsDataBase= *[character] Default* "/potentials/"
- elementalParticlesDataBase= *[character] Default* "/dataBases/elementalParticles.lib"

CHAPTER

NINE

OUTPUTS

Besides the standard output, openLowdin can generate other type of outputs to view the results of an APMO calculation. Lowdin input has an "OUTPUTS" block to request these outputs. Currently, it can generate:

- Molden files for each species.
- AIM files. These files are generated with the molden2AIM program
- Gaussian cubes for orbitals and density
- Gnuplot 2D and 3D graphs for density and orbitals
- Gaussian fchk files

9.1 Molden and AIM files

To generate molden or AIM files simply add in the OUTPUTS block:

- moldenFile
- wfnFile
- wfxFile
- NBO47File

openLowdin will generate an .molden, .wfn, .wfx, or .47 file for each quantum species in the input.

For molden, there are three format types that can be selected with the CONTROL option

Table 1: moldenFileFormat =

QUANTUM	Define the coordinates, GTO and MO for each quantum species individually.
STANDARD	Same that QUANTUM but including the coordinates of classical particles. (Default)
MIXED	Same that STANDARD but including 1s GTO for each classical particles with zero contribution in the MO.

All three formats work with the MOLDEN software. Other visualization codes may require the MIXED or QUANTUM formats.

> If CI or NOCI calculations with "CIStatesToPrint" greater that zero were selected, the molden files will use the CI or NOCI natural orbitals. Also, adding "state=N" in the moldenFile line allow us to select the natural orbitals of the Nth excited state.

See *Molden examples* for full input examples to generate molden files

9.2 Gaussian Cubes

openLowdin generates Gaussian density or orbital cubes of a chosen species. These cubes can be read by many visualization programs, such as VMD. To generate cubes, add in the OUTPUTS block the lines

- orbitalCube
- densityCube

openLowdin will generate a .cub file for each cube requested.

Add in each line "species=symbol", where symbol is the quantum species to be plotted. To generate a cube for each species, use "species=ALL". For orbital plots, select an orbital with "orbital=N". The default is the HOMO of each species.

The position of the center of the cube is declared with "center=X Y Z" or with "point1=X Y Z". With "cubeSize=N" we declare the length of one side of the cube. The number of points is controled either by defining the number of points per side with "pointsPerDim=N", or the separation between points in one dimension, with "scanStep=N"

9.3 Gnuplot 2D and 3D graphs

openLowdin generates plots of the density or orbitals of a chosen species using Gnuplot. To do this, add in the OUTPUTS block the lines

- densityPlot
- orbitalPlot

As with the orbital cubes "species=symbol" selects the desired quantum species to be plotted. species=ALL is supported. For orbital plots, choose an orbital with "orbital=N", the default will be the HOMO of each species.

To create 2D plots, add dimensions=2, select and axis and provide TWO endpoints.

- axis="A" limitA=p1 p2, replace A with x, y, z

When using the axis directive, "offsetX=", "offsetY=", "offsetZ=" may be used to shift the plot.

A more general definition may be provided with

- point1=X1 Y1 Z1 point2=X2 Y2 Z2, where X,Y,Z are coordinates of each endpoint

For example, to plot the electronic density from -2.0 to 2.0 along the Z axis add one of the following lines

- densityPlot species="e-" dimensions=2 axis="z" limitZ=-2.0 2.0
- densityPlot species="e-" dimensions=2 point1= 0.0 0.0 -2.0 point2=0.0 0.0 2.0

To create 3D plots, set dimensions=3 and provide a plane with two endpoints for each axis

- plane="AB" limitA=p1 p2 limitB=p3 p4, replace A and B with x, y, z

  an offset along the unused axis may be selected to shift the plot.

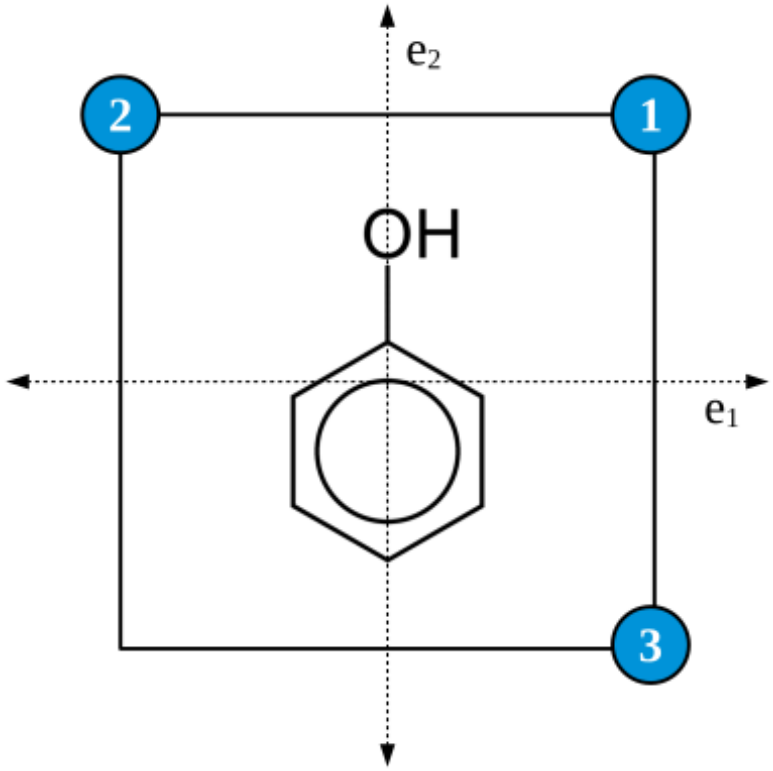
A more general definition is also possible, where the user provides THREE points corresponding to the corners of a rectangle.

- point1=X1 Y1 Z1 point2=X2 Y2 Z2 point2=X3 Y3 Z3, where X,Y,Z are coordinates of each corner

For example, to plot electron density in the YZ plane, from -2.5 to 2.5 in both axis, use either of these lines

- densityPlot species="e-" dimensions=3 plane=yz limitY=-2.5 2.5 limitZ=-2.5 2.5
- densityPlot species="e-" dimensions=3 point1=0.0 2.5 2.5 point2=0.0 -2.5 2.5 point3=0.0 2.5 -2.5

In the general approach, the order of the corners is important. point1 must be the central coorner. See the following figure for an example



In both 2D and 3D plots, the number of points is controled either by defining the number of points per side with "pointsPerDim=N", or the separation between points in one dimension, with "scanStep=N"

openLowdin will generate three files for Gnuplot: A .dens or a .orb with the raw data of the plot; a .gnp with the gnuplot script to generate the plot; and a .eps with the plot itself.

9.4 Fchk files

Gaussian fchk files may be used to pass the results to other programs. Currenty, openLowdin uses them to exchange information with Erkale for orbital localization. Add the line in the OUTPUTS

- fchkFile

to generate a file for each quantum species in the input. See *Molden examples* for an input example that employs orbital localization.

CHAPTER

TEN

INTEGRALS

openLOWDIN computes analitically one-body and two-body integrals based on atomic orbitals expressed as a sum of contracted GTFs under a cartesian coordinate representation (see *Basis*)

$$\chi_i^\alpha(\mathbf{r}_i) = \sum_\mu^{N_{bas}^\alpha} C_\mu^\alpha \varphi_\mu^\alpha(\mathbf{r}_i; \mathbf{R}_\mu) \tag{10.1}$$

In matrix form, the following integrals are implemented in openLOWDIN

10.1 One-body integrals

All computed under an Obara-Saika scheme for any angular momentum in the basis set

Overlap

$$S_{\mu\nu}^\alpha = \int dr_i \varphi_\mu^\alpha(i) \varphi_\nu^\alpha(i) \tag{10.2}$$

Kinetic

$$T_{\mu\nu}^\alpha = \int dr_i \varphi_\mu^\alpha(i) \nabla_i \varphi_\nu^\alpha(i) \tag{10.3}$$

Point charge potential

$$\mathbb{PC}_{\mu\nu}^\alpha = \sum_I \int dr_i \varphi_\mu^\alpha(i) \frac{1}{r_i - R_I} \varphi_\nu^\alpha(i) \tag{10.4}$$

Moment integrals

$$\mathbb{UA}_{\mu\nu}^\alpha = \int dr_i \varphi_\mu^\alpha(i) r_{i,\mathbb{A}} \varphi_\nu^\alpha(i) \quad \mathbb{A} = x, y, z \tag{10.5}$$

$$\mathbb{UAB}_{\mu\nu}^\alpha = \int dr_i \varphi_\mu^\alpha(i) r_{i,\mathbb{A}} r_{i,\mathbb{B}} \varphi_\nu^\alpha(i) \quad \mathbb{A}, \mathbb{B} = x, y, z \tag{10.6}$$

Harmonic integrals

$$\mathbb{HA}_{\mu\nu}^\alpha = \int dr_i \varphi_\mu^\alpha(i) r_i^2 \varphi_\nu^\alpha(i) \tag{10.7}$$

Three-center integrals

$$\mathbb{FC}_{\mu\nu}^\alpha = \sum_\sigma \int dr_i \varphi_\mu^\alpha(i) \varphi_\sigma(i) \varphi_\nu^\alpha(i) \tag{10.8}$$

10.2 Two-body integrals

These integrals are computed with LIBINT library https://github.com/evaleev/libint

Four-center intraspecies coulomb potential

$$\langle \mu^\alpha \nu^\alpha | \sigma^\alpha \lambda^\alpha \rangle = \int \int dr_i dr_j \varphi_\mu^\alpha(i) \varphi_\nu^\alpha(j) \frac{1}{r_i - r_j} \varphi_\sigma^\alpha(i) \varphi_\lambda^\alpha(j) \tag{10.9}$$

Four-center interspecies coulomb potential

$$\langle \mu^\alpha \nu^\beta | \sigma^\alpha \lambda^\beta \rangle = \int \int dr_i dr_j \varphi_\mu^\alpha(i) \varphi_\nu^\beta(j) \frac{1}{r_i - r_j} \varphi_\sigma^\alpha(i) \varphi_\lambda^\beta(j) \tag{10.10}$$

Five-center, intra- and interspecies

$$\langle \mu^\alpha \nu^\beta | V_2^{\alpha,\beta}(\mathbf{r}_i^\alpha, \mathbf{r}_j^\beta) | \sigma^\alpha \lambda^\beta \rangle = \sum_\tau C_\tau^{\alpha\beta} \int \int dr_i dr_j \varphi_\mu^\alpha(i) \varphi_\nu^\beta(j) \exp[-a_\tau^{\alpha\beta}(r_i - r_j)^2] \varphi_\sigma^\alpha(i) \varphi_\lambda^\beta(j) \tag{10.11}$$

10.2.1 Input options

- `tv`= *[float]* deprecated *Default* `1.0E-6`
- `integralThreshold`= *[float]* threshold to store integrals in disk above the given value. *Default* `1.0E-10`
- `integralStackSize`= *[integer]* write and load integrals temporary files by stacks of this values. *Default* `30000`
- `integralStorage`= *[character]* select storage scheme for two-particles integrals

`"DISK"`	Storage all non-zero integrals in disk at $SCRATCH folder, after four-index permutational symmetries. *Default*
`"MEMORY"`	Allocate a four dimensional array in RAM memory
`"DIRECT"`	Compute integrals on-the-fly (only for the SCF step)

- `integralScheme`= *[character]* select two-particles library. *Default* `"LIBINT"`
- `schwarzInequality`= *[logical]* perfoms Schwarz inequality to skip blocks of small integrals. Deprecated, now it's used by default within libint interface. *Default* `.false.`

CHAPTER

ELEVEN

HARTREE-FOCK, HF

- frozen= *[character] Default* "NONE"
- freezeNonElectronicOrbitals= *[logical] Default* .false.
- freezeElectronicOrbitals= *[logical] Default* .false.
- hartreeProductGuess= *[logical] Default* .false.
- readCoefficients= *[logical] Default* .true.
- readFchk= *[logical] Default* .false.
- writeCoefficientsInBinary= *[logical] Default* .true.
- readEigenvalues= *[logical] Default* .false.
- readEigenvaluesInBinary= *[logical] Default* .true.
- writeEigenvaluesInBinary= *[logical] Default* .true.
- noSCF= *[logical] Default* .false.
- finiteMassCorrection= *[logical] Default* .false.
- removeTranslationalContamination= *[logical] Default* .false.
- buildTwoParticlesMatrixForOneParticle= *[logical] Default* .false.
- buildMixedDensityMatrix= *[logical] Default* .false.
- onlyElectronicEffect= *[logical] Default* .false.
- electronicWaveFunctionAnalysis= *[logical] Default* .false.
- isOpenShell= *[logical] Default* .false.
- getGradients= *[logical] Default* .false.
- HFprintEigenvalues= *[logical] Default* .true.
- HFprintEigenvectors= *[character] Default* "OCCUPIED"
- overlapEigenThreshold= *[float] Default* 1.0E-8_8
- electricField(:)= *[float] Default* 0.0_8
- multipoleOrder= *[integer] Default* 0

CHAPTER

TWELVE

SCF

- nonElectronicEnergyTolerance= *[float] Default* 1.0E-8
- electronicEnergyTolerance= *[float] Default* 1.0E-8
- nonelectronicDensityMatrixTolerance= *[float] Default* 1.0E-6
- electronicDensityMatrixTolerance= *[float] Default* 1.0E-6
- totalEnergyTolerance= *[float] Default* 1.0E-8
- totalDensityMatrixTolerance= *[float] Default* 1.0E-6
- densityFactorThreshold= *[float] Default* 1.0E-8
- diisSwitchThreshold= *[float] Default* 0.5
- diisSwitchThreshold_bkp= *[float] Default* 0.5
- electronicLevelShifting= *[integer] Default* 0.0
- nonelectronicLevelShifting= *[integer] Default* 0.0
- exchangeOrbitalThreshold= *[float] Default* 0.8
- waveFunctionScale= *[integer] Default* 1000.0
- scfNonelectronicMaxIterations= *[integer] Default* 50
- scfElectronicMaxIterations= *[integer] Default* 50
- scfGlobalMaxIterations= *[integer] Default* 200
- listSize= *[integer] Default* -20
- convergenceMethod= *[character] Default* 1!!(0)NONE,(1)DAMPING,(2)DIIS,(3)LEVELSHIFTING(4)DAMPING/DIIS
- diisDimensionality= *[integer] Default* 10
- iterationScheme= *[character] Default* 3!!(0)NONELECRONICFULLY/e-(1)ELECTRONICFULLY(2)CONVERGEDINDIVIDIALL
- scfElectronicTypeGuess= *[character] Default* "HCORE"
- scfNonelectronicTypeGuess= *[character] Default* "HCORE"
- scfConvergenceCriterium= *[character] Default* "ENERGY"!ENERGY,DENSITY,BOTH
- diisErrorInDamping= *[logical] Default* .false.
- activateLevelShifting= *[logical] Default* .false.
- exchangeOrbitalsInSCF= *[logical] Default* .false.
- forceClosedShell= *[logical] Default* .false.

- debugScfs= *[logical] Default* .false.
- scfGhostSpecies= *[character] Default* "NONE"

CHAPTER

THIRTEEN

POTENTIALS

13.1 Inter-potentials

13.2 External-potentials

CHAPTER

FOURTEEN

PROPERTIES

CHAPTER

FIFTEEN

DENSITY FUNCTIONAL THEORY, DFT

- gridStorage= *[character] Default* "DISK"
- electronCorrelationFunctional= *[character] Default* "NONE"
- electronExchangeFunctional= *[character] Default* "NONE"
- electronExchangeCorrelationFunctional= *[character] Default* "NONE"
- nuclearElectronCorrelationFunctional= *[character] Default* "NONE"
- positronElectronCorrelationFunctional= *[character] Default* "NONE"
- betaFunction= *[character] Default* "NONE"
- gridRadialPoints= *[integer] Default* 35
- gridAngularPoints= *[integer] Default* 110
- gridNumberOfShells= *[integer] Default* 5
- finalGridRadialPoints= *[integer] Default* 50
- finalGridAngularPoints= *[integer] Default* 302
- finalGridNumberOfShells= *[integer] Default* 5
- polarizationOrder= *[integer] Default* 1
- numberOfBlocksInAuxiliaryFunctions= *[integer] Default* 3
- fukuiFunctions= *[logical] Default* .false.
- auxiliaryDensity= *[logical] Default* .false.
- storeThreeCenterElectronIntegrals= *[logical] Default* .true.
- callLibxc= *[logical] Default* .true.
- nuclearElectronDensityThreshold= *[float] Default* 1E-10
- electronDensityThreshold= *[float] Default* 1E-10
- gridWeightThreshold= *[float] Default* 1E-10
- betaParameterA= *[integer] Default* 0.0
- betaParameterB= *[integer] Default* 0.0
- betaParameterC= *[integer] Default* 0.0

15.1 Subsystem embedding Options

- `subsystemEmbedding`= *[logical] Default* `.false.`
- `localizeOrbitals`= *[logical] Default* `.false.`
- `subsystemLevelShifting`= *[integer] Default* `1.0E6`
- `subsystemOrbitalThreshold`= *[float] Default* `0.1`
- `subsystemBasisThreshold`= *[float] Default* `0.0001`
- `erkaleLocalizationMethod`= *[character] Default* `"MU"`

CHAPTER

SIXTEEN

MANY-BODY PERTURBATION THEORY, MBPT

OpenLowdin can perform second order many-body perturbation theory (MP2) corrections to the energy on systems including any kind of quantum particle. These corrections are probably the simplest way to recover the effects of correlation between particles of different species.

$$E_{\text{MP2}} = \text{E}_{\text{HF}} + \sum_{\alpha}^{N_{typ}} \text{E}^{(2)}_{\alpha\alpha} + \sum_{\alpha\beta}^{N_{typ}} \text{E}^{(2)}_{\alpha\beta}, \tag{16.1}$$

$$E^{(2)}_{\alpha\alpha} = \frac{q_\alpha^2}{4} \sum_{ij}^{oc_\alpha} \sum_{ab}^{vir_\alpha} \frac{|\ \langle i_\alpha j_\alpha \mid\mid a_\alpha b_\alpha \rangle\ |^2}{\varepsilon_{\alpha i} + \varepsilon_{\alpha j} - \varepsilon_{\alpha a} - \varepsilon_{\alpha b}},$$

$$E^{(2)}_{\alpha\beta} = q_\alpha q_\beta \sum_{i}^{oc_\alpha} \sum_{j}^{oc_\beta} \sum_{a}^{vir_\alpha} \sum_{b}^{vir_\beta} \frac{|\ \langle i_\alpha j_\beta \mid a_\alpha b_\beta \rangle\ |^2}{\varepsilon_{\alpha i} + \varepsilon_{\beta j} - \varepsilon_{\alpha a} - \varepsilon_{\beta b}}.$$

For more information on APMO-MP2 calculations see S. A. Gonz'alez, A. Reyes, "Nuclear Quantum Effects on the He2H+ Complex With the Nuclear Molecular Orbital Approach", Int. J. Quant. Chem. 110 689 (2010) https://doi.org/10.1002/qua.22118

In addition, OpenLowdin can compute second order Epsein-Nesbet (EN2) corrections, which correspond to renormalized MP2 equations.

See *MP2 with quantum nucleus* for an example of a MP2 calculation in openLowdin

- `mpCorrection`= *[integer] Default* `1`
- `mpFrozenCoreBoundary`= *[integer] Default* `0`
- `mpOnlyElectronicCorrection`= *[logical] Default* `.false.`
- `epsteinNesbetCorrection`= *[integer] Default* `1`

CHAPTER

SEVENTEEN

PROPAGATHOR THEORY, PT

OpenLOWDIN can calculate ionization potentials for any species employing the propagator formalism, where ionization energy for an specific orbital is calculated as the Koopmans value plus self-energy corrections. The current implementation includes second order corrections (APMO/P2), second order plus transition operator corrections (APMO/TOEP2), third order corrections (APMO/P3) and renormalized third order corrections (APMO/REN-P3) as in the multicomponent extension of the outer valence Green function method (APMO/OVGF)

At second order, there are intraspecies and interspecies contributions to the self energy corrections to the eigenvalue of reference orbital p:

$$\begin{aligned}\omega_{\alpha p} =& \epsilon_{\alpha p} + \Sigma_{\alpha pp}(\omega_{\alpha p}) \\ \Sigma^{(2)}_{\alpha pp}(\omega_{\alpha p}) =& \sum_{i}^{oc_\alpha}\sum_{ab}^{vir_\alpha} \frac{|\langle p_\alpha i_\alpha || a_\alpha b_\alpha\rangle|^2}{\omega_{\alpha p} + \epsilon_{\alpha i} - \epsilon_{\alpha a} - \epsilon_{\alpha b}} + \sum_{ij}^{oc_\alpha}\sum_{a}^{vir_\alpha} \frac{|\langle p_\alpha a_\alpha || i_\alpha j_\alpha\rangle|^2}{\omega_{\alpha p} + \epsilon_{\alpha a} - \epsilon_{\alpha i} - \epsilon_{\alpha j}} \\ &+ \sum_{a}^{vir_\alpha}\sum_{i}^{oc_\beta}\sum_{b}^{vir_\beta} \frac{|\langle p_\alpha i_\beta | a_\alpha b_\beta\rangle|^2}{\omega_{\alpha p} + \epsilon_{\beta i} - \epsilon_{\alpha a} - \epsilon_{\beta b}} + \sum_{i}^{oc_\alpha}\sum_{j}^{oc_\beta}\sum_{a}^{vir_\beta} \frac{|\langle p_\alpha a_\beta | i_\alpha j_\beta\rangle|^2}{\omega_{\alpha p} + \epsilon_{\beta a} - \epsilon_{\alpha i} - \epsilon_{\beta j}}\end{aligned} \tag{17.1}$$

More details on the multicomponent propagator methods can be found in (romero.JCP.137.074105.2012,romero.JCP.141.114103.2014)

To perform a propagator calculations using LOWDIN, the order of PT (2 or 3) must be specified in the "TASKS" block using the keyword "propagatorTheoryCorrection". Currently, the third order corrections is only available from a UHF reference.

A default calculation obtains ionization energies for the HOMO and LUMO orbitals of all the species present in the input. Using the "IonizeMO" and "ionizeSpecies" in the "CONTROL" block, allows the user to select specific orbitals for the propagator calculation. In that case, set up the flag "ptJustOneOrbital=.T." to save computational time.

Transition operator corrections latter take advantage of fractional occupation to include additional relaxation in calculated ionization energies. To perform calculations with this method, select the UHF reference, add to the control block the "ptTransitionOperator=.T." and select a fractional occupation using "MOfractionOccupation". The recommended value is 0.5.

For third order calculations, you can select in the "CONTROL" block the type of correction to be used by adding "ptP3Method=" with "P3","EP3","OVGF-A","OVGF-B","OVGF-C" and "REN-P3" as options. By default, the six correction types are computed.

See *PT2 with quantum nucleus*, *TOEP2 with quantum nucleus* and *PT3 for a Ps-atom complex* for examples of propagator calculations in openLowdin

- `ptOnlyOneSpecieCorrection`= *[logical] Default* `.false.`
- `selfEnergyScan`= *[logical] Default* `.false.`
- `ptTransitionOperator`= *[logical] Default* `.false.`
- `ptJustOneOrbital`= *[logical] Default* `.false.`

- selfEnergySpacing= *[float] Default* 0.5_8
- selfEnergyRange= *[float] Default* 5.0_8
- ptOrder= *[integer] Default* 1
- ptMaxIterations= *[integer] Default* 50
- ptIterationMethod2Limit= *[integer] Default* 1
- ptIterationScheme= *[integer] Default* 1
- ptMaxNumberOfPolesSearched= *[integer] Default* 10
- ptFactorSS= *[integer] Default* 0
- ptFactorOS= *[integer] Default* 0
- ptP3Method= *[character] Default* "NONE"
- ptP3Method(1)= *[character] Default* "ALL"

CHAPTER

EIGHTEEN

CONFIGURATION INTERACTION, CI

The APMO/CI wave function is written as a linear combination of CI configurations between all quantum species

$$|\Phi_0\rangle = c_0|\Psi_0\rangle + \sum_{\alpha}\sum_{ia\in\alpha} c_i^a|\Psi_i^a\rangle + \sum_{\alpha,\beta}\sum_{\substack{ia\in\alpha\\ jb\in\beta}} c_{ij}^{ab}|\Psi_{ij}^{ab}\rangle + \sum_{\alpha,\beta}\sum_{\substack{ia\in\alpha\\ jb\in\alpha\\ kc\in\beta}} c_{ijk}^{abc}|\Psi_{ijk}^{abc}\rangle + \cdots \tag{18.1}$$

- `configurationInteractionLevel`= *[character] Default* `"NONE"`
- `numberOfCIStates`= *[integer] Default* `1`
- `CIdiagonalizationMethod`= *[character] Default* `"DSYEVR"`
- `CIdiagonalDressedShift`= *[character] Default* `"NONE"`
- `CIactiveSpace`= *[character] Default* `0!!Full`
- `CIstatesToPrint`= *[integer] Default* `1`
- `CImaxNCV`= *[integer] Default* `30`
- `CIsizeOfGuessMatrix`= *[integer] Default* `300`
- `CIstackSize`= *[integer] Default* `5000`
- `CIConvergence`= *[float] Default* `1E-4`
- `CImatvecTolerance`= *[float] Default* `1E-10`
- `CISaveEigenVector`= *[logical] Default* `.false.`
- `CILoadEigenVector`= *[logical] Default* `.false.`
- `CIJacobi`= *[logical] Default* `.false.`
- `CIBuildFullMatrix`= *[logical] Default* `.false.`
- `CIMadSpace`= *[integer] Default* `5`
- `CINaturalOrbitals`= *[logical] Default* `.false.`
- `CIPrintEigenVectorsFormat`= *[character] Default* `"OCCUPIED"`
- `CIPrintThreshold`= *[float] Default* `1E-1`

CHAPTER

NINETEEN

NON-ORTHOGONAL CONFIGURATION INTERACTION, NOCI

- nonOrthogonalConfigurationInteraction= *[logical] Default* .false.
- translationScanGrid(:)= *[integer] Default* 0
- rotationalScanGrid= *[integer] Default* 0
- rotationAroundZMaxAngle= *[integer] Default* 360
- rotationAroundZStep= *[integer] Default* 0
- nestedRotationalGrids= *[integer] Default* 1
- translationStep= *[integer] Default* 0.0
- nestedGridsDisplacement= *[integer] Default* 0.0
- configurationEnergyThreshold= *[integer] Default* 1.0
- configurationOverlapThreshold= *[float] Default* 1.0E-8
- configurationMaxDisplacement(:)= *[integer] Default* 0.0
- configurationMinDisplacement(:)= *[integer] Default* 0.0
- configurationMaxNPDistance= *[integer] Default* 1.0E8
- configurationMinPPDistance= *[integer] Default* 0.0
- configurationMaxPPDistance= *[integer] Default* 1.0E8
- configurationEquivalenceDistance= *[float] Default* 1.0E-8
- empiricalOverlapParameterA= *[float] Default* 0.0604
- empiricalOverlapParameterB= *[float] Default* 0.492
- empiricalOverlapParameterE0= *[integer] Default* 0.0
- empiricalOverlapParameterSc= *[integer] Default* 0.0
- configurationUseSymmetry= *[logical] Default* .false.
- readNOCIgeometries= *[logical] Default* .false.
- empiricalOverlapCorrection= *[logical] Default* .false.
- onlyFirstNOCIelements= *[logical] Default* .false.
- computeROCIformula= *[logical] Default* .false.

CHAPTER

TWENTY

TUTORIALS

In this section you can find multiples examples for openLOWDIN calculations

- *Positronic systems*
- *Positron covalent bond*
- *Quantum Nuclei*
- *MP2 with quantum nucleus*
- *Negative Muons*
- *Quantum Drude Oscillators, QDOs*
- *External Potentials*
- *Molden examples*

CHAPTER

TWENTYONE

POSITRONIC SYSTEMS

This is an example on how to run a *Configuration Interaction, CI* calculations for PsH

```
GEOMETRY
        e-(H)   SHARON-E-6S2P         0.00        0.00         0.00 addParticles=1
        H        dirac                        0.00        0.00         0.00
        e+        SHARON-E+6S2P        0.00        0.00        0.00
END GEOMETRY

TASKS
        method = "UHF"
        configurationInteractionLevel ="FCI"
        !configurationInteractionLevel ="CISD"
END TASKS

CONTROL
readCoefficients=F
numberOfCIstates=3
CINaturalOrbitals=T
        CIStatesToPrint = 1
        !CIdiagonalizationMethod = "DSYEVX"
        CIdiagonalizationMethod = "JADAMILU"
        !CIPrintEigenVectorsFormat = "NONE"
        CIPrintEigenVectorsFormat = "OCCUPIED"
        !CIPrintEigenVectorsFormat = "ORBITALS"
        CIPrintThreshold = 5e-2
        buildTwoParticlesMatrixForOneParticle=T
END CONTROL

INPUT_CI
        species="E-ALPHA" core=0 active=0
        species="E-BETA" core=0 active=0
        species="E+" core=0 active=0
END INPUT_CI
```

CHAPTER

TWENTYTWO

POSITRON COVALENT BOND

This is an example on how to compute the binding energy of a dihydride positron-bound system, using *Configuration Interaction, CI* calculations, as was done in https://doi.org/10.1002/anie.201800914

$$[\mathrm{H}^{-}\mathrm{e}^{+}\mathrm{H}^{-}] \longrightarrow \mathrm{PsH} + \mathrm{H}^{-}$$

This input computes the energy of the dihydride system

```
!The goal of this calculation is to compute the binding energy of a positron bound␣
↪complex
!Reported:
!E(e+H2^2-): -1.279372 a.u.

SYSTEM_DESCRIPTION='e+H2^2- from Charry 2018 (10.1002/anie.201800914)'

!add two electrons (one for each hydrogen anion)
!remove one positron
GEOMETRY
        e-(H)   AUG-CC-PVDZ         0.00        0.00         -1.6 addParticles=1
        e-(H)   AUG-CC-PVDZ         0.00        0.00         1.6 addParticles=1
        e+        E+-H-AUG-CC-PVDZ        0.00        0.00        -1.6
        e+        E+-H-AUG-CC-PVDZ        0.00        0.00        1.6 addParticles=-1
        H        dirac                0.00        0.00         -1.6
        H        dirac                0.00        0.00         1.6
END GEOMETRY

!method to solve the SCF - CI only works for unrestricted reference
!CI level strings chooses the desired excitations to be included. FCI is all possible␣
↪excitations

TASKS
        method = "UHF"
        configurationInteractionLevel ="CISDTQ"
        !configurationInteractionLevel ="CIS","CISD","CISDT","CISDTQ","FCI"
END TASKS

!Compute only the "numberOfCIstates" states. Here we select the ground and the first␣
↪excited state
!Compute the density matrix for "CIstatesToPrint" states, for density outputs
!Generate the natural orbitals, for visualization in molden files
!The Davidson diagonalization implemented in JADAMILU is the recomended method.
!For small systems, full matrix diagonalization with DSYEVX is possible
```

(continues on next page)

(continued from previous page)

```
!CI EigenVectors with coefficient higher than "CIPrintThreshold" are printed
!Printing format "OCCUPIED" shows the coefficients, "ORBITALS" shows the strings, "NONE"␣
↪skips printing
!Strict SCF convergence improves the quality of the CI results (not required for the FCI)

CONTROL
        numberOfCIstates=2
        CIStatesToPrint=2
        CINaturalOrbitals=T
        CIdiagonalizationMethod = "JADAMILU"
        !CIdiagonalizationMethod = "DSYEVX"
        CIPrintEigenVectorsFormat = "OCCUPIED"
        !CIPrintEigenVectorsFormat = "NONE","ORBITALS"
        CIPrintThreshold = 5e-2
        totalEnergyTolerance=1E-12
END CONTROL

!INPUT_CI block help us define the frozen core and active virtuals orbitals. Here we are␣
↪not restricting the excitation space
INPUT_CI
        species="E-ALPHA" core=0 active=0
        species="E-BETA" core=0 active=0
        species="E+" core=0 active=0
END INPUT_CI

!With CI, moldenFiles, 1D and 2D density slices and density cubes are good ways to␣
↪visualize the density results
OUTPUTS
        moldenFile state=1
        moldenFile state=2
        densityPlot dimensions=2 point1=0.0 0.0 -6.0 point2=0.0 0.0 6.0 state=1␣
↪scanStep=0.001
        densityPlot dimensions=2 point1=0.0 0.0 -6.0 point2=0.0 0.0 6.0 state=2␣
↪scanStep=0.001
        densityPlot dimensions=3 point1=0.0 -3.0 -4.0 point2=0.0 -3.0 4.0 point3=0.0 3.0␣
↪-4.0 state=1 scanStep=0.01
        densityPlot dimensions=3 point1=0.0 -3.0 -4.0 point2=0.0 -3.0 4.0 point3=0.0 3.0␣
↪-4.0 state=2 scanStep=0.01
END OUTPUTS
```

Then, we have to subtract the energy obtained from calculations of the dissociated species

```
!The goal of this calculation is to compute the binding energy of a positron bound␣
↪complex
!Reported:
!E(PsH): -0.734559

SYSTEM_DESCRIPTION='PsH from Charry 2018 (10.1002/anie.201800914)'
```

(continues on next page)

(continued from previous page)

```
GEOMETRY
        e-(H)   AUG-CC-PVDZ         0.00        0.00         0.00 addParticles=1
        e+        E+-H-AUG-CC-PVDZ        0.00        0.00        0.00
        H        dirac                0.00        0.00         0.00
END GEOMETRY

TASKS
        method = "UHF"
        configurationInteractionLevel ="FCI"
END TASKS

CONTROL
        numberOfCIstates=1
        CIStatesToPrint=1
        CINaturalOrbitals=T
        CIdiagonalizationMethod = "JADAMILU"
        CIPrintEigenVectorsFormat = "OCCUPIED"
        CIPrintThreshold = 5e-2
END CONTROL

OUTPUTS
        moldenFile state=1
        densityPlot dimensions=2 point1=0.0 0.0 -6.0 point2=0.0 0.0 6.0 state=1
END OUTPUTS
```

```
!The goal of this calculation is to compute the binding energy of a positron bound␣
↪complex
!Reported:
!E(H-): -0.524029

SYSTEM_DESCRIPTION='H- from Charry 2018 (10.1002/anie.201800914)'

GEOMETRY
        e-(H)   AUG-CC-PVDZ         0.00        0.00         0.00 addParticles=1
        H        dirac                0.00        0.00         0.00
END GEOMETRY

TASKS
        method = "UHF"
        configurationInteractionLevel ="FCI"
END TASKS

CONTROL
        numberOfCIstates=1
        CIStatesToPrint=1
        CINaturalOrbitals=T
        CIdiagonalizationMethod = "JADAMILU"
```

(continues on next page)

(continued from previous page)

```
        CIPrintEigenVectorsFormat = "OCCUPIED"
        CIPrintThreshold = 5e-2
END CONTROL

OUTPUTS
        moldenFile state=1
        densityPlot dimensions=2 point1=0.0 0.0 -6.0 point2=0.0 0.0 6.0 state=1
END OUTPUTS
```

CHAPTER

TWENTYTHREE

QUANTUM NUCLEI

CHAPTER

TWENTYFOUR

MP2 WITH QUANTUM NUCLEUS

A MP2 (second order *Many-Body Perturbation Theory, MBPT*) input on the hydrogen fluoride molecule, where the electrons and hydrogen nucleus are treated as quantum particles should be like this

```
GEOMETRY
        e-(F)   cc-pvtz   0.00 0.00 0.91
        e-(H)   cc-pvtz   0.00 0.00 0.00
        F       dirac     0.00 0.00 0.91
        H_1     dzspnb    0.00 0.00 0.00
END GEOMETRY
TASKS
        method="RHF"
        mollerPlessetCorrection=2
END TASKS
CONTROL
        mpFrozenCoreBoundary=1
END CONTROL
```

Here, APMO-MP2 calculations are performed when the option "mollerPlessetCorrection=2" is present in the "TASKS" block. MP2 calculations may use RHF or UHF as reference.

The CONTROL option mpFrozenCoreBoundary: Omits this number of occupied electronic molecular orbitals in the MP2 calculations (core electrons). Default 0.

For other species, the number of core orbitals along with the number of active virtuals, can be controlled with the "INPUT_CI" block in the input.

A MP2 output will include the summary of the MP2 results

```
 POST HARTREE-FOCK CALCULATION
 MANY-BODY PERTURBATION THEORY:
============================

             MOLLER-PLESSET FORMALISM
              ORDER OF CORRECTION =     2

               E(0) + E(1) =    -100.017935705706
                      E(2) =      -0.283903659181
                             ____________________
                   E(MP2) =    -100.301839364887

             ---------------------------------
             E(n){ Species }     E(n) / Hartree
```

(continues on next page)

(continued from previous page)

```
 ----------------------------------

    E(2){ E- } =      -0.273701262404
   E(2){ H_1 } =       0.000000000000

E(2){ E-/H_1 } =      -0.010202396777
```

Where the E(2) is the second order correction to the energy, and the E(MP2) is the Hartree-Fock energy plus E(2). This summary also includes the intraspecies and interspecies contributions to E(2).

CHAPTER

TWENTYFIVE

PT2 WITH QUANTUM NUCLEUS

This is a minimal input for a second order propagator theory (*Propagathor Theory, PT*) calculation to obtain the electronic ionization energies and proton binding energies for a water molecule. The PT2 corrections are computed when "propagatorTheoryCorrection=2" is added to the input:

```
 GEOMETRY
 e-(O)        6-31G            0.0000  0.0000  0.1173
 e-(H)        6-31G            0.0000  0.7572 -0.4692
 e-(H)        6-31G       0.0000 -0.7572 -0.4692
 O               dirac            0.0000  0.0000  0.1173
 H-a_1        Nakai-7-SPD 0.0000  0.7572 -0.4692
 H-b_1        Nakai-7-SPD 0.0000 -0.7572 -0.4692
 END GEOMETRY

 TASKS
        method = "RHF"
        propagatorTheoryCorrection=2
 END TASKS
```

The PT2 output will include a summary of the Koopmans' (KT) and self-energy corrected (EP2) results for the highest occupied and lowest unoccupied orbital of each species,

```
 POST HARTREE-FOCK CALCULATION
 PROPAGATOR THEORY:
==============================

       PROPAGATOR FORMALISM FOR SEVERAL FERMIONS SPECIES
        ORDER OF CORRECTION =     2
          The following articles must be cited:
          -------------------------------------
             J. Chem. Phys. 137, 074105 (2012)
             J. Chem. Phys. 138, 194108 (2013)
                             CPL coming soon
          -------------------------------------
       SPECIES:        E-
       -------------------------------------------------
           Orbital     KT (eV)     EP2 (eV)       P.S
       -------------------------------------------------
              5      -13.335244  -10.140654    0.853889
              6        5.362715    4.925340    0.952097
       -------------------------------------------------
       SPECIES:     H-A_1
```

(continues on next page)

(continued from previous page)

```
-------------------------------------------------
   Orbital     KT (eV)     EP2 (eV)       P.S
-------------------------------------------------
      1      -24.603421  -17.007032    0.877287
      2      -23.948321  -57.734065    0.538455
-------------------------------------------------
SPECIES:     H-B_1
-------------------------------------------------
   Orbital     KT (eV)     EP2 (eV)       P.S
-------------------------------------------------
      1      -24.603421  -17.007032    0.877287
      2      -23.948321  -57.734065    0.538455
-------------------------------------------------
```

For the occupied orbitals, KT and EP2 results are estimates of the energy requiered to remove a particle from the corresponding orbital (ionization potential). For the unoccupied ones, KT and EP2 results are estimates of the energy gained by adding a particle to the orbital (electron affinity). Here, the Pole Strength (P.S) serves as a quantity that validates the diagonal (pseudoparticle) approximation employed. A P.S value below 0.85 usually indicates that the diagonal approximation is not reliable.

CHAPTER

TWENTYSIX

TOEP2 WITH QUANTUM NUCLEUS

Here, we compute the second order propagator theory (*Propagathor Theory, PT*) corrections to a partially ionized water molecule electronic orbital (HOMO). The TOEP2 method requires a UHF reference.

First, run a regular UHF calculation to generate the molecular orbitals, let's name it "H2O.UHF.lowdin". These molecular orbitals will be used as the guess for the partially ionized SCF.

```
GEOMETRY
        e-(O)        6-31G            0.0000  0.0000  0.1173 multiplicity=1
        e-(H)        6-31G            0.0000  0.7572 -0.4692
        e-(H)        6-31G       0.0000 -0.7572 -0.4692
        O        dirac             0.0000  0.0000  0.1173
        H-a_1        Nakai-7-SPD 0.0000  0.7572 -0.4692
        H-b_1        Nakai-7-SPD 0.0000 -0.7572 -0.4692
END GEOMETRY

TASKS
        method = "UHF"
END TASKS
```

Now, we generate the TOEP2 input, let's name it "H2O.TOEP2.lowdin". In addition to the PT2 flag, "propagatorTheoryCorrection=2", we add to the input the flag "ptTransitionOperator=T" along with "IonizeMO" and "ionizeSpecies" to select the orbital and the species, and "MOfractionOccupation" to select the occupation.

```
GEOMETRY
        e-(O)        6-31G            0.0000  0.0000  0.1173 multiplicity=1
        e-(H)        6-31G            0.0000  0.7572 -0.4692
        e-(H)        6-31G       0.0000 -0.7572 -0.4692
        O        dirac             0.0000  0.0000  0.1173
        H-a_1        Nakai-7-SPD 0.0000  0.7572 -0.4692
        H-b_1        Nakai-7-SPD 0.0000 -0.7572 -0.4692
END GEOMETRY

TASKS
        method = "UHF"
        propagatorTheoryCorrection=2
END TASKS

CONTROL
        readCoefficients=T         !read molecular orbitals generated with H2O.UHF.lowdin
        ionizeSpecies="E-ALPHA"
```

(continues on next page)

(continued from previous page)

```
        ionizeMO=5
        MOfractionOccupation=0.5
        ptTransitionOperator=T
END CONTROL
```

Before running this calculation, rename the molecular orbitals file "H2O.UHF.vec" to "H2O.TOEP2.vec"

The output will include the summary of the P2 corrections to the partially ionized orbital

```
         PROPAGATOR FORMALISM FOR SEVERAL FERMIONS SPECIES
          ORDER OF CORRECTION =     2  + TRANSITION OPERATOR
            The following articles must be cited:
            -------------------------------------
               J. Chem. Phys. 127, 134106 (2007)
               J. Chem. Phys. 137, 074105 (2012)
               J. Chem. Phys. 138, 194108 (2013)
                                CPL coming soon
            -------------------------------------
         SPECIES:   E-ALPHA
         ------------------------------------------------------------------------------------
↪----------
             Orbital     KT (eV)    EP2 (eV)        P.S   SCS-EP2(eV)       P.S   SOS-
↪EP2(eV)       P.S
         ------------------------------------------------------------------------------------
↪----------
                5      -10.364472  -10.768693    0.327967  -10.740363    0.357031  -10.
↪725240    0.369623
         ------------------------------------------------------------------------------------
↪----------
```

This summary includes the Koopmans' (KT) and self-energy corrected (EP2) binding energy for the selected orbital. In addition, PT2 calculations with UHF reference include opposite-spin-scaled (SOS) and spin-component-scaled (SCS) results.

For the transition-operator results, the Pole Strength (P.S) value does not indicates the quality of the approximation.

CHAPTER

TWENTYSEVEN

PT3 FOR A PS-ATOM COMPLEX

A PT3 (third order *Propagathor Theory, PT*) input on positronium chloride (PsCl), where the electrons and the positron are treated as quantum particles looks like this

```
GEOMETRY
e-(Cl) aug-cc-pvdz 0.0 0.0 0.0 addParticles=1 multiplicity=1
Cl     Dirac       0.0 0.0 0.0
E+     PsX-DZ      0.0 0.0 0.0
END GEOMETRY

TASKS
        method = "uhf"
        propagatorTheoryCorrection = 3
END TASKS

CONTROL
        ionizeSpecies = "e+"
        ionizeMO= 1
        ptJustOneOrbital=.T.
END CONTROL
```

APMO-PT3 calculations are performed when the option "propagatorTheoryCorrection=3" is present in the "TASKS" block. PT3 calculations only work with UHF as reference. Here, to save computational time, only the self-energy corrections to the positron occupied orbital will be performed. This is indicated by the "ionizeSpecies", "ionizeMO" and "ptJustOneOrbital" entries in the CONTROL block.

The corresponing PT3 output will include the following summary

```
-----------------------------------------------------------------
       SUMMARY OF PROPAGATOR RESULTS FOR THE SPIN-ORBITAL: 1  OF SPECIES:E+
    ---------------------------------------------
             Method      BE (eV)     Pole S.
    ---------------------------------------------
          KT           -3.918673
          EP2          -4.579523    0.975236
          P3           -4.904336    0.954164
          EP3          -4.830791    0.951763
          OVGF-A       -4.915085    0.944111
          OVGF-B       -4.945115    0.946181
          OVGF-C       -4.900723    0.945407
          REN-P3       -4.946967    0.952206
    ---------------------------------------------
```

In this summary, KT is the eigenvalue of the orbital, which is the reference approximation to the particle binding energy according to Koopmans' theorem. The EP2 row corresponds to the second order propagator theory corrected binding energy.

Currently, openLowdin computes the third order corrections with six different formulas, denoted as P3, EP3, OVGF-A, OVGF-B, OVGF-C and REN-P3. Check J. Chem. Phys. 141, 114103 (2014) https://doi.org/10.1063/1.4895043 for the full details.

P3 and EP3 refer to the partial and full third order propagator, respectively. The OVGF-x are the multicomponent extensions of the outer valence Green function methods, which are renormalized EP3 results. REN-P3 is the renormalized third order partial propagator (P3).

As in PT2 calculations, the Pole Strength serves as a quantity that validates the diagonal (pseudoparticle) approximation employed. A P.S value below 0.85 usually indicates that the diagonal approximation is not reliable.

If not all the corrections are desired, you can select in the "CONTROL" block the type of correction to be used by adding "ptP3Method=" with "P3","EP3","OVGF-A","OVGF-B","OVGF-C" or "REN-P3" as options.

As in MBPT calculations, the CONTROL option "mpFrozenCoreBoundary" allows the user to omits a number of occupied electronic molecular orbitals (core electrons). For other species, the number of core orbitals along with the number of active virtuals, can be controlled with the "INPUT_CI" block in the input.

CHAPTER

TWENTYEIGHT

NEGATIVE MUONS

CHAPTER

TWENTYNINE

QUANTUM DRUDE OSCILLATORS, QDOS

CHAPTER

THIRTY

EXTERNAL POTENTIALS

CHAPTER

THIRTYONE

MOLDEN EXAMPLES

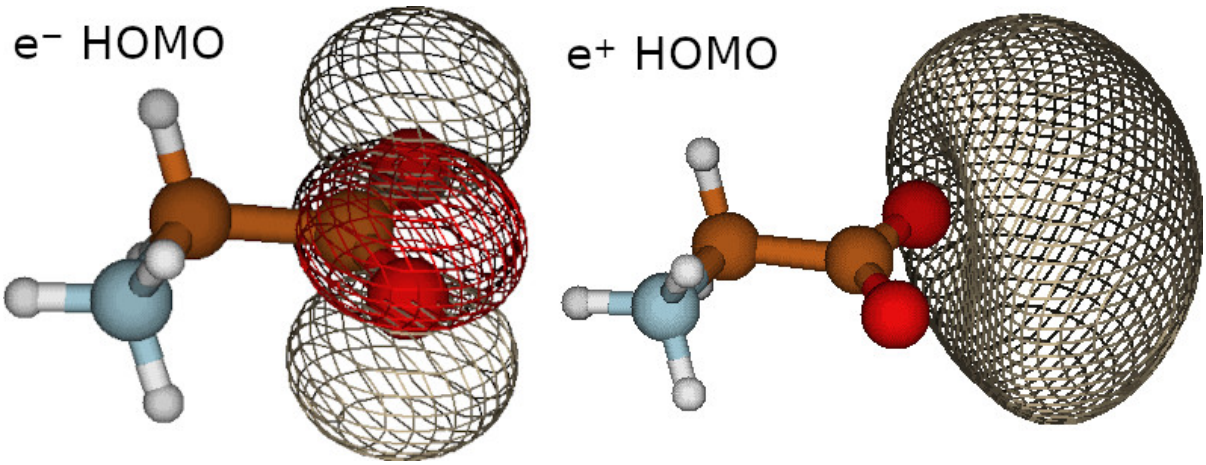
Running the following input of positronic glycine

```
SYSTEM_DESCRIPTION='Gly.e+-molden'

GEOMETRY
e-(N) 6-311G     0.851078    0.379034    0.538699
e-(C) 6-311G     0.011668   -0.714576    1.101899
e-(C) 6-311G     0.016098   -0.668656    2.636109
e-(O) 6-311G     0.714088    0.201754    3.154209
e-(O) 6-311G    -0.685842   -1.520986    3.175689
e-(H) 6-311G     1.816418    0.263714    0.801099
e-(H) 6-311G     0.409998   -1.655016    0.751139
e-(H) 6-311G    -0.992842   -0.593136    0.726189
e-(H) 6-311G     0.527967    1.271782    0.883152
e-(H) 6-311G     0.787819    0.376813   -0.469282
e+ PSX-TZ        0.714088    0.201754    3.154209
e+ PSX-TZ       -0.685842   -1.520986    3.175689 addParticles=-1
N dirac          0.851078    0.379034    0.538699
C dirac          0.011668   -0.714576    1.101899
C dirac          0.016098   -0.668656    2.636109
O dirac          0.714088    0.201754    3.154209
O dirac         -0.685842   -1.520986    3.175689
H dirac          1.816418    0.263714    0.801099
H dirac          0.409998   -1.655016    0.751139
H dirac         -0.992842   -0.593136    0.726189
H dirac          0.527967    1.271782    0.883152
H dirac          0.787819    0.376813   -0.469282
END GEOMETRY

TASKS
        method = "RHF"
END TASKS

OUTPUTS
        moldenFile
END OUTPUTS
```

Produces two molden files, one for the electrons and one for the positron, their filenames are provided in the output.

```
-----------------------------------------------------
MOLDENFILE              1
                   for all species
         FileName: Gly.e+-molden.E-.molden
         FileName: Gly.e+-molden.E+.molden
-----------------------------------------------------
```

These files contain the the electronic and positronic orbitals. We can visualize these orbitals using the molden software (https://www.theochem.ru.nl/molden/), as observed in the following figure



31.1 Localized orbitals and fchk files

Localized orbitals are generated with the Erkale software (https://github.com/susilehtola/erkale), adding the following lines in the CONTROL block

```
SYSTEM_DESCRIPTION='Gly.e+-localize-molden'

GEOMETRY
e-(N) 6-311G     0.851078    0.379034    0.538699
e-(C) 6-311G     0.011668   -0.714576    1.101899
e-(C) 6-311G     0.016098   -0.668656    2.636109
e-(O) 6-311G     0.714088    0.201754    3.154209
e-(O) 6-311G    -0.685842   -1.520986    3.175689
e-(H) 6-311G     1.816418    0.263714    0.801099
e-(H) 6-311G     0.409998   -1.655016    0.751139
e-(H) 6-311G    -0.992842   -0.593136    0.726189
e-(H) 6-311G     0.527967    1.271782    0.883152
e-(H) 6-311G     0.787819    0.376813   -0.469282
e+ PSX-TZ        0.714088    0.201754    3.154209
e+ PSX-TZ       -0.685842   -1.520986    3.175689 addParticles=-1
N dirac          0.851078    0.379034    0.538699
C dirac          0.011668   -0.714576    1.101899
C dirac          0.016098   -0.668656    2.636109
O dirac          0.714088    0.201754    3.154209
O dirac         -0.685842   -1.520986    3.175689
H dirac          1.816418    0.263714    0.801099
```

(continues on next page)

(continued from previous page)

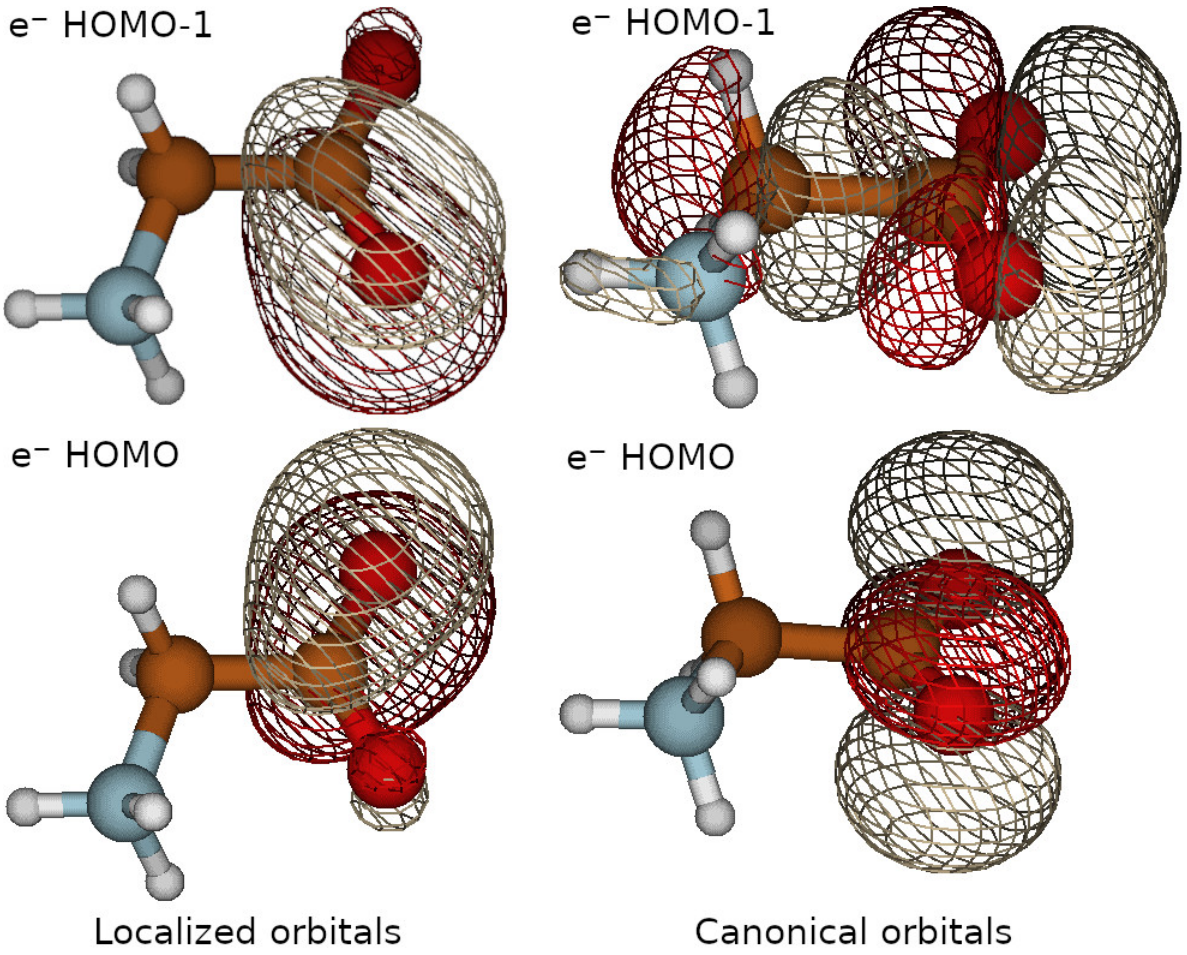
```
H dirac          0.409998   -1.655016    0.751139
H dirac         -0.992842   -0.593136    0.726189
H dirac          0.527967    1.271782    0.883152
H dirac          0.787819    0.376813   -0.469282
END GEOMETRY

TASKS
        method = "RHF"
END TASKS

CONTROL
        localizeOrbitals=.T.
        erkaleLocalizationMethod="MU"
END CONTROL

OUTPUTS
        moldenFile
END OUTPUTS
```

Here with "MU" we selected the Pipek-Mozay localization scheme using Mulliken charges. Check Erkale manual for a full list of the localization procedures available. To transfer the orbitals to Erkale, openLowdin generates fchk files. In the output, you will find the Erkale localization log.

```
 ------------------------------------------------------
            FCHKFILE    1
   for all species
          FileName: Gly.e+-localize-molden.E-.fchk
          FileName: Gly.e+-localize-molden.E+.fchk
 ------------------------------------------------------

  ERKALE ORBITAL LOCALIZATION
 ============================

ERKALE - Localization from Hel, serial version.
(c) Susi Lehtola, 2010-2016.

[...]

Localizing orbitals: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Initializing generalized Pipek-Mezey calculation with Mulliken charges... done.
Initialization of Pipek-Mezey took 0.00 s
  iter              J        delta J         <G,G>
     1   1.154017e+01   1.517727e+00  1.174829e+00   0.00 s
     2   1.274872e+01   1.208552e+00  1.673368e+00   0.00 s
[...]
    89   1.576852e+01   7.696281e-09  5.667527e-09   0.00 s
Converged.
Localization done in 0.06 s.
```

For this example, the localization procedure only affects the electronic orbitals, because there is only a single positronic orbital. When localized orbitals are requested, openlowdin will generate the molden files with them. Check in the

following figure a comparison between the non-localized (right) and localized (left) HOMO of positronic glycine



31.2 CI excited states

When we run a configuration interaction calculation we can generate molden files for the excited states. For example, in the *Positron covalent bond* (e+H-H-.CISDTQ-DZ.lowdin) example we added "state=2" to get the natural orbitals of the first excited state. In the output of that calculation we find

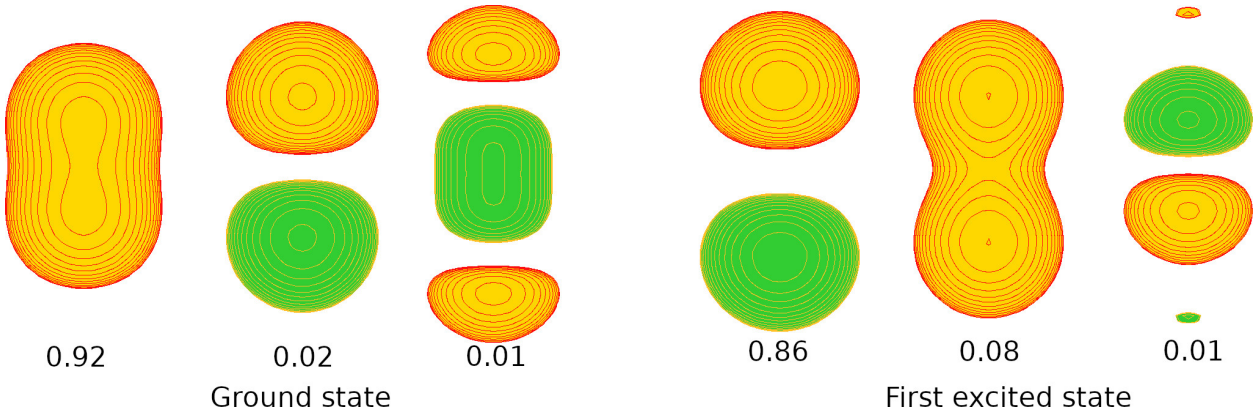
```
  We are printing molden files for the CI states!
-----------------------------------------------------------
        MOLDENFILE    1
  for all species
        FileName: e+H-H-.CISDTQ-DZ.E-ALPHA.molden
        FileName: e+H-H-.CISDTQ-DZ.E-BETA.molden
        FileName: e+H-H-.CISDTQ-DZ.E+.molden
-----------------------------------------------------------

  We are printing molden files for the CI states!
-----------------------------------------------------------
        MOLDENFILE    2
  for all species
for excited state:     2
```

(continues on next page)

(continued from previous page)

```
        FileName: e+H-H-.CISDTQ-DZ.E-ALPHA-s2.molden
        FileName: e+H-H-.CISDTQ-DZ.E-BETA-s2.molden
        FileName: e+H-H-.CISDTQ-DZ.E+-s2.molden
-------------------------------------------------------------
```

In the following figure we plot the positronic natural orbitals with the highest contributions to the ground (right) and first excited (left) states



CHAPTER

THIRTYTWO

INDICES AND TABLES

- genindex
- modindex
- search