1-electron property integrals

Also available are 1-electron property integrals.

1elPropertyIntegrals	-	1-electron property integrals

Currently the following options are valid:

angular_momentum	-	Angular momentum integrals
dipole	-	Dipole moment integrals
higherMoment	-	Octupole moment integrals
quadrupole	-	Quadrupole moment integrals
velocity	-	Velocity integrals
soc	-	Spin-orbit coupling integrals

Example

```
{
  "1elPropertyIntegrals": ["dipole", "quadrupole"],
  "JSONFormats": ["json"]
}
```

Note

SOC integrals are computed with the settings chosen in the GBW file, except for the relativistic Hamiltonian (see *1-electron relativistic property integrals* for the relativistic version). In the case of SOMF, the SCF density from the density container is used.

1-electron relativistic integrals

1-electron relativistic integrals follow the same notation as the corresponding non-relativistic ones (see *1-electron integrals*).

Example

```
{
  "1elIntegralsRel":["H", "T", "V"],
  "JSONFormats": ["json"]
}
```

This example will produce and store in the corresponding json file the X2C-transformed H-Matrix, as well as the separate potential and kinetic contributions. Note that while the "relativistic" overlap matrix is available for completeness, in X2C it is identical to the non-relativistic overlap by construction.

1-electron relativistic property integrals

Also available are 1-electron relativistic (X2C) property integrals similar to the non-relativistic ones but with reduced options.

1elPropertyRelIntegrals	-	relativistic 1-electron property integrals

Currently the following options are valid:

angular_momentum	-	Angular momentum integrals
dipole	-	Dipole moment integrals
quadrupole	-	Quadrupole moment integrals
soc	-	Spin-orbit coupling integrals

Example

```
{
    "1elPropertyRelIntegrals": ["dipole","quadrupole","angular_momentum"]
}
```

Origin setting

The origin of the electric property is per default the Cartesian origin but also the center of mass and the center of nuclear charges can be selected. Additionally an arbitrary position can be given as x,y,z coordinates when ori_el = 3 is chosen. Currently the following options are valid:

ori_el	-	0 - Cartesian origin
	-	1 - center of mass
	-	2 - center of nuclear charge
	-	3 - arbitrary position
ori_el_xyz	-	position of the origin(x,y,z)

Example

```
{
    "1elPropertyIntegrals": ["dipole","quadrupole"],
    "ori_el": 3,
    "ori_el_xyz": [0.0, 1.0, 1.0]
}
```

Fock matrix contributions

The following two-electron contributions to the AO-basis Fock matrix can be exported:

J	-	Coulomb
K	-	Exchange
VXC	-	DFT XC potential
Vsol	-	Implicit solvent potential
F	-	Total Fock matrix

Note

- All terms are computed using the SCF (spin-)density from the .densities file (named "scfp" and "scfr").
- The same settings and approximations (RI, COSX, grids) as in the original calculation are used.
- All prefactors are already included, such that $F = J + K + VXC + Vsol$.
- For consistency, the output dimensions are always $N_{\text{spin-ops}} \times N_{\text{AO}} \times N_{\text{AO}}$, even though all matrices are symmetric and some terms (J and Vsol) are equal for alpha/beta in the UHF case.
- The 1-electron contributions ("H") are not included and must be requested separately (see *1-electron integrals*, as well as *1-electron relativistic integrals* for the relativistic case).

Example

```
{
  "1elIntegrals": ["H"],
  "FockMatrix": ["F", "J", "K", "VXC", "Vsol"],
  "JSONFormats": ["json"]
}
```

2-electron integrals

ORCA_2json can produce and write on disk three main categories of 2-electron integrals.

1. Two-electron integrals in atomic basis (*2-electron integrals in AO basis*)
2. Two-electron integrals in molecular basis (*2-electron integrals in MO basis*)
3. Two-electron integrals using the resolution of identity approximation (RI). (*RI 2-electron integrals*)

2-electron integrals in AO basis

In atomic basis the two-electron integrals can be saved in Coulomb order or in Exchange order. The keywords for the two options are shown in the next table.

AO_PQRS	-	AO basis integrals in Coulomb order
AO_PRQS	-	AO basis integrals in Exchange order

2-electron integrals in MO basis

In molecular basis, ORCA follows the accepted notation where by I,J,K and L we specify "internal" orbitals, meaning occupied in the reference wavefunction while by A,B,C and D we specify "external" orbitals, meaning orbitals that are empty in the reference wavefunction. With P,Q,R and S we specify all possible orbitals, meaning both "internal" and "external". The available keywords in *"orca_2json"* for the two-electron integrals in the molecular basis are the ones shown in the table below:

MO_IJKL	-	Coulomb 0-external
MO_IJKA	-	Coulomb 1-external
MO_IJAB	-	Coulomb 2-external
MO_IABC	-	Coulomb 3-external
MO_ABCD	-	Coulomb 4-external
MO_PQRS	-	Coulomb ALL integrals
MO_IKJL	-	Exchange 0-external
MO_IKJA	-	Exchange 1-external
MO_IAJB	-	Exchange 2-external
MO_IBAC	-	Exchange 3-external
MO_ACBD	-	Exchange 4-external
MO_PRQS	-	Exchange ALL integrals

Example

```
{
  "2elIntegrals":["MO_IJKL", "MO_ABCD"],
  "Thresh": 1e-8
}
```

RI 2-electron integrals

Using the Resolution of Identity (RI) one can create the integrals in a more efficient way. There are two main categories of RI integrals: the 3-index integrals, where only only half of the transformation has taken place, and the 4-index integrlas where the integrals are totally transformed in the molecular basis. The notation of the integrals follows the one we just described for the two-electron integrals in the molecular basis.

RI_IJV	-	RI 3-index 0-external
RI_IAV	-	RI 3-index 1-external
RI_ABV	-	RI 3-index 2-external
RI_IJKL	-	RI 4-index Coulomb 0-external
RI_IJKA	-	RI 4-index Coulomb 1-external
RI_IJAB	-	RI 4-index Coulomb 2-external
RI_IABC	-	RI 4-index Coulomb 3-external
RI_ABCD	-	RI 4-index Coulomb 4-external
RI_IKJL	-	RI 4-index Exchange 0-external
RI_IKJA	-	RI 4-index Exchange 1-external
RI_IAJB	-	RI 4-index Exchange 2-external
RI_IBAC	-	RI 4-index Exchange 3-external
RI_ACBD	-	RI 4-index Exchange 4-external

In addition to the integrals in case of RI integrals also the used RI Metric is available through the option Vaux:

Vaux	-	true/false

Example

```
{
  "2elIntegrals":["RI_IJKL", "RI_IJV"],
  "Vaux":true
}
```

Full Integral Transformation

The full transformation integrals can be selected via the FullTrafo keyword.

FullTrafo	-	true/false

More 2-electron integrals

Also the non-redundant 2-electron integrals are available for the RI and the nonRI case. Therefore the options 2elNonRedIntegrals or 2elNonRedRIIntegrals must be specified.

2elNonRedIntegrals	-	true/false
2elNonRedRIIntegrals	-	true/false

Orbital Windows, AuxBasisType and Threshold

The orbital window can either be selected automatically by the transformation routine or given by the user via the OrbWin keyword. The internal and external space (i0,i1,a0,a1) is defined via an integer list.

for example:

OrbWin	-	[0,8,9,15]
	-	[0,12,13,30,0,12,13,30]

The default AUX basis type is AuxC but can be changed with the keyword AuxBasisType. Please keep in mind that only those basis types used during the ORCA run can be selected.

AuxBasisType	-	AuxJ
	-	AuxJK
	-	AuxC

To reduce the number of the integrals the keyword Thresh can be used to decrease the selected integrals to save disk space. This effects ONLY the printing and not the accuracy of the generated integrals. The default print threshold is 1.e-15.

Thresh	-	printout threshold (default 1.e-15)

Example

```
{
        "OrbWin": [0,7,8,85,0,0,0,0],
        "Thresh": 1e-8,
        "AuxBasisType": "AuxC",
        "Vaux": true,
        "2elIntegrals": ["RI_IJKL"]
}
```

TDDFT amplitude data (CIS/RPA)

The TDDFT amplitudes and root informations can be requested (no triplet information yet). The available options are:

CIS	-	TDDFT amplitudes
CISNRoots	-	informations of all roots
CISROOT	-	List of roots

Example

```
{
        "CIS": true,
        "CISNRoots": false,
        "CISRoot": [1,4,7,12]
}
```

Example

```
{
        "CIS": true,
        "CISNRoots": true
}
```

JSON Format

The JSON Format that is created can also be defined in the configuration file through the *JSONFormats* variable.

JSONFormats	-	JSON output format

The available JSON formats are:

json	-	ASCII JSON format
bson	-	binary JSON format
ubjson	-	universal binary json format
msgpck	-	MessagePack format

Example

```
{
  "JSONFormats":["json","bson","ubjson","msgpck"]
}
```

MO Coefficients

MOCoefficients	-	molecule orbital information (true\false)

Example

```
{
  "MOCoefficients": true
}
```

9.3.4 Import JSON data into ORCA

Some information like geometry, basis set and Molecular orbitals stored in the json format written by orca_2json can be used to create a new gbw-file to be specified as an orbital file in the orca input. The program is called as a standalone via command line.

```
orca_2json BaseName.json -gbw
```

Basic Information

In order to create a functional gbw-file the following information must be provided:

per Atom	-	Basis
	-	Coords
	-	ElementNumber
	-	NuclearCharge
per molecule	-	Charge
	-	CoordinateUnits
	-	HFTyp
	-	Multiplicity
	-	MolecularOrbitals

NOTE

Please keep in mind that *MolecularOrbitals* is a composite of 2 different components, namely *"EnergyUnit"* and *"MOs"*. Then *"MOs"* contains *"MOCoefficients"*, *"Occupancy"*, *"OrbitalEnergy"*, "OrbitalSymLabel"* and "OrbitalSymmetry"*.

Example

The following file (let's call it *filename.json*) is a json file for H_2 molecule with STO-3G basis set.

```
{
  "Molecule": {
    "Atoms": [
      {"Basis": [
          {"Coefficients": [ 0.1543289707029839,
              0.5353281424384732,0.44463454202535485],
            "Exponents": [3.42525091,0.62391373,0.1688554],
            "Shell": "s"}],
        "Coords": [0.0,0.0,0.0],
        "ElementNumber": 1,
        "NuclearCharge": 1.0},
      {"Basis": [
          {"Coefficients": [0.1543289707029839,
              0.5353281424384732,0.44463454202535485],
            "Exponents": [3.42525091,0.62391373,0.1688554],
            "Shell": "s"}],
        "Coords": [0.0,0.0,0.8],
        "ElementNumber": 1,
        "NuclearCharge": 1.0}],
    "Charge": 0,
    "CoordinateUnits": "Angs",
    "HFTyp": "RHF",
    "MolecularOrbitals": {
      "EnergyUnit": "Eh",
      "MOs": [{"MOCoefficients": [
            -0.5554171364661243,-0.5554171364661241],
          "Occupancy": 2.0,
```

(continues on next page)

(continued from previous page)

```
          "OrbitalEnergy": -0.5544958795933514,
          "OrbitalSymLabel": "A",
          "OrbitalSymmetry": 0},
        {"MOCoefficients": [
           -1.1482994800696493,1.1482994800696493],
          "Occupancy": 0.0,
          "OrbitalEnergy": 0.6126180830925017,
          "OrbitalSymLabel": "A",
          "OrbitalSymmetry": 0}]},
    "Multiplicity": 1}
}
```

running then the command

```
orca_2json filename.json -gbw
```

should create a gbw file that ORCA can read.

Definition of the real solid harmonic Gaussian orbitals

When integrals over real solid harmonic Gaussian orbitals are issued into a JSON file, the precise definition of these orbitals becomes important. ORCA uses its own peculiar conventions for the arrangement and the phases of the individual components of the orbital shells.

The definitions of the angular parts of all shell components up to angular momentum l ($\ell = 8$) are documented below. These correspond to the real solid harmonics $S_{\ell,m}(x,y,z)$ that are normalized the following way:

$$\int_0^\pi \sin\vartheta \, d\vartheta \int_{-\pi}^{\pi} d\varphi \, r^{-2\ell} S^2_{\ell,m}(x,y,z) = 1$$

$R_\ell(r)$ is the common radial part of a shell with angular momentum ℓ that consists of a specific basis set dependent linear combination of Gaussian primitives. It is normalized independently:

$$\int_0^\infty r^{2\ell+2} R^2_\ell(r) \, dr = 1$$

The factors r^2 and $\sin\vartheta$ in these integrals arise from the volume element in spherical polar coordinates:

$$\begin{aligned} x &= r\sin\vartheta\cos\varphi \\ y &= r\sin\vartheta\sin\varphi \\ z &= r\cos\vartheta \\ dx\,dy\,dz &= r^2\sin\vartheta\,dr\,d\vartheta\,d\varphi \end{aligned}$$

Angular momentum s ($l = 0$)

$$s = \frac{1}{2\sqrt{\pi}} R_s(r)$$

Angular momentum p ($l = 1$)

$$
\begin{aligned}
N_p &= \frac{1}{2}\sqrt{\frac{3}{\pi}} \\
p^{(0)} = p_0 \quad &= N_p\, z\, R_p(r) \\
p^{(1)} = p_{+1} &= N_p\, x\, R_p(r) \\
p^{(2)} = p_{-1} &= N_p\, y\, R_p(r)
\end{aligned}
$$

Angular momentum d ($l = 2$)

$$
\begin{aligned}
N_d &= \frac{1}{2}\sqrt{\frac{15}{\pi}} \\
d^{(0)} = d_0 \quad &= \frac{\sqrt{3}}{6}\, N_d \left(3z^2 - r^2\right) R_d(r) \\
d^{(1)} = d_{+1} &= N_d\, xz\, R_d(r) \\
d^{(2)} = d_{-1} &= N_d\, yz\, R_d(r) \\
d^{(3)} = d_{+2} &= N_d\, \frac{x^2 - y^2}{2}\, R_d(r) \\
d^{(4)} = d_{-2} &= N_d\, xy\, R_d(r)
\end{aligned}
$$

Angular momentum f ($l = 3$)

$$
\begin{aligned}
N_f &= \frac{1}{2}\sqrt{\frac{105}{\pi}} \\
f^{(0)} = f_0 \quad &= \frac{\sqrt{15}}{30}\, N_f\, z \left(5z^2 - 3r^2\right) R_f(r) \\
f^{(1)} = f_{+1} &= \frac{\sqrt{10}}{20}\, N_f\, x \left(5z^2 - r^2\right) R_f(r) \\
f^{(2)} = f_{-1} &= \frac{\sqrt{10}}{20}\, N_f\, y \left(5z^2 - r^2\right) R_f(r) \\
f^{(3)} = f_{+2} &= \frac{1}{2}\, N_f \left(x^2 - y^2\right) z\, R_f(r) \\
f^{(4)} = f_{-2} &= N_f\, xyz\, R_f(r) \\
f^{(5)} = f_{+3} &= -\frac{\sqrt{6}}{12}\, N_f\, x \left(x^2 - 3y^2\right) R_f(r) \\
f^{(6)} = f_{-3} &= -\frac{\sqrt{6}}{12}\, N_f\, y \left(3x^2 - y^2\right) R_f(r)
\end{aligned}
$$

Angular momentum g ($l = 4$)

$$
\begin{aligned}
N_g &= \frac{3}{2}\sqrt{\frac{35}{\pi}} \\
g^{(0)} = g_0 \quad &= \frac{\sqrt{35}}{280} N_g \left(35z^4 - 30z^2r^2 + 3r^4\right) R_g(r) \\
g^{(1)} = g_{+1} &= \frac{\sqrt{14}}{28} N_g \, xz \left(7z^2 - 3r^2\right) R_g(r) \\
g^{(2)} = g_{-1} &= \frac{\sqrt{14}}{28} N_g \, yz \left(7z^2 - 3r^2\right) R_g(r) \\
g^{(3)} = g_{+2} &= \frac{\sqrt{7}}{28} N_g \left(x^2 - y^2\right) \left(7z^2 - r^2\right) R_g(r) \\
g^{(4)} = g_{-2} &= \frac{\sqrt{7}}{14} N_g \, xy \left(7z^2 - r^2\right) R_g(r) \\
g^{(5)} = g_{+3} &= -\frac{\sqrt{2}}{4} N_g \, x \left(x^2 - 3y^2\right) z \, R_g(r) \\
g^{(6)} = g_{-3} &= -\frac{\sqrt{2}}{4} N_g \, y \left(3x^2 - y^2\right) z \, R_g(r) \\
g^{(7)} = g_{+4} &= -\frac{1}{8} N_g \left(x^4 - 6x^2y^2 + y^4\right) R_g(r) \\
g^{(8)} = g_{-4} &= -\frac{1}{2} N_g \, xy \left(x^2 - y^2\right) R_g(r)
\end{aligned}
$$

Angular momentum h ($l = 5$)

$$
\begin{aligned}
N_h &= \frac{1}{2}\sqrt{\frac{11}{\pi}} \\
h^{(0)} = h_0 \quad &= \frac{1}{8} N_h \, z \left(63z^4 - 70z^2r^2 + 15r^4\right) R_h(r) \\
h^{(1)} = h_{+1} &= \frac{\sqrt{15}}{8} N_h \, x \left(21z^4 - 14z^2r^2 + r^4\right) R_h(r) \\
h^{(2)} = h_{-1} &= \frac{\sqrt{15}}{8} N_h \, y \left(21z^4 - 14z^2r^2 + r^4\right) R_h(r) \\
h^{(3)} = h_{+2} &= \frac{\sqrt{105}}{4} N_h \left(x^2 - y^2\right) z \left(3z^2 - r^2\right) R_h(r) \\
h^{(4)} = h_{-2} &= \frac{\sqrt{105}}{2} N_h \, xyz \left(3z^2 - r^2\right) R_h(r) \\
h^{(5)} = h_{+3} &= -\frac{\sqrt{70}}{16} N_h \, x \left(x^2 - 3y^2\right) \left(9z^2 - r^2\right) R_h(r) \\
h^{(6)} = h_{-3} &= -\frac{\sqrt{70}}{16} N_h \, y \left(3x^2 - y^2\right) \left(9z^2 - r^2\right) R_h(r) \\
h^{(7)} = h_{+4} &= -\frac{3\sqrt{35}}{8} N_h \left(x^4 - 6x^2y^2 + y^4\right) z \, R_h(r) \\
h^{(8)} = h_{-4} &= -\frac{3\sqrt{35}}{2} N_h \, xy \left(x^2 - y^2\right) z \, R_h(r) \\
h^{(9)} = h_{+5} &= \frac{3\sqrt{14}}{16} N_h \, x \left(x^4 - 10x^2y^2 + 5y^4\right) R_h(r) \\
h^{(10)} = h_{-5} &= \frac{3\sqrt{14}}{16} N_h \, y \left(5x^4 - 10x^2y^2 + y^4\right) R_h(r)
\end{aligned}
$$

Angular momentum i ($l = 6$)

$$
\begin{aligned}
N_i &= \frac{1}{2}\sqrt{\frac{13}{\pi}} \\
i^{(0)} = i_0 \quad &= \frac{1}{16} N_i \left(231z^6 - 315z^4r^2 + 105z^2r^4 - 5r^6\right) R_i(r) \\
i^{(1)} = i_{+1} &= \frac{\sqrt{21}}{8} N_i\, xz \left(33z^4 - 30z^2r^2 + 5r^4\right) R_i(r) \\
i^{(2)} = i_{-1} &= \frac{\sqrt{21}}{8} N_i\, yz \left(33z^4 - 30z^2r^2 + 5r^4\right) R_i(r) \\
i^{(3)} = i_{+2} &= \frac{\sqrt{210}}{32} N_i \left(x^2 - y^2\right) \left(33z^4 - 18z^2r^2 + r^4\right) R_i(r) \\
i^{(4)} = i_{-2} &= \frac{\sqrt{210}}{16} N_i\, xy \left(33z^4 - 18z^2r^2 + r^4\right) R_i(r) \\
i^{(5)} = i_{+3} &= -\frac{\sqrt{210}}{16} N_i\, x \left(x^2 - 3y^2\right) z \left(11z^2 - 3r^2\right) R_i(r) \\
i^{(6)} = i_{-3} &= -\frac{\sqrt{210}}{16} N_i\, y \left(3x^2 - y^2\right) z \left(11z^2 - 3r^2\right) R_i(r) \\
i^{(7)} = i_{+4} &= -\frac{3\sqrt{7}}{16} N_i \left(x^4 - 6x^2y^2 + y^4\right) \left(11z^2 - r^2\right) R_i(r) \\
i^{(8)} = i_{-4} &= -\frac{3\sqrt{7}}{4} N_i\, xy \left(x^2 - y^2\right) \left(11z^2 - r^2\right) R_i(r) \\
i^{(9)} = i_{+5} &= \frac{3\sqrt{154}}{16} N_i\, x \left(x^4 - 10x^2y^2 + 5y^4\right) z\, R_i(r) \\
i^{(10)} = i_{-5} &= \frac{3\sqrt{154}}{16} N_i\, y \left(5x^4 - 10x^2y^2 + y^4\right) z\, R_i(r) \\
i^{(11)} = i_{+6} &= \frac{\sqrt{462}}{32} N_i \left(x^2 - y^2\right) \left(x^4 - 14x^2y^2 + y^4\right) R_i(r) \\
i^{(12)} = i_{-6} &= \frac{\sqrt{462}}{16} N_i\, xy \left(3x^4 - 10x^2y^2 + 3y^4\right) R_i(r)
\end{aligned}
$$

Angular momentum k ($l = 7$)

Note that in ORCA 6.0 and earlier, basis functions of angular momentum quantum number 7 had the label j, in contrast to the usual convention in spectroscopy where the letter j is skipped. This inconsistency has been corrected

in the current release of ORCA.

$$
\begin{aligned}
N_k &= \frac{1}{2}\sqrt{\frac{15}{\pi}} \\
k^{(0)} = k_0 \quad &= \frac{1}{16}\, N_k\, z \left(429z^6 - 693z^4r^2 + 315z^2r^4 - 35r^6\right) R_k(r) \\
k^{(1)} = k_{+1} &= \frac{\sqrt{7}}{32}\, N_k\, x \left(429z^6 - 495z^4r^2 + 135z^2r^4 - 5r^6\right) R_k(r) \\
k^{(2)} = k_{-1} &= \frac{\sqrt{7}}{32}\, N_k\, y \left(429z^6 - 495z^4r^2 + 135z^2r^4 - 5r^6\right) R_k(r) \\
k^{(3)} = k_{+2} &= \frac{\sqrt{42}}{32}\, N_k \left(x^2 - y^2\right) z \left(143z^4 - 110z^2r^2 + 15r^4\right) R_k(r) \\
k^{(4)} = k_{-2} &= \frac{\sqrt{42}}{16}\, N_k\, xyz \left(143z^4 - 110z^2r^2 + 15r^4\right) R_k(r) \\
k^{(5)} = k_{+3} &= -\frac{\sqrt{21}}{32}\, N_k\, x \left(x^2 - 3y^2\right) \left(143z^4 - 66z^2r^2 + 3r^4\right) R_k(r) \\
k^{(6)} = k_{-3} &= -\frac{\sqrt{21}}{32}\, N_k\, y \left(3x^2 - y^2\right) \left(143z^4 - 66z^2r^2 + 3r^4\right) R_k(r) \\
k^{(7)} = k_{+4} &= -\frac{\sqrt{231}}{16}\, N_k \left(x^4 - 6x^2y^2 + y^4\right) z \left(13z^2 - 3r^2\right) R_k(r) \\
k^{(8)} = k_{-4} &= -\frac{\sqrt{231}}{4}\, N_k\, xy \left(x^2 - y^2\right) z \left(13z^2 - 3r^2\right) R_k(r) \\
k^{(9)} = k_{+5} &= \frac{\sqrt{231}}{32}\, N_k\, x \left(x^4 - 10x^2y^2 + 5y^4\right) \left(13z^2 - r^2\right) R_k(r) \\
k^{(10)} = k_{-5} &= \frac{\sqrt{231}}{32}\, N_k\, y \left(5x^4 - 10x^2y^2 + y^4\right) \left(13z^2 - r^2\right) R_k(r) \\
k^{(11)} = k_{+6} &= \frac{\sqrt{6006}}{32}\, N_k \left(x^2 - y^2\right) \left(x^4 - 14x^2y^2 + y^4\right) z\, R_k(r) \\
k^{(12)} = k_{-6} &= \frac{\sqrt{6006}}{16}\, N_k\, xy \left(3x^2 - y^2\right) \left(x^2 - 3y^2\right) z\, R_k(r) \\
k^{(13)} = k_{+7} &= -\frac{\sqrt{429}}{32}\, N_k\, x \left(x^6 - 21x^4y^2 + 35x^2y^4 - 7y^6\right) R_k(r) \\
k^{(14)} = k_{-7} &= -\frac{\sqrt{429}}{32}\, N_k\, y \left(7x^6 - 35x^4y^2 + 21x^2y^4 - y^6\right) R_k(r)
\end{aligned}
$$

Angular momentum l ($l = 8$)

Note that in ORCA 6.0 and earlier, basis functions of angular momentum quantum number 8 were never actually supported. The likely result when attempting to use them was a crash.

Also note that the symbol l is reserved in basis set files for sp shells that have a common set of Gaussian exponents.

Thus for angular momentum 8 and higher, the numerical value needs to be used instead of the letter.

$$
\begin{aligned}
N_l &= \frac{1}{2}\sqrt{\frac{17}{\pi}} \\
l^{(0)} = l_0 \;\; &= \frac{1}{128}\, N_l \left(6435z^8 - 12012z^6r^2 + 6930z^4r^4 - 1260z^2r^6 + 35r^8\right) R_l(r) \\
l^{(1)} = l_{+1} &= \frac{3}{32}\, N_l\, xz \left(715z^6 - 1001z^4r^2 + 385z^2r^4 - 35r^6\right) R_l(r) \\
l^{(2)} = l_{-1} &= \frac{3}{32}\, N_l\, yz \left(715z^6 - 1001z^4r^2 + 385z^2r^4 - 35r^6\right) R_l(r) \\
l^{(3)} = l_{+2} &= \frac{3\sqrt{70}}{64}\, N_l \left(x^2 - y^2\right) \left(143z^6 - 143z^4r^2 + 33z^2r^4 - r^6\right) R_l(r) \\
l^{(4)} = l_{-2} &= \frac{3\sqrt{70}}{32}\, N_l\, xy \left(143z^6 - 143z^4r^2 + 33z^2r^4 - r^6\right) R_l(r) \\
l^{(5)} = l_{+3} &= -\frac{\sqrt{1155}}{32}\, N_l\, x \left(x^2 - 3y^2\right) z \left(39z^4 - 26z^2r^2 + 3r^4\right) R_l(r) \\
l^{(6)} = l_{-3} &= -\frac{\sqrt{1155}}{32}\, N_l\, y \left(3x^2 - y^2\right) z \left(39z^4 - 26z^2r^2 + 3r^4\right) R_l(r) \\
l^{(7)} = l_{+4} &= -\frac{3\sqrt{77}}{64}\, N_l\, \left(x^4 - 6x^2y^2 + y^4\right) \left(65z^4 - 26z^2r^2 + r^4\right) R_l(r) \\
l^{(8)} = l_{-4} &= -\frac{3\sqrt{77}}{16}\, N_l\, xy \left(x^2 - y^2\right) \left(65z^4 - 26z^2r^2 + r^4\right) R_l(r) \\
l^{(9)} = l_{+5} &= \frac{3\sqrt{1001}}{32}\, N_l\, x \left(x^4 - 10x^2y^2 + 5y^4\right) z \left(5z^2 - r^2\right) R_l(r) \\
l^{(10)} = l_{-5} &= \frac{3\sqrt{1001}}{32}\, N_l\, y \left(5x^4 - 10x^2y^2 + y^4\right) z \left(5z^2 - r^2\right) R_l(r) \\
l^{(11)} = l_{+6} &= \frac{\sqrt{858}}{64}\, N_l \left(x^2 - y^2\right) \left(x^4 - 14x^2y^2 + y^4\right) \left(15z^2 - r^2\right) R_l(r) \\
l^{(12)} = l_{-6} &= \frac{\sqrt{858}}{32}\, N_l\, xy \left(x^2 - 3y^2\right) \left(3x^2 - y^2\right) \left(15z^2 - r^2\right) R_l(r) \\
l^{(13)} = l_{+7} &= -\frac{3\sqrt{715}}{32}\, N_l\, x \left(x^6 - 21x^4y^2 + 35x^2y^4 - 7y^6\right) z\, R_l(r) \\
l^{(14)} = l_{-7} &= -\frac{3\sqrt{715}}{32}\, N_l\, y \left(7x^6 - 35x^4y^2 + 21x^2y^4 - y^6\right) z\, R_l(r) \\
l^{(15)} = l_{+8} &= -\frac{3\sqrt{715}}{128}\, N_l \left(x^8 - 28x^6y^2 + 70x^4y^4 - 28x^2y^6 + y^8\right) R_l(r) \\
l^{(16)} = l_{-8} &= -\frac{3\sqrt{715}}{16}\, N_l\, xy \left(x^2 - y^2\right) \left(x^4 - 6x^2y^2 + y^4\right) R_l(r)
\end{aligned}
$$

9.4 Property File

One of the files that ORCA produces, during a calculation, is the *property file*. The name of the file is `basename.property.txt`, where `basename` is the basename of the input file.

The property file has mainly two usages in ORCA. The first usage is to work as basis for the *Compound* scripting language. *Compound* reads all its information concerning properties through the property file and not through parsing of the ORCA output. The second usage of the property file is to make it easier for other programs, or potential GUIs, to create interfaces with ORCA.

The advantage of the property file compared to the normal ORCA output is that it's syntax is well defined and so it is easier to parse it. It can also be converted to JSON format as shown below.

Writing to the property file is enabled by default for most calculations. It can be disabled via `!NoPropFile` or toggled on/off in the `%output` block:

```
%output
  PropFile false  # default: true
end
```

Caution

Since the whole file is read and re-written multiple times for each single point calculation, optimizations of systems with a few thousand atoms can incur a few milliseconds of stacking I/O overhead, per optimization cycle (i.e., a few seconds per cycle after several hundred steps). Thus, it is recommended to disable the property file for such calculations and this is done by default for `GOAT`, `MD-L-Opt`, `MM`, or `GFN-FF` chosen in the simple input.

9.4.1 TXT Format

After any ORCA calculation a property file is created with the extension `.property.txt`. The file is a text file and one can read it and edit it with any available text editor. Below we will analyze the syntax of the file.

The file always starts with the following three lines:

```
***************************************************
****************** ORCA 6.1.x *********************
***************************************************
```

where, obviously the version of ORCA changes (it is also stored in `Calculation_Status -> version`).

Then, the file consists of a list of properties. Each property starts with the symbol `$` followed by the name of the property and ends with the symbol `$` followed by `End`.

For example:

```
$SCF_Nuc_Gradient
   &GeometryIndex 1
   &NAtoms [&Type "Integer"] 2
   &gradNorm [&Type "Double"]       7.6940943132805237e-02
   &grad [&Type "ArrayOfDoubles", &Dim (6,1)]
                                                        0

0                                     6.7009699964043727e-10
1                                    -1.3755030800675890e-10
2                                     5.4405462640094826e-02
3                                    -6.7009699282161039e-10
4                                     1.3755031156980307e-10
5                                    -5.4405462640095381e-02
   &Method [&Type "String"] "SCF"
   &Level [&Type "String"] "Relaxed density"
   &Mult [&Type "Integer"] 1
   &State [&Type "Integer"] 0
   &Irrep [&Type "Integer"] 0
$End
```

Each property consists of *components*. Each component starts with the symbol `&` and has no ending symbol.

For example:

```
   &gradNorm [&Type "Double"]       7.6940943132805237e-02
```

The first component in every property is `GeometryIndex`, which shows the geometry to which the current property belongs. For example:

```
&GeometryIndex 1
```

Note that geometry indices start from 1.

The remaining components have the following syntax:

- First the start of component symbol `&`.
- Then follows the name of the component.
- Then a bracket opens with various bracket information about the component. For details on the syntax of the bracket information please check *Bracket Information*.

After the bracket there are different options.

If the type is `String` then a string is expected starting with quotation marks. For example:

```
&Method [&Type "String"] "SCF"
```

If the type is `Double` or `Integer` then a number of the appropriate type is expected. For example:

```
&NAtoms [&Type "Integer"] 2 "Number of atoms"
```

As in the example above, a component description or comment in quotation marks may be optionally present after the value in the same line as the component name. Note that for `Array` components, the comment actually comes *before* the data.

Finally, if the type is a kind of `Array` then, after the optional comment, an array is written starting from the next line. Arrays are written in blocks of up to 8 whitespace-sparated columns. Each block starts with a line with column indices (starting from 0). The second line in a block is reserved but currently ignored. The next lines start with the row index and continue with the data for the respective row/columns. For example:

```
   &dipoleTotal [&Type "ArrayOfDoubles", &Dim (3,1)] "Total"
                                                        0

0                                      -5.1716015643064861e-01
1                                       5.1774970936823518e-02
2                                       7.8324050140722279e-01
```

Note that `Complex` scalars are printed as two consecutive values for the real and imaginary part (before the optional comment), whereas for `ArrayOfComplex` the real part of the whole array is printed first, followed by the imaginary part.

Bracket Information

Bracket information is a list of entries separated with ‘,’. The first and most important bracket entry is the `Type`, which can be one of the following:

- `Double`
- `Integer`
- `Boolean`
- `Complex`
- `String`
- `ArrayOfDoubles`
- `ArrayOfIntegers`
- `ArrayOfBooleans`
- `ArrayOfComplex`
- `Coordinates`

For example:

```
&NAtoms [&Type "Integer"] 2
```

Then, in case the `Type` is a kind of array, the bracket must contain the dimensions of the array, using the `Dim` entry. Note that all arrays are 2-dimensional! For example:

```
&ATNO [&Type "ArrayOfIntegers", &Dim (2,1)]
```

Optionally, `Units` may also be given, if the value is not in a.u. For example:

```
&dipoleMagnitude [&Type "Double", &Units "Debye"]       9.4000050960598935e-01
```

9.4.2 JSON Format

The property file can be also produced in a JSON format. Internally this happens through transformation of the `txt` format to JSON format. There are two ways to create a JSON property file.

The first way is through the normal ORCA input using the `JSONPropFile` keyword.

```
%Output
  JSONPropFile True
End
```

this will create a `basename.property.txt` and in addition a `basename.property.json` file.

The second way is through the *`orca_2json` command*. For this, one first has to run a normal ORCA input, that will create a `basename.property.txt` file, and then convert it to `basename.property.json` using the command:

```
orca_2json basename -property
```

The file is structured as follows:

```
{
  "<GlobalPropertyName>": {
    "<ComponentName>": <value>,
    <More components>
  },
  <More global properties>
  "Geometries": [
    {
      "Geometry": {
        "Coordinates": {
            "Cartesians": [
                [
                    "<Symbol>",
                    <X>,
                    <Y>,
                    <Z>
                ],
                <More atoms>
            ],
            "Type": "Cartesians",
            "Units": "a.u."
        },
        "NAtoms": <value>,
        <More components>
      },
      "<PropertyName>": {
        "<ComponentName>": <value>,
        <More components>
```

(continues on next page)

(continued from previous page)

```
      },
      <More properties for this geometry>
    },
    <More geometries>
  ]
}
```

The "global", i.e. geometry-independent properties are: `Calculation_Status`, `Calculation_Info`, and `Calculation_Timings`.

Most component data types are directly converted to the corresponding JSON types. All `Array` components are stored as two-dimensional using nested lists (even if one dimension is of length 1). `Complex` and `ArrayOfComplex` values are stored as separate real and imaginary parts:

```
"ComplexComponentName": {
  "re": <real part>,
  "im": <imaginary part>
},
"ComplexArrayComponentName": {
  "re": [
      [<r11>, <r12>, <...>],
      [<r21>, <r22>, <...>],
      <...>
    ],
  "im": [
      [<i11>, <i12>, <...>],
      [<i21>, <i22>, <...>],
      <...>
    ]
}
```

9.4.3 List of Available Properties

Table 9.1: List of available properties

Property[1]	M[2]	Component	T[3]	O[4]	L[5]	Description	Units
`AutoCI_Energies` [`Energy`]	Y	`numOfEl`	I	N	N		
		`numOfCorrEl`	I	N	N		
		`numOfAlphaCorrEl`	I	Y	N		
		`numOfBetaCorrEl`	I	Y	N		
		`refEnergy`	AD	N	N	Reference energy for each state	
		`corrEnergy`	AD	N	N	Total correlation energy for each state	
		`Method`	S	N	N		
		`totalEnergy`	AD	N	N	Total energy of each state	
		`Mult`	AI	Y	N	Multiplicity of each state	
		`Irrep`	AI	Y	N	Irreducible representation of each state	
		`RelCorr`	S	Y	N	Relativistic correction (SOC and/or SSC)	
		`NBlocks`	I	Y	N	Number of multiplicity blocks	
		`NRoots`	AI	Y	N	Number of roots in each block	
		`NTotalRoots`	I	Y	N	Total number of roots	
		`Block`	AI	Y	N	Block index of each state	
		`Root`	AI	Y	N	Root index within the block	
		`FollowIRoot`	I	Y	N	Index of the followed root	

continues on next page

Table 9.1 – continued from previous page

		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
AutoCI_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
EFG_Tensor	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
AutoCI_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
AutoCI_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
AutoCI_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	

continues on next page

Table 9.1 – continued from previous page

		Isotope	D	N	Y	Atomic mass
		I	D	N	Y	Spin of the nuclei
		PFAC	D	N	Y	Prefactor PFAC=g*e*g*N*be*bN (in MHz)
		ARaw	AD	N	Y	Raw tensor
		AEigenvalues	AD	N	Y	Eigenvalues
		orientation	AD	N	Y	Eigenvectors
		AIso	D	N	Y	
AUTOCI_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
		NUC	I	N	Y	Index of the nuclei
		Elems	I	N	Y	Atomic number of the nuclei
		SDSO	AD	N	Y	Diamagnetic contribution
		SPSO	AD	N	Y	Paramagnetic contribution
		STot	AD	N	Y	Total tensor
		orientation	AD	N	Y	Eigenvectors
		sTotEigen	AD	N	Y	Eigenvalues
		siso	D	N	Y	
		saniso	D	N	Y	
AutoCI_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
AutoCI_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
AutoCI_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.

continues on next page