with now two structures being read from file, and the `Docking approach` is labeled as `independent`, meaning each structure will be docked independently of each other.

After everything, the output is:

```
------------------------------------
LOWEST INTERACTION ENERGY: -18.482854 kcal/mol (structure 6)
------------------------------------

Total time for docking:       4.84 minutes

The lowest energy structure was 2, with energy -49.259349.
Docked structures saved to    Basename.docker.xyz
```

and one can see that the lowest interaction energy was that of structure 2 (the uracil), meaning it interacts strongly with the `HOST` than the water molecule given. Now the file `Basename.docker.xyz` will contain all final structures, ordered by interaction energy.

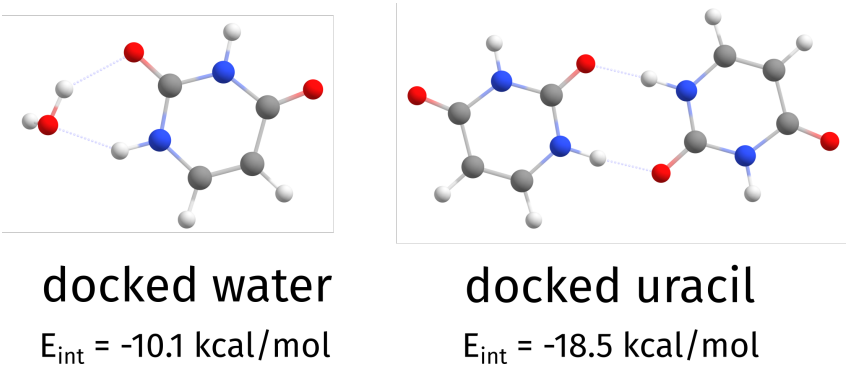

Fig. 4.21: Independent docking of water and uracil on top of an uracil molecule

Note

By default, the docking approach uses a fixed random seed and should always give the same result on the same machine. To make it always completely random add `%DOCKER RANDOMSEED TRUE END`.

Note

In order to use the faster GFN-FF instead of GFN2-XTB, use `!DOCK(GFNFF)`. For a quicker (and less accurate) docking, use `!QUICKDOCK`.

Note

To try multiple conformers of the `GUEST`, the ensemble file printed by GOAT `Basename.finalensemble.xyz` can be directly given here and the whole ensemble will be tested against a give `HOST`.

A detailed summary of the other options can be found on *Keywords*.

4.13.5 Underlying theory

The basic idea behind the DOCKER is to use a type of swarm intelligence [573] to find local minima for where the HOST would best be positioned, based on the energies and gradients of a given Potential Energy Surface (PES).

Swarm intelligence algorithms were invented trying to mimic the behavior of animals such as bees and ants, and it actually fits the problem here quite well (as shown below). The basic steps needed for the algorithm are:

1. Optimize HOST and GUEST;
2. Create a spacial grid around the HOST;
3. Initialize a possible set of random solutions;
4. Let the swarm intelligence find local minima;
5. Preoptimize with GFN-FF a large number of these minima;
6. Collect a fraction of the best solutions found on 4. and fully optimize them;
7. The structure with the lowest energy is considered the best solution.

To quickly demonstrate how the initial set of solutions "evolve" to find different local minima, take as an example the substituted biphenyl as HOST below. The figure below demonstrates how a water molecule (GUEST) is docked onto the initial HOST.

Each gray ball represents one possible placing of the water molecule (ignoring here its rotation angle). At the start, they are all distributed over the HOST without any bias. As the iterations go by, the algorithm starts to detect that both sides of the molecule have lower energy solutions than the aromatic rings, as expected, and the possible solutions start to converge to those points.

In the end, solutions containing water molecules binding to both sides are taken, the system is then fully optimized and one finds that the best biding occurs with the hydroxy group to the right.

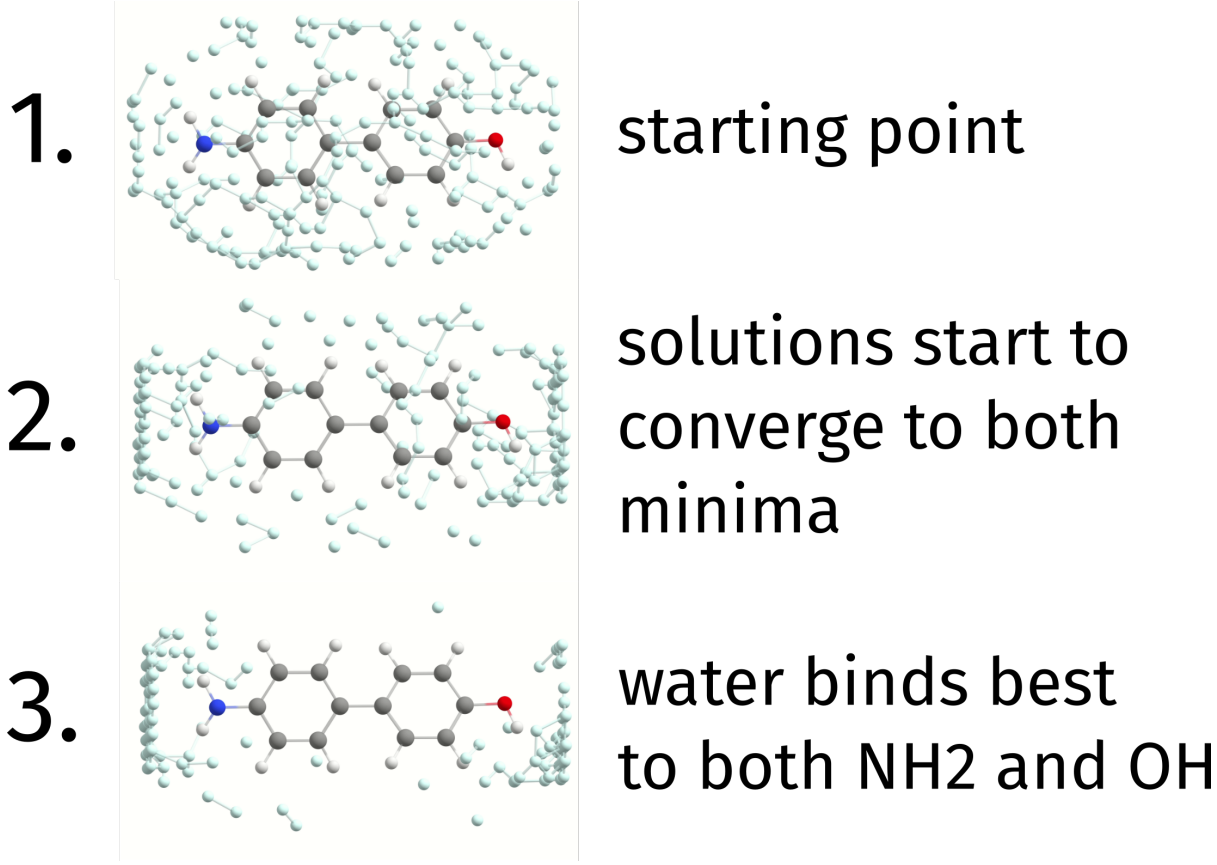

Fig. 4.22: The docking process of a water molecule on the substituted biphenyl.

Important

Right now the DOCKER can only be used together with the fast GFN-XTB methods, it will be generalized later. It is also not fully exploring the parallelization potential yet, and can be much faster for the next versions.

4.13.6 Looking Deeper into the Output

The details related to the grid and the swarm optimization can be found on the output, here is one example taken from the water dimer example from the earlier section:

```
Creating spatial grid
   Grid Max Dimension                5.50 Angs
   Angular Grid Step                 32.73 degrees
   Cartesian Grid Step               0.50 Angs
   Points per Dimension              11 points
Initializing workers
   Population Density                0.50 worker/Ang^2
   Population Size                   57
Swarm intelligence search
   Minimization Algorithm            mutant particle swarm
   Min, Max Iterations               (3 , 10)
```

The `Population Density` here defines how many solutions will be placed on the initial grid around the `HOST` and can be controlled by `%DOCKER SWARMPOPDENSITY 0.50 END`, while the population size is a direct consequence of that, unless specified by `SWARMPOPSIZE` under `%DOCKER`.

Another important piece of information is the `Min, Max Iterations`, which define the minimum number of iterations before checking for convergence, and the maximum possible number of iterations. These can be changed by setting `SWARMMINITER` and `SWARMMAXITER` under the `%DOCKER`.

Then during the swarm search itself (here called the "evolution step"), the `HOST` and `GUEST` are partially optimized using the same criteria as from `!SLOPPYOPT`, and their energy at convergence is taken as a convergence criteria. The printing, still for the previous example, reads:

Note

All of the optimizations respect constraints and/or wall potentials given on the `%GEOM` block. The `HOST` can be easily frozen with `%DOCKER FIXHOST TRUE END`.

```
   Iter    Emin          avDE         stdDE            Time
           (Eh)       (kcal/mol)    (kcal/mol)       (min)
   ---------------------------------------------------------

       1 -10.147462     2.756033     1.821981         0.03
       2 -10.147462     2.121389     1.610208         0.03
       3 -10.148583     2.313606     1.365227         0.03
       4 -10.148583     1.846998     1.188680         0.02
       5 -10.148583     1.587332     1.168207         0.02
No new minimum found after 3 (SwarmMinIter) steps.
```

Here we have the number of iterations, the energy of the best solution found during the process, the average energy difference from the lowest solutions and its standard deviation. The first number shows if the system is evolving towards a lower energy solutions and the later two give an idea of the "spread" of the energies found.

If no new minimum is found over `SWARMMINITER` iterations, the algorithm is considered to be converged and the best solutions are taken to the final optimization step.

4.13.7 The final steps

After the evolution step, a total of $max(sqrt(SwarmPopSize), 5)$ structures are taken from the set of solutions for a final full optimization. The output looks like:

```
Running final optimization
   Maximum number of structures      7
   Minimum energy difference         0.10 kcal/mol
   Maximum RMSD                      0.25 Angs
   Optimization strategy             regular
   Coordinate system                 redundant 2022
   Fixed host                        false
```

To avoid redundant solutions being optimized here, a check is made such that only structures with an energy difference of at least a `Minimum energy difference` and `Maximum RMSD` (obtained after an optimal rotation using the quaternion approach) are taken. The coordinate system is also be automatically set to Cartesian if there are more than 300 atoms in total. Set `AUTOCOORDSYS FALSE` to avoid that or `%GEOM COORDSYS CARTESIAN END` to use Cartesian coordinates all the way during the docking process.

If some optimizations fail during the process, they will be flagged as `optimization failed`, as in the example below:

```
   Struc    Eopt       Interaction Energy       Time
            (Eh)               (kcal/mol)      (min)
   ------------------------------------------------

    1      -10.149006          -4.968378       0.01
    2      -10.149005          -4.967965       0.01
    3       optimization failed                0.01
    4      -10.149007          -4.968641       0.01
   (...)
```

That only means some of the solutions failed to fully optimize at the end, and on the printed file their energy is set to 1000 Eh to show the job was not completed. If you want to increase the number of iterations, or change the final convergence criteria, just change them using the `%GEOM` block as usual.

4.13.8 Adding a bond bias to the docking process

The bond biases available for geometry optimization (see Section *Bond Bias Potentials*) can also be applied to the DOCKER, but the input here is slightly different.

As the DOCKER does not know the total number of atoms from the start, the `%DOCKER` block needs the first atom numbers from `GUEST` and `HOST` in that order and not the number of atoms from the final complex.

For example, to add a bond bias between atom 2 from the `GUEST` and atom 19 from the `HOST` during the docking process, please use:

```
%DOCKER
  BIAS { B 2 19 } END
END
```

The same parameters and their defaults from the `%GEOM` block are used here.

Note

This will also be applied to the SOLVATOR if the `CLUSTERMODE DOCKING` (default) is used. The `GUEST` atom number corresponds to that of the solvent, as printed in the output.

Important

If biases are added from the %DOCKER block, they will override any other bias from the %GEOM block.

4.13.9 Defining the center and extent of the grid

As explained above, the placement of GUEST molecules is done with the help of a spatial grid. By default, the grid is built around the centroid of the host molecule. However, in cases a different position for the grid is wanted, it can be changed by setting an arbitrary center and extents.

Its center can be defined in terms of Cartesian coordinates simple via:

```
%DOCKER
  GRIDCENTER 1.00, 0.642, -1.234 # coordinates in Angstroem
END
```

or a list of atoms can also be given, and the centroid of those atoms will be taken. If a single atom is given, that will of course be the center:

```
%DOCKER
  GRIDCENTERATOMS {0:5 7} END # numbers starting from 0, as usual
END
```

Important

If GRIDCENTERATOMS were given and the HOST is optimized, which is the default case, the grid center will be defined only **after** the geometry is optimized. For the GRIDCENTER option, the center will be fixed at the desired Cartesian coordinates.

The extent of the grid around that center can be controlled by:

```
%DOCKER
  GRIDEXTENT 15 # extent in Angstroem
END
```

and all x, y and z coordinates will expand that much from the center. It has to be a cube, and all dimensions will be the same. If an extent is not given, the default values will be used.

Note

There is no problem setting the grid extent to regions where atoms are present. The grid will be only built around these, same as for the default case.

4.13.10 General Tips

1. In order to quick change the PES of the docking process, you can use !DOCK(GFN2-XTB), !DOCK(GFN1-XTB),!DOCK(GFN0-XTB) or !DOCK(GFN-FF).
2. For a quicker and less accurate docking the input !QUICKDOCK is also available.
3. There are four levels of "docking" procedure, from simple to more elaborate: SCREENING, QUICK, NORMAL (default) and COMPLETE, which can be given as %DOCK DOCKLEVEL COMPLETE END.
4. To increase the final number of structures optimized in the end, just change NOPT under %DOCKER.
5. If the guest topology is somehow changed during the docking process, that structure will be discarded. This can be turned off by setting %DOCK CHECKGUESTTOPO FALSE END.

Important

By the time of the ORCA6.1 release, this algorithm was still not published. Publications are under preparation.

4.13.11 Keywords

A collection of GOAT related simple input keywords is given in Table 4.11. All `%docker` block options are summarized in Table 4.12.

Table 4.11: Simple input keywords for the DOCKER algorithm.

Keyword	Description
`QUICKDOCK`	Set `DOCKLEVEL QUICK`
`NORMALDOCK`	Set `DOCKLEVEL NORMAL`
`COMPLETEDOCK`	Set `DOCKLEVEL COMPLETE`
`DOCK(GFN-FF)`	Set `EVPES GFNFF`
`DOCK(GFN0-XTB)`	Set `EVPES GFN0XTB`
`DOCK(GFN1-XTB)`	Set `EVPES GFN1XTB`
`DOCK(GFN2-XTB)`	Set `EVPES GFN2XTB`

Table 4.12: `%docker` block input keywords for t

Keyword	Options	Description
`GUEST`	`"filename.xyz"`	An .xyz file (can be multistructure), from where the guest(s) will be read. Can contain different charges ar
`DOCKLEVEL`	`screening`	Defines a general strategy for docking.
	`quick`	Will alter things like that population density and final number of optimized structrures. default is `normal`
	`normal`	
	`complete`	
`PRINTLEVEL`	`low`	
	`normal`	Default
	`high`	Will print many extra files!
`NREPEATGUEST`	`1`	Number of times to repeat the content of the “GUEST” file
`CUMULATIVE`	`true`	Add the contents of the “GUEST” file one on top of each other? default is `false`, meaning each will be
`GUESTCHARGE`	`0`	Can be used to defined a charge for the guest (default 0)
`GUESTMULT`	`3`	Same for multiplicty (default 1) both can also be given via the comment line of the “filename.xyz”, see ab
`RAMDOMSEED`	`true`	Whether to allow for the process to be trully random and will give different results everytime (default `fal`
`FIXHOST`	`true`	Freeze coordinatef for the HOST during all steps? (default `false`)
`OPTLEVEL`	`sloppyopt`	Use default optimization criteria inside the DOCKER
	`looseopt`	
	`normalopt`	
`AUTOCOORDSYS`	`false`	Automatically use Cartesian coordinates for the docking process (default `false`)
`ALLOWMETALCOORD`	`true`	will allow the DOCKER to place atoms close enough to metals such as to create coordination bonds. by d
`MBONDFAC`	`1.0`	A multiplying factor for the radii when defining a bond with metals. Default is 1.0
Grid options:		
`GRIDCENTER`	`1.00, 0.642, -1.234`	Cartesian coordinates for the center in Angström
`GRIDCENTERATOMS`	`{0:5 7} end`	A list of atoms from which the centroid will be taken
`GRIDEXTENT`	`15`	The extent, from the center, in one dimension in Angström
Swarm intelligence step:		
`POPDENSITY`	`5.0`	Population density per Angström squared. Default depends on the Docking level
`POPSIZE`	`500`	A fixed number for the population size
`SWARMMAXITER`	`10`	Maximum number of iterations
`SWARMMINITER`	`3`	Minimum number to check for convergence and minimum required of equal steps to signal convergence
`SWARMPOPDENSITY`	`0.5`	The population density based on the HOST grid
`SWARMPOPSIZE`	`150`	A fixed number for the swarm population size
`SWARMPES`	`cdp`	Which PES to use **only** during the evolution step. The default `cdp` is **a lot** faster than the others.
	`gfnff`	Can be different from the final optimization.
	`gfn0xtb`	
	`gfn1xtb`	
	`gfn2xtb`	
`CHECKTOPO`	`false`	Check the topology of both host and guest during the docking process? Default is true.
Pre optimizations		
`PREOPT`	`false`	Controls whether the PreOpt step will be done. Default `true`.
Final optimizations:		
`NOPT`	`10`	A fixed number of structures to be optimized
`NOOPT`	`false`	Do not optimize any structure at all? (default `false`)

CHAPTER

FIVE

SPECTROSCOPY AND PROPERTIES

5.1 Population Analysis

Atomic population related quantities are not real molecular properties since they are not observables. They are nevertheless highly useful for interpreting experimental and computational findings. Various ways of analyzing a computed SCF wavefunction are available within ORCA. By default, ORCA provides very detailed information about calculated molecular orbitals and bonds through *Mulliken*, *Löwdin*, and *Mayer* population analyses.

See also

As these methods typically create a very large amount of specific output, we generally recommend to read the *Control of Output* section.

The `!ReducedPOP` keyword is of particular use as it reduces the printed information in the population analysis section, providing orbital population of each atom with percent contribution per basis function type. This is highly useful in figuring out the character of the MOs.

5.1.1 Mulliken Population Analysis

The Mulliken population analysis [574] is, despite all its known considerable weaknesses, the standard in most quantum chemical programs. It partitions the total density using the assignment of basis functions to given atoms in the molecules and the basis function overlap. If the total charge density is written as $\rho(\vec{r})$ and the total number of electrons is N we have:

$$\int \rho(\vec{r})\, d\vec{r} = N \tag{5.1}$$

and from the density matrix $\mathbf{P}$ and the basis functions ϕ:

$$\rho(\vec{r}) = \sum_{\mu\nu} P_{\mu\nu}\phi_\mu(\vec{r})\,\phi_\nu(\vec{r}) \tag{5.2}$$

therefore:

$$\int \rho(\vec{r})\, d\vec{r} = \sum_{\mu\nu} P_{\mu\nu} \underbrace{\int \phi_\mu(\vec{r})\,\phi_\nu(\vec{r})\, d\vec{r}}_{S_{\mu\nu}} \tag{5.3}$$

$$= \sum_{\mu\nu} P_{\mu\nu} S_{\mu\nu} \tag{5.4}$$

After assigning each basis function to a given center this can be rewritten:

$$= \sum_A \sum_B \sum_\mu {}^A \sum_\nu {}^B P^{AB}_{\mu\nu} S^{AB}_{\mu\nu} \tag{5.5}$$

$$= \sum_A \sum_\mu {}^A \sum_\nu {}^A P_{\mu\nu}^{AA} S_{\mu\nu}^{AA} + 2 \sum_A \sum_{B<A} \sum_\mu {}^A \sum_\nu {}^B P_{\mu\nu}^{AB} S_{\mu\nu}^{AB} \tag{5.6}$$

Mulliken proposed to divide the second term equally between each pair of atoms involved and define the number of electrons on center A, N_A, as:

$$N_A = \sum_\mu {}^A \sum_\nu {}^A P_{\mu\nu}^{AA} S_{\mu\nu}^{AA} + \sum_{B\neq A} \sum_\mu {}^A \sum_\nu {}^B P_{\mu\nu}^{AB} S_{\mu\nu}^{AB} \tag{5.7}$$

such that $\sum_A N_A = N$. The charge of an atom in the molecule is then:

$$Q_A = Z_A - N_A \tag{5.8}$$

where Z_A is the core charge of atom A. The cross terms between pairs of basis functions centered on different atoms is the overlap charge and is used in ORCA to define the Mulliken bond order:

$$B_{AB} = 2 \sum_\mu {}^A \sum_\nu {}^B P_{\mu\nu}^{AB} S_{\mu\nu}^{AB} \tag{5.9}$$

Basic Usage

The Mulliken population analysis can be invoked by the `!MULLIKEN` input keyword:

```
! MULLIKEN
```

It can further be requested via the `Print[ P_Mulliken ] 1` keyword in the `%output` block

```
%output
  Print[ P_Mulliken ] 1  # default = on
end
```

A number of additional options can be specified in the `%output` block to control the details of the Mulliken population analysis. By default the Mulliken population analysis is turned on.

```
%output
  Print[ P_AtCharges_M  ] 1     # Print atomic charges
  Print[ P_OrbCharges_M ] 1     # Print orbital charges
  Print[ P_FragCharges_M] 1     # Print fragment charges
  Print[ P_BondOrder_M  ] 1     # Print bond orders
  Print[ P_FragBondOrder_M ] 1  # Print fragment b.o.
  Print[ P_ReducedOrbPop_M ] 1  # Print reduced orb. Charges
  Print[ P_AtPopMO_M    ] 1     # Print atomic charges in each MO
  Print[ P_OrbPopMO_M   ] 1     # Print orbital populaiton for each MO
  Print[ P_ReducedOrbPopMO_M] 1 # Print reduced orbital pop for each MO
  Print[ P_FragPopMO_M  ] 1     # Print the fragment population for for each MO
end
```

These options allow to get very detailed information about the computed wavefunctions and is much more convenient than to look at the MOs directly. A "reduced orbital population" is a population per angular momentum type. For example the sum of populations of each p_z orbital at a given atom is the reduced orbital population of the p_z function.

Note that for *finite temperature HF or KS-DFT* calculations (`SmearTemp` > 0 K, fractional occupation numbers or *FOD analysis*), only the Mulliken reduced orbital charges based on ρ^{FOD} will be printed.

5.1.2 Löwdin Population Analysis

The Löwdin analysis [180] is somewhat more straightforward than the Mulliken analysis. In the Löwdin method one changes to a basis where all overlap integrals vanish. This is accomplished via Löwdins symmetric orthogonalization matrix $\mathbf{S}^{-1/2}$. Using this transformation matrix the new basis functions are multicentered but are in a least square sense as similar as possible to the original, strictly localized, atomic basis functions. The similarity of the transformed functions and original functions is explored in the population analysis. The density matrix transforms as:

$$\mathbf{P}^L = \mathbf{S}^{1/2}\mathbf{P}\mathbf{S}^{1/2} \tag{5.10}$$

Then the atomic populations are:

$$N_A = \sum_{\mu}{}^{A} P^L_{\mu\mu} \tag{5.11}$$

The bond order is defined from the Wiberg index [575] that was first used in the context of semiempirical methods (that are formulated in the Löwdin basis right from the start):

$$B_{AB} = \sum_{\mu}{}^{A} \sum_{\nu}{}^{B} \left(P^L_{\mu\nu}\right)^2 \tag{5.12}$$

Basic Usage

The Löwdin population analysis can be invoked by the `!LOEWDIN` input keyword. By default the Löwdin population analysis is turned on and provides some more detail than the Mulliken analysis.

```
! MULLIKEN
```

It can further be requested via the `Print[ P_Loewdin ] 1` keyword in the `%output` block:

```
%output
  Print[ P_Loewdin ] 1  # default = on
end
```

The details of the Löwdin population analysis printout can be controlled via the `%output` block:

```
%output
  Print[ P_AtCharges_L  ] 1     # Print atomic charges
  Print[ P_OrbCharges_L ] 1     # Print orbital charges
  Print[ P_FragCharges_L] 1     # Print fragment charges
  Print[ P_BondOrder_L  ] 1     # Print bond orders
  Print[ P_FragBondOrder_L ] 1  # Print fragment b.o.
  Print[ P_ReducedOrbPop_L ] 1  # Print reduced orb. Charges
  Print[ P_AtPopMO_L    ] 1     # Print atomic charges in each MO
  Print[ P_OrbPopMO_L   ] 1     # Print orbital population for each MO
  Print[ P_ReducedOrbPopMO_L] 1 # Print reduced orbital pop for each MO
  Print[ P_FragPopMO_L  ] 1     # Print the fragment population for each MO
end
```

In addition one can set the threshold for the printing of the bond order in the `%method` block.

```
%method
  LOEWDIN_BONDORDERTHRESH 0.05
end
```

5.1.3 Frontier Molecular Orbital Populations

As the frontier orbitals are typically of specific interest, *Mulliken* and *Löwdin* populations of the HOMO and LUMO can be requested via the "FMOPop" keyword:

```
! FMOPop
```

The respective Mulliken and Loewdin population of the HOMO and LUMO frontier orbitals will be printed in the output:

```
-------------------------------------------
FRONTIER MOLECULAR ORBITAL POPULATION ANALYSIS
-------------------------------------------

ANALYZING ORBITALS: HOMO=   6 LUMO=   7

-----------------------------------------------------------------------------
   Atom        Q(Mulliken)     Q(Loewdin)     Q(Mulliken)     Q(Loewdin)
              <<<<<<<<<<<HOMO>>>>>>>>>>>   <<<<<<<<<<<LUMO>>>>>>>>>>>
-----------------------------------------------------------------------------
   0-C         0.937186        0.906827       0.804044        0.755610
   1-O         0.062814        0.093173       0.195956        0.244390
-----------------------------------------------------------------------------
```

5.1.4 Mayer Population Analysis

Mayers bonding analysis [576, 577, 578, 579] is another creative attempt to define chemically useful indices. The Mayer atomic charge is identical to the Mulliken charge. The Mayer bond order is defined as:

$$B_{AB} = \sum_{\mu}^{A} \sum_{\nu}^{B} (\mathbf{PS})_{\mu\nu} (\mathbf{PS})_{\nu\mu} + (\mathbf{RS})_{\mu\nu} (\mathbf{RS})_{\nu\mu} \tag{5.13}$$

Here **P** is the total electron density matrix and **R** is the spin-density matrix. These Mayer bond orders are very useful. Mayer's total valence for atom A is defined as:

$$V_A = 2N_A - \sum_{\mu}^{A} \sum_{\nu}^{A} (\mathbf{PS})_{\mu\nu} (\mathbf{PS})_{\nu\mu} \tag{5.14}$$

In normal bonding situations and with normal basis sets V_A should be reasonably close to the valence of atom A in a chemical sense (i.e. close to four for a carbon atom). The bonded valence is given by:

$$X_A = V_A - \sum_{B \neq A} B_{AB} \tag{5.15}$$

and finally the free valence (a measure of the ability to form further bonds) is given by:

$$F_A = V_A - X_A \tag{5.16}$$

Basic Usage

The Mayer population analysis can be invoked via the !MAYER keyword:

```
! MAYER
```

It can further be requested via the Print[P_Mayer] 1 keyword in the %output block:

```
%output
  Print[ P_Mayer ] 1  # default = on
end
```

The output is rather simple and short and can not be further controlled. By default the Mayer population analysis is turned on. In addition one can set the threshold for the printing of the bond order in the `%method` block.

```
%method
  MAYER_BONDORDERTHRESH 0.1
end
```

5.1.5 Natural Population Analysis

If you have access to a version of the `gennbo` program from Weinhold's group[1] you can request the Natural Population analysis via the *NBO interface*.

Important

The NPA is only performed if the NBO program is provided properly. If not, ORCA will skip the NPA analysis.

Basic Usage

The Natural population analysis can be invoked via the `!NPA` keyword:

```
! NPA
```

It can further be requested via the `Print[ P_NPA ] 1` keyword in the `%output` block:

```
%output
  Print[ P_NPA ] 1  # default = off
end
```

5.1.6 Hirshfeld Population Analysis

The partitioning method by Hirshfeld is one of the most used approaches in the so-called atoms in molecules (AIM) methods.[580] In this case, the AIM density of atom A, $\rho_A(\vec{r})$ is written as:

$$\rho_A(\vec{r}) = \rho(\vec{r}) w_A(\vec{r}) \tag{5.17}$$

Here, $\rho(\vec{r})$ is the total charge density at position $\vec{r}$, and $w_A(\vec{r})$ a weighting function, that within the Hirshfeld method is equal to:

$$w_A(\vec{r}) = \frac{\rho^0_A(\vec{r})}{\rho^0(\vec{r})} \tag{5.18}$$

where $\rho^0_A(\vec{r})$ is the pro-atomic density of atom A and $\rho^0(\vec{r}) = \sum_A \rho^0_A(\vec{r})$ the pro-molecular density. The ratio in eq. (5.17) is known as *stockholder*. From eqs. (5.17) and (5.18) one can calculate the Hirshfeld charges as:

$$Q_A^{\text{Hirsh.}} = Z_A - \int \rho_A(\vec{r}) d\vec{r} \tag{5.19}$$

In ORCA, the pro-atomic density within the Hirshfeld method is calculated via density fitting with a set of Gaussian s-functions per element.

[1] Information about the NBO program can be found at http://nbo7.chem.wisc.edu

Basic Usage

The Hirshfeld population analysis can be invoked by the `!HIRSHFELD` input keyword.

```
! HIRSHFELD
```

It can further be requested via the `Print[ P_Hirshfeld ] 1` keyword in the `%output` block:

```
%output
  Print[ P_Hirshfeld ] 1  # default = off
end
```

Example

As an example we request the Hirshfeld charges for a water molecule:

```
!HF cc-pVDZ TightSCF HIRSHFELD

* xyz 0 1
  O    0.00000006589375       0.00157184228646       0.00000000004493
  H    0.77316868532439      -0.58666889665624      -0.00000000000005
  H   -0.77316876182122      -0.58666895650640      -0.00000000000005
*
```

ORCA prints the following information in the output file:

```
------------------
HIRSHFELD ANALYSIS
------------------

Total integrated alpha density =      4.999998580
Total integrated beta density  =      4.999998580

  ATOM     CHARGE      SPIN
   0 O   -0.333756    0.000000
   1 H    0.166879    0.000000
   2 H    0.166879    0.000000

  TOTAL   0.000003    0.000000
```

5.1.7 MBIS Charges

The Minimal Basis Iterative Stockholder (MBIS) method is a variant of the Hirshfeld method.[581] The idea behind this approach is that the pro-atomic density $\rho_A^0(\vec{r})$ is expanded in a minimal set of atom-centered s-type Slater functions $\rho_{Ai}^0(\vec{r})$:

$$\rho_A^0(\vec{r}) = \sum_{i=1}^{m_A} \rho_{Ai}^0(\vec{r}) \tag{5.20}$$

with $\rho_{Ai}^0(\vec{r})$ equal to:

$$\rho_{Ai}^0(\vec{r}) = \frac{N_{Ai}}{\sigma_{Ai}^3 8\pi} \exp\left(-\frac{\left|\vec{r} - \vec{R_A}\right|}{\sigma_{Ai}}\right) \tag{5.21}$$

Here, m_A is the number of shells of atom A. The populations N_{Ai}, and the widths σ_{Ai} can be written as:

$$N_{Ai} = \int \rho(\vec{r}) \frac{\rho_{Ai}^0(\vec{r})}{\rho^0(\vec{r})} d\vec{r} \tag{5.22}$$

$$\sigma_{Ai} = \frac{1}{3N_{Ai}} \int \rho(\vec{r}) \frac{\rho^0_{Ai}(\vec{r})}{\rho^0(\vec{r})} \left| \vec{r} - \vec{R_A} \right| d\vec{r} \tag{5.23}$$

In order to compute the AIM densities $\rho_A(\vec{r})$, the MBIS method uses an iterative algorithm where: (1) an initial guess is generated for the set of N_{Ai} and σ_{Ai} and the pro-atomic densities are calculated through eqs. (5.20) and (5.21), (2) the new set of N_{Ai} and σ_{Ai} are obtained via eqs. (5.22) and (5.23), (3) if convergence is reached for $\rho_A(\vec{r})$, the iterative process stops, otherwise we go back to (1) but now one uses the last estimates for N_{Ai} and σ_{Ai}.

Once, the MBIS iterative process stops, the MBIS charges are calculated as:

$$Q_A^{\text{MBIS.}} = Z_A - \int \rho_A(\vec{r}) d\vec{r} \tag{5.24}$$

Basic Usage

The MBIS population analysis can be invoked by the `!MBIS` input keyword.

```
! MBIS
```

It can further be requested via the `Print[ P_MBIS ] 1` keyword in the `%output` block:

```
%output
  Print[ P_MBIS ] 1  # default = off
end
```

The convergence threshold for the MBIS charges is set to 10^{-6}. However, it can be changed via the tag `MBIS_CHARGETHRESH` in the `%method` block:

```
%method
  MBIS_CHARGETHRESH 0.0001
end
```

MBIS Quantities

ORCA can also print the following MBIS-related quantities:

1. atomic dipole moments,
2. atomic quadrupole moments,
3. atomic octupole moments, and
4. third radial moment of the MBIS density $\left(\langle r^3 \rangle_A = \int \left| \vec{r} - \vec{R_A} \right|^3 \rho_A(\vec{r}) d\vec{r} \right)$.

The printing of these properties is controlled by the `MBIS_LARGEPRINT` in the `%method` block:

```
%method
  MBIS_LARGEPRINT true  # default = false
end
```

If this option is activated, an extra iteration is performed after reaching the convergence threshold for the charges.

The origin for the calculation of the atomic dipole, quadrupole and octupole moments is the center of each atom (default). The user can also define a global origin (independent of the atom) through the `MBIS_ORIGIN_MULT` keyword in the `%method` block:

```
%method
  MBIS_ORIGIN_MULT  CenterOfCoords     # origin of coordinate system (0,0,0)
                    CenterOfMass       # center of mass
                    CenterOfNucCharge  # center of nuclear charge
                    CenterXYZ          # arbitrary position, set coordinates with␣
```

(continues on next page)

(continued from previous page)

```
↪MBIS_ORIMULT_XYZ
                    CenterOfEachAtom   # center of each atom (default)

  MBIS_ORIMULT_XYZ  x,y,z # set the coordinates, otherwise 0,0,0 (unit: Angstrom)
end
```

Example

If we request the MBIS charges for a HF calculation at the cc-pVDZ level of a chloroform molecule:

```
!HF cc-pVDZ TightSCF MBIS

* xyz 0 1
  C   -0.00000997794639     -0.00091664148112      0.45499807439812
  H    0.00000069467312      0.00031189002174      1.53703126401237
  Cl   0.00003188789531      1.69433732001280     -0.08420513240263
  Cl   1.46635420502892     -0.84684178730039     -0.08421103795485
  Cl  -1.46637680965097     -0.84689178125304     -0.08420916805301
*
```

ORCA prints the following information at the end of the output file:

```
------------------
MBIS ANALYSIS
------------------

Convergence threshold (charges)      ...        1.000e-06
Number of iterations                 ...        46

Total integrated alpha density       ...     29.000001385
Total integrated beta density        ...     29.000001385

  ATOM     CHARGE    POPULATION     SPIN
   0 C    0.208633    5.791367    0.000000
   1 H    0.169417    0.830583    0.000000
   2 Cl  -0.126877   17.126877    0.000000
   3 Cl  -0.125586   17.125586    0.000000
   4 Cl  -0.125590   17.125590    0.000000

  TOTAL  -0.000003   58.000003    0.000000

 MBIS VALENCE-SHELL DATA:
  ATOM   POPULATION   WIDTH(A.U.)
   0 C    4.122213    0.508675
   1 H    0.830583    0.358785
   2 Cl   8.532439    0.524031
   3 Cl   8.531380    0.523959
   4 Cl   8.531381    0.523959
```

The second block corresponds to the valence Slater function, which is caracterized by its population $N_{A,v}$ and width $\sigma_{A,v}$.

5.1.8 CHELPG Charges

Atomic charges can also be calculated using the CHarges from ELectrostatic Potentials using a Grid-based (CHELPG) method according to Breneman and Wiberg.[582] In this approach, the atomic charges are fitted to reproduce the electrostatic potential on a regular grid around the molecule, while constraining the sum of all atomic charges to the molecule's total charge. An additional constraint can be added, so the CHELPG charges also reproduce the total dipole moment of the molecule.

Basic Usage

In ORCA the CHELPG charges can be (i) requested within a calculation, or (ii) calculated via the standalone *orca_chelpg* utility. To follow path (i) one has to add the `!CHELPG` simple input keyword to the input file.

```
! CHELPG
```

Further settings can be controlled via the `%chelpg` block.

```
%chelpg
  GRID 0.3       # Spacing of the regular grid in Angstroems
  RMAX 2.8       # Maximum distance of all atoms to any gridpoint in Angstroems
  VDWRADII COSMO # VDW Radii of Atoms, COSMO default
           BW    # Breneman, Wiberg radii
  DIPOLE FALSE   # If true, then the charges also reproduce the total dipole moment
end
```

By default the program uses the COSMO VDW radii for the exclusion of gridpoints near the nuclei, as these are defined for all atoms. The BW radii are similar, but only defined for very few atom types.

The charges may exhibit some dependence on the molecule's orientation in space, or some artificial variations in symmetric molecules. These effects can be minimized by increasing the CHELPG grid size, either by setting the `GRID` parameter in the `%chelpg` block, or by using the `!CHELPG(LARGE)` simple input keyword.

```
! CHELPG(LARGE)
```

If one wants that the calculated CHELPG charges reproduce the total dipole moment of the molecule, as well as the electrostatic potential, then the following tag has to be added to the `%chelpg` block:

```
%chelpg
  DIPOLE TRUE  # The default is set to FALSE
end
```

In particular, the constraint affects the x,y,z components of the total dipole moment, so they reproduce the exact three components of the total dipole moment calculated via one-electron integrals.

5.1.9 RESP Charges

ORCA also features the possibility of calculating the so-called RESP charges [583], which are essentially CHELPG charges with some restraint to avoid over fitting. These can be invoked with the defaults by simply using `!RESP` on the main input.

These RESP charges can have two types of penalty function: quadratic or hyperbolic (default), as described on the original paper. These require up to two penalty values, which are by default given by $a = 0.0005$ and $b = 0.1$.

These values can be changed inside the `%CHELPG` block by using:

```
%chelpg
  RESPA 0.01   # for the a value
  RESPB 0.002  # for the b value
end
```

and the penalty function can be chosen with:

```
%chelpg
  RESPPENALTY QUADRATIC # default is HYPREBOLIC
end
```

Also, if needed, the individual values for the a parameters and reference charges q_0 can be given via a special file with:

```
%chelpg
  RESPWFILE     "wfilename"   # these will define the weights (or a parameters)␣
↪for each atom
  RESPQREFFILE  "qfilename"   # these might define reference charges different␣
↪from zero
end
```

The format for both these files is similar to a `.xyz` file, where the first line has the number of atoms, the second line is just a comment, and every subsequent line contains a single number which is either the weights or the reference charges per atom. They can not be given on a single file, there must be two different ones.

Note

The results from the RESP files obtained here are **not** necessarily the same one would get from other software. Two main reasons are: the grids are not the same and some other software use a multilevel approach, first starting with a certain value for the weights and then switching to a different value, which is not the case here.

5.1.10 Local Spin Analysis

It is common practice in chemistry to think about the interaction of open-shell systems in terms of local spin states. For example, in dimeric or oligomeric transition metal clusters, the 'exchange coupling' between open shell ions that exist locally in high-spin states is extensively studied. Diradicals would be typical systems in organic chemistry that show this phenomenon. In quantum mechanics, however, the total spin is not a local property, but instead a property of the system as a whole. The total spin squared, S^2, and its projection onto the z-axis, S_z, commute with the non-relativistic Hamiltonian and hence, the eigenfunctions of the non-relativistic Hamiltonian can be classified according to good quantum numbers S and M according to:

$$\mathbf{S}^2 \left|\Psi^{SM}\right\rangle = S(S+1) \left|\Psi^{SM}\right\rangle$$

$$S_z \left|\Psi^{SM}\right\rangle = M \left|\Psi^{SM}\right\rangle$$

where $\left|\Psi^{SM}\right\rangle$ is an exact eigenfunction of the non-relativistic Hamiltonian or an approximation to it that conserves the total spin as a good quantum number. The total spin itself is given by the sum over the individual electron spins as:

$$\mathbf{S} = \sum_i \mathbf{s}(i)$$

And hence,

$$\mathbf{S}^2 = \sum\nolimits_{i,j} \mathbf{s}(i)\mathbf{s}(j)$$

is a two-electron property of the system. It is obviously not trivial to relate the chemically very meaningful concept of local spin to a rigorous quantum mechanical treatment. While there are various proposals of how to deal with this problem, we follow here a proposal of Clark and Davidson[584]. The following equations are implemented in the SCF and CASSCF modules of Orca.

Clark and Davidson define fragment projection operators with the property:

$$P_A P_B = \delta_{AB} P_A$$

and:

$$\sum_A P_A = 1$$

Then using this identity:

$$\mathbf{S} = \sum_i \sum_A \mathbf{s}(i) P_{A(i)}$$

$$\mathbf{S} = \sum_A \sum_i \mathbf{s}(i) P_A(i)$$

$$= \sum_A \mathbf{S}_A(i)$$

they show that the local spin operators obey the standard relations for spin operators:

$$\mathbf{S}_A = \mathbf{S}_A^\dagger$$

$$\mathbf{S}_A \times \mathbf{S}_A = i\hbar \mathbf{S}_A$$

Hence

$$\mathbf{S}^2 = \sum_A \sum_B \mathbf{S}_A \mathbf{S}_B$$

But then importantly:

$$\mathbf{S}_A \mathbf{S}_B = \sum_i \sum_j \mathbf{s}(i)\mathbf{s}(j) P_A(i) P_B(j)$$

$$= \tfrac{3}{4}\delta_{AB} \sum_A P_A(i) + \sum_i \sum_{j>i} \mathbf{s}(i)\mathbf{s}(j)\{P_A(i)P_B(j) + P_A(j)P_B(i)\}$$

With the first- and second-order density matrix:

$$\gamma(\mathbf{x}, \mathbf{x}') = N \int \Psi(\mathbf{x}, \mathbf{x}_2, ..., \mathbf{x}_N)\Psi^*(\mathbf{x}', \mathbf{x}_2, ..., \mathbf{x}_N) d\mathbf{x}_2 ... d\mathbf{x}_N$$

$$\Gamma(\mathbf{x}_1, \mathbf{x'}_1; \mathbf{x}_2, \mathbf{x'}_2) = \binom{N}{2} \int \Psi(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)\Psi^*(\mathbf{x'}_1, \mathbf{x'}_2, ..., \mathbf{x}_N) d\mathbf{x}_3 ... d\mathbf{x}_N$$

(with $\binom{N}{2} = \frac{1}{2}N(N-1)$). Then:

$$\langle \mathbf{S}_A \mathbf{S}_B \rangle = \tfrac{3}{4}\delta_{AB} tr(\gamma P_A) + 2tr(P_A(1)P_B(2)\mathbf{s}(1)\mathbf{s}(2)\Gamma(1,1;2,2))$$

In terms of the number of electrons on site 'A'and the expectation value of S_z^A

$$\left\langle S_z^A \right\rangle = \tfrac{1}{2} tr(\gamma^{\alpha-\beta} P_A)$$

$$\left\langle N^A \right\rangle = tr(\gamma^{\alpha+\beta} P_A)$$

in terms of molecular orbitals:

$$\left\langle S_z^A \right\rangle = \tfrac{1}{2} \sum_{p,q} \gamma_{pq}^{\alpha-\beta} \langle p|P_A|q\rangle$$

$$\left\langle N^A \right\rangle = \sum_{p,q} \gamma_{pq}^{\alpha+\beta} \langle p|P_A|q\rangle$$

McWeeny and Kutzelnigg[585] show that for the expectation value of s(1)s(2), the relevant irreducible part of the two-body density can be expressed in terms of the spinless density matrix of second order:

$$R_0^{(0)}(1,1';2,2') = -\tfrac{1}{3}\Gamma(1,1';2,2') - \tfrac{2}{3}\Gamma(2,1';1,2')$$

$$= -\tfrac{1}{3}\sum_{pqrs}\Gamma^{pq}_{rs}p(1)q(1')r(2)s(2') + 2\Gamma^{pq}_{rs}p(2)q(1')r(1)s(2')$$

$$= -\tfrac{1}{3}\sum_{pqrs}(\Gamma^{pq}_{rs} + 2\Gamma^{rq}_{ps})p(1)q(1')r(2)s(2')$$

with a normalization factor of $\frac{3}{4}$ after spin integration. Hence using this:

$$\langle \mathbf{S}_A\mathbf{S}_B\rangle = \tfrac{3}{4}\delta_{AB}tr(\gamma P_A) + \tfrac{6}{4}tr(P_A(1)P_B(2)R_0^{(0)}(1,1;2,2))$$

And then performing the integral:

$$\langle \mathbf{S}_A\mathbf{S}_B\rangle = \tfrac{3}{4}\delta_{AB}tr(\gamma P_A) - \underbrace{\tfrac{6}{4}\tfrac{1}{3}}_{\frac{1}{2}}\sum_{pqrs}(\Gamma^{pq}_{rs} + 2\Gamma^{rq}_{ps})P^A_{pq}P^B_{rs}$$

This is the final and perhaps most compact equation. The projection operator can be defined in very many different ways. The easiest is to Löwdin orthogonalize the basis set:

$$\left|\mu^A_L\right\rangle = \sum_{\nu^A}\left|\nu^A\right\rangle S^{-1/2}_{\mu\nu}$$

where L denotes the Löwdin basis. This means that molecular orbitals are expressed in the orthogonal basis as:

$$\mathbf{c}_L = \mathbf{S}^{+1/2}\mathbf{c}$$

and the density as:

$$\mathbf{P}_L = \mathbf{S}^{+1/2}\mathbf{P}\mathbf{S}^{+1/2}$$

The fragment projector is defined as:

$$P_A = \sum_{\mu_L\in A}|\mu_L\rangle\langle\mu_L|$$

Clark and Davidson suggest a slightly more elaborate projector in which first, the intra-fragment overlap is eliminated. This happens with a matrix U that for two fragments takes form:

$$\mathbf{U} = \begin{pmatrix}\mathbf{S}_A^{-1/2} & \mathbf{0}\\ \mathbf{0} & \mathbf{S}_B^{-1/2}\end{pmatrix}$$

where is the block of basis functions belonging to fragment A. Likewise:

$$\mathbf{U}^{-1} = \begin{pmatrix}\mathbf{S}_A^{+1/2} & \mathbf{0}\\ \mathbf{0} & \mathbf{S}_B^{+1/2}\\ \end{pmatrix}$$

Then the 'pre-overlap'is:

$$\tilde{\mathbf{S}} = \mathbf{U}^\dagger\mathbf{S}\mathbf{U}$$

This contains the unit matrix in the intra-fragment blocks and non-zero elements elsewhere. This overlap matrix is the finally orthogonalized to obtain the globally orthogonal Löwdin basis. We finally transform the MO coefficients by the following transformation:

$$\mathbf{c}_L = \mathbf{S}^{+1/2}\mathbf{U}^{-1}\mathbf{c}$$

For the projectors, operating with the two MOs i and j gives:

$$\langle i|P_A|j\rangle = \sum_{\mu_L\in A}\sum_{\kappa^B_L\tau^C_L}\left\langle\kappa^B_L|\mu^A_L\right\rangle\left\langle\mu^A_L|\tau^C_L\right\rangle c^L_{\kappa i}c^L_{\tau j}$$

$$= \sum_{\mu_L \in A} \sum_{\kappa_L^B \tau_L^C} \delta_{AB}\delta_{AC}\delta_{\kappa\mu}\delta_{\tau\mu} c_{\kappa i}^L c_{\tau j}^L$$

$$= \sum_{\mu_L \in A} c_{\mu i}^L c_{\mu j}^L$$

Herrmann et al.[586] give the correct expression of the expectation values for a single spin-unrestricted determinant

$$\begin{aligned}\langle \mathbf{S}_A \mathbf{S}_B \rangle = & \tfrac{3}{4}\delta_{AB}\left\{\sum_i P_{ii}^A + \sum_{\bar{i}} P_{\bar{i}\bar{i}}^A\right\} \\ & + \tfrac{1}{4}\left\{\sum_{ij} P_{ii}^A P_{jj}^B + \sum_{\bar{i}\bar{j}} P_{\bar{i}\bar{i}}^A P_{\bar{j}\bar{j}}^B - \sum_{ij} P_{ij}^A P_{ij}^B - \sum_{\bar{i}\bar{j}} P_{\bar{i}\bar{j}}^A P_{\bar{i}\bar{j}}^B - \sum_{\bar{i}j} P_{\bar{i}\bar{i}}^A P_{jj}^B - \sum_{i\bar{j}} P_{ii}^A P_{\bar{j}\bar{j}}^B\right\} \\ & - \sum_{i\bar{j}} P_{i\bar{j}}^A P_{i\bar{j}}^B\end{aligned}$$

Which is used in the Orca implementation.

The use of the Local spin-implementation is very easy. All that is required is to divide the molecule into *fragments*. The rest happens automatically. For example, let us consider two nitrogen atoms at the dissociation limit. While the total spin state is S=0, the tow nitrogen atoms local exist in high-spin states (S=3/2). Consider the following test job:

```
! HF def2-SVP UHF TightSCF PModel

%scf brokensym 3,3 end

* xyz 0 1
N(1) 0 0 0
N(2) 0 0 1094
*
```

and the output:

```
-------------------
LOCAL SPIN ANALYSIS (Loewdin* projector)
-------------------


(1) A.E. Clark; E.R. Davison J. Chem. Phys. (2001), 115(16), pp 7382-7392
(2)  C. Herrmann, M. Reiher, B.A. Hess J. Chem. Phys. (2005) 122, art 034102-1

Number of fragments       = 2
Number of basis functions = 28
Number of atoms           = 2

   ... Fragment AO indices were mapped
   ... intra-fragment orthogonalization completed
   ... Global Loewdin orthogonalizer constructed
   ... Loewdin orthogonalized occupied orbitals constructed


 <SA*SB>        1            2
---------------------------------
   1    :     3.7568
   2    :    -2.2500       3.7568


               <SzA>      Seff(A)
        -------------------------
   1    :     1.5000      1.5017
   2    :    -1.5000      1.5017
```

thus perfectly corresponding to the expectations. The same can be done at the CASSCF level:

```
! HF def2-SVP UHF TightSCF PModel

%casscf nel 6 norb 6 nroots 1 end

* xyz 0 1
N(1) 0 0 0
N(2) 0 0 1094
*
```

With the result:

```
<SA*SB>        1            2
-----------------------------------
   1    :     3.7500
   2    :    -3.7500       3.7500


               <SzA>*      Seff(A)
        --------------------------
   1    :       n.a.      1.5000
   2    :       n.a.      1.5000

* = for a singlet state all <SzA> values are zero by definition
```

Thus, cleanly confirming the expectations. In addition, if nroots > 1, the printing will contain the state-specific analysis of all roots.

As a less trivial example, consider a typical Fe(III) antiferromatically coupled transition metal dimer. An appropriate input may be:

```
! PBE def2-SV(P) TightSCF KDIIS SOSCF PModel

%scf
  brokensym 5,5
end


* xyz -2 1
Fe(1)         -1.93818         0.53739        -0.00010
Fe(2)          1.06735         0.47031         0.00029
S(3)          -0.38935         2.59862        -0.00983
S(3)          -0.48170        -1.59050         0.01091
S(1)           2.68798         0.43924         1.99710
S(1)           2.68692         0.42704        -1.99712
S(2)          -3.55594         0.56407        -1.99889
S(2)          -3.55850         0.58107         1.99646
H(1)           3.91984         0.39462         1.47608
H(1)           3.91940         0.39536        -1.47662
H(2)          -4.78410         0.69179        -1.48280
H(2)          -4.78991         0.49249         1.47983
*
```

Where one of the bridging sulfurs was assigned to each site respectively.

```
 <SA*SB>        1            2            3
-----------------------------------------------
   1    :     7.3346
   2    :    -4.5300       7.3349
   3    :    -0.7229      -0.7230       1.9403


               <SzA>      Seff(A)
```

(continues on next page)