Approximate relativistic methods

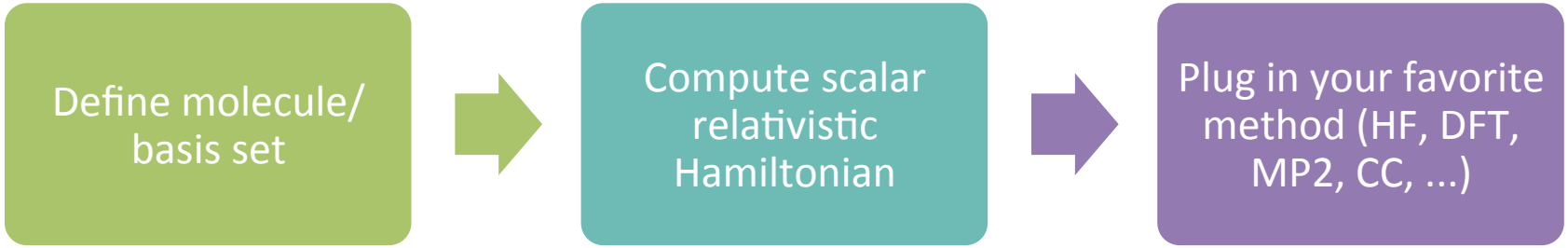
Psi4 Developers' Conference Nov 2104

Prakash Verma and
Francesco Evangelista
Emory University

Approximate relativistic methods

PROS:

- Approximate methods to include relativistic effects that avoid to use 2- and 4-componend formalisms.
- Many variants (ZORA, IORA, DKH, NESC, X2C).
- They are very efficient.
- Spin-free one-electron versions only modify the AO basis one-electron integrals.



CONS:

- Need to use appropriate basis set (e.g. cc-pVDZ ⇾ cc-pVDZ-DK).
- Spin-orbit effects are missing.

Dirac equation

One-particle Dirac equation

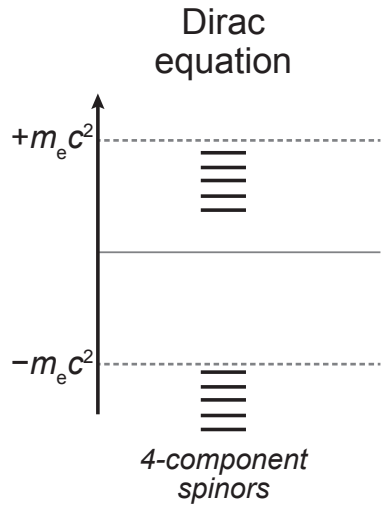
$$\hat{D}\begin{pmatrix}\phi^L\\ \phi^S\end{pmatrix} = E\begin{pmatrix}\phi^L\\ \phi^S\end{pmatrix}$$

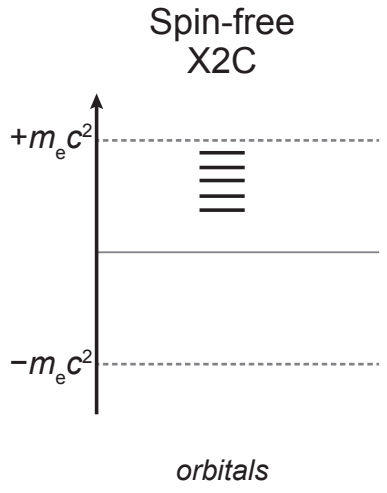
$$\hat{D} = \begin{pmatrix}\hat{V} & c\,(\vec{\sigma}.\,\vec{p})\\ c\,(\vec{\sigma}.\,\vec{p}) & \hat{V} - 2mc^2\end{pmatrix}$$

Exact two-component method (X2C)

$$\hat{U}^\dagger \hat{D}\hat{U} = \hat{U}^\dagger \begin{bmatrix} h_{\mathrm{LL}} & h_{\mathrm{LS}} \\ h_{\mathrm{SL}} & h_{\mathrm{SS}} \end{bmatrix} \hat{U} = \begin{bmatrix} h_{++}^{\mathrm{FW}} & 0 \\ 0 & h_{--}^{\mathrm{FW}} \end{bmatrix}$$

$$\hat{U}^\dagger \begin{pmatrix}\phi_+^L\\ \phi_+^S\end{pmatrix} = \begin{pmatrix}\phi^{FW}\\ 0\end{pmatrix}$$





Computational Aspects

Solve the modified Dirac equation in a kinetically-balanced basis set

$$\phi_i^S = \sum_\mu C_{\mu i}^S \frac{\vec{\sigma}\cdot\vec{p}}{2mc}\chi_\mu$$

$$\begin{pmatrix} V & T \\ T & \frac{W}{4m^2c^2} - T \end{pmatrix}\begin{pmatrix} C^L \\ C^S \end{pmatrix} = \begin{pmatrix} S & 0 \\ 0 & \frac{T}{2mc^2} \end{pmatrix}\begin{pmatrix} C^L \\ C^S \end{pmatrix}\epsilon$$

Need the following integrals

$$S_{\mu\nu} = \langle\chi_\mu|\chi_\nu\rangle$$

$$T_{\mu\nu} = \langle\chi_\mu|\frac{-\nabla^2}{2}|\chi_\nu\rangle$$

$$V_{\mu\nu} = \langle\chi_\mu|\sum_A \frac{Z_A}{|\vec{r} - \vec{R_A}|}|\chi_\nu\rangle$$

$$W_{\mu\nu} = \langle\chi_\mu|(\vec{\sigma}\cdot\vec{p})V(\vec{\sigma}\cdot\vec{p})|\chi_\nu\rangle$$

Spin-free version

$$W_{\mu\nu} = W_{\mu\nu}^{\mathrm{SF}} + W_{\mu\nu}^{\mathrm{SO}}$$

$$W_{\mu\nu}^{\mathrm{SF}} = \langle\chi_\mu|\vec{p}\cdot(V\vec{p})|\chi_\nu\rangle$$

Implementation

The class RelPotentialInt computes the W integrals (no integral derivatives yet)

$$W_{\mu\nu}^{\mathrm{SF}} = \langle \chi_\mu | \vec{p} \cdot (V\vec{p}) | \chi_\nu \rangle = \langle \vec{p}\chi_\mu | V | \vec{p}\chi_\nu \rangle$$

```
/*! \ingroup MINTS
 * \class RelPotentialInt
 * \brief Computes relativistic potential integrals.
 * Use an IntegralFactory to create this object. */
class RelPotentialInt : public OneBodyAOInt
{
    /// Computes integrals between two shell objects.
    void compute_pair(const GaussianShell&, const GaussianShell&);
    /// Computes integrals between two shell objects.
```

Implementation

X2C integrals are computed by the MintsHelper class

```
void MintsHelper::integrals()
{
...
     // Compute and dump one-electron SO integrals.
     if (options_.get_str("RELATIVISTIC") == "NO"){
          // Overlap
          so_overlap()->save(psio_, PSIF_OEI);
          // Kinetic
          so_kinetic()->save(psio_, PSIF_OEI);
          // Potential
          so_potential()->save(psio_, PSIF_OEI);
     }else if (options_.get_str("RELATIVISTIC") == "X2C"){
          outfile->Printf( " Using relativistic (X2C) overlap, kinetic,
and potential integrals.\n");
     }
```

(there is another call, perhaps redundant, in `one_electron_integrals()`)

Calling X2C

Calling X2C

```
set {
  basis cc-pVDZ-DK
  relativistic x2c
}

energy('x2c')
energy(’ccsd(t)')
```