Localized orbital schemes: Wannier, IAO, …

Wannier functions: minimize the localization functional (with pbc)

Intrinsic atomic orbitals (IAO): find molecular polarized AOs (with pbc or obc)

Other methods: Foster-Boys, Fourth moment, Edmiston-Ruedenberg, …
see https://www.wikiwand.com/en/Localized_molecular_orbitals

Intrinsic Atomic Orbitals

We start from identifying the occupied molecular orbitals of a given wave function, parameterized by the big basis B_1

Occupied MOs $|i\rangle = \sum_{\mu} |\mu\rangle C_i^{\mu}$ (usually delocalized, polarized by the molecular environment)

It is often hard to associate them with an atom.

It would be easy to interpret if we can expand $|i\rangle$ in terms of some small free atom basis, but it would be inaccurate, since free-atom AOs contain no polarization due to the molecular environment.

We want to find some $\{|\rho\rangle\} \subset B_1$, and they are associate with the atoms, such that

$$|i\rangle = \sum_{\rho} |\rho\rangle C_i'^{\rho}$$

G. Knizia, JCTC, 9 4834 (2013)

Intrinsic Atomic Orbitals

We start from identifying the occupied molecular orbitals of a given wave function, parameterized by the big basis B_1

Occupied MOs $|i\rangle = \sum_{\mu} |\mu\rangle C_i^{\mu}$ (usually delocalized, polarized by the molecular environment)

It is often hard to associate them with an atom.

One can construct the polarized AOs that can exactly express the occupied $|i\rangle$ using a reference minimal free-atom AO basis B_2.

Perform the following projection:

$$|\rho\rangle = (O\tilde{O} + (1-O)(1-\tilde{O}))P_{12}|\tilde{\rho}\rangle$$

polarized AOs, $|\rho\rangle \in B_1$

unpolarized AOs, $|\tilde{\rho}\rangle \in B_2$

Projectors of the occupied orbitals in B_1 and B_2 :

$$O = \sum_i |i\rangle\langle i|$$

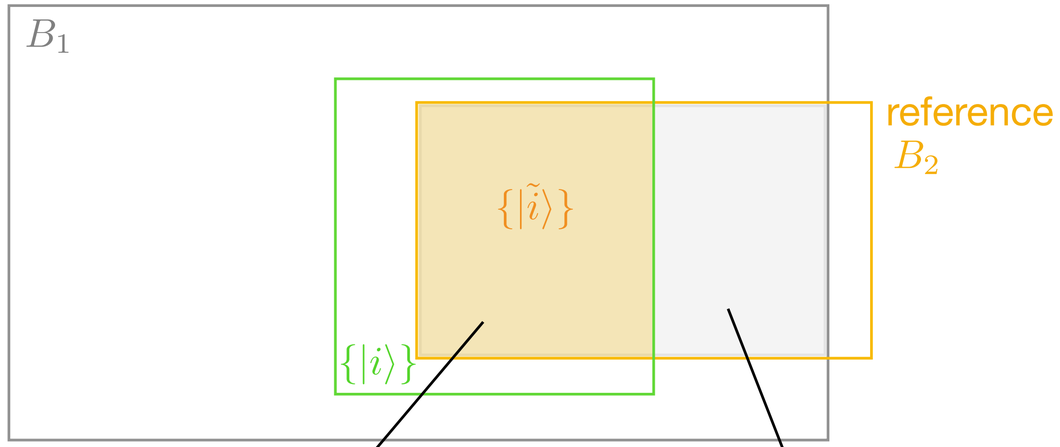
$$\tilde{O} = \sum_i |\tilde{i}\rangle\langle\tilde{i}| \qquad \{|\tilde{i}\rangle\} = \mathrm{orth}\,\{P_{12}P_{21}|i\rangle\}$$

Projector from B_2 to B_1 : $P_{12} = \sum_{\mu\nu\in B_1} |\mu\rangle S^{\mu\nu}\langle\nu|$

G. Knizia, JCTC, 9 4834 (2013)

Intrinsic Atomic Orbitals

big basis (that parameterizes our wave function)



$$|\rho\rangle = (O\tilde{O} + (1-O)(1-\tilde{O}))P_{12}|\tilde{\rho}\rangle$$

$O\tilde{O}\,\{P_{12}\{|\tilde{\rho}\rangle\}, |\tilde{\rho}\rangle \in B_2\}$
(occupied)

$(1-O)(1-\tilde{O})\,\{P_{12}\{|\tilde{\rho}\rangle\}, |\tilde{\rho}\rangle \in B_2\}$
(unoccupied)

Compute the intrinsic atomic orbitals for H_2

1. scf calculation (see last 2 downfolding school tutorials)

```
import pyscf
import pyscf.lo
from pyscf.tools import cubegen
import matplotlib.pyplot as plt
import numpy as np

mol = pyscf.gto.M(
    atom="H 0. 0. 0.; H 0. 0. 2”,
    basis=‘ccpvdz',
    )

mf = pyscf.scf.RHF(mol)
mf.kernel()
```

Start from r = 2 angstrom (close to the atomic limit)

Check that your

converged SCF energy = -0.921908594114884

The active space: the first 2 MOs

2. get the IAO, then orthonormalize

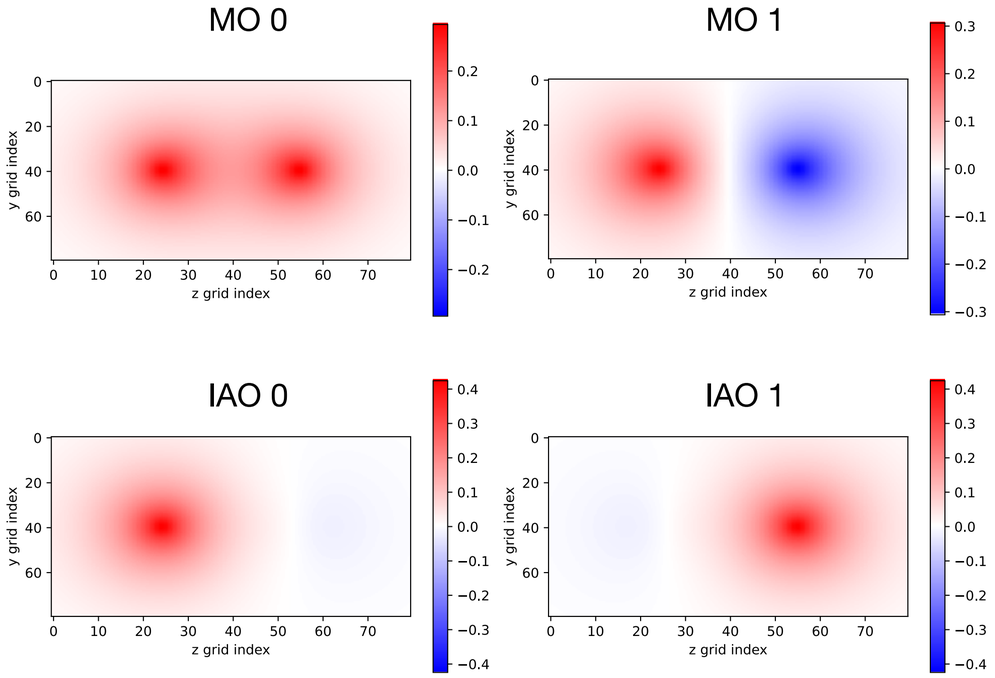
```
a = pyscf.lo.iao.iao(mol, mf.mo_coeff[:,:2])
a = pyscf.lo.vec_lowdin(a, mf.get_ovlp())
```

Plot the MOs and IAOs

3. Plot the orbital values in real space (on the yz plane)

```
def plot_orbital(orbital, output):
    plt.figure()
    range = np.max(np.abs(orbital))
    cm = plt.imshow(orbital[orbital.shape[0]//2,:,:], cmap='bwr',vmin=-range, vmax=range, aspect=0.5)
    plt.xlabel('z grid index')
    plt.ylabel('y grid index')
    plt.colorbar()
    plt.savefig(output,bbox_inches='tight')
```

```
# plot mo and iao
for i in np.arange(2):
  mo = cubegen.orbital(mol, f'h2_mo{i}.cube', mf.mo_coeff[:,i])
  iao = cubegen.orbital(mol, f'h2_iao{i}.cube', a[:,i])
  plot_orbital(mo, f'h2_mo{i}.pdf')
  plot_orbital(iao, f'h2_iao{i}.pdf')
```



Obtain the basis rotation matrix

4. The rotation matrix R between the MO basis and the IAO basis: $\langle A_i|B_j\rangle$

```
R = np.einsum("ij,ik,kl->jl", a, mf.get_ovlp(), mf.mo_coeff[:,:2])
print(R)
```

```
[[ 0.70710678  0.70710678]
 [ 0.70710678 -0.70710678]]
```

```
# Note that the rotation matrix is unitary.
print(R @ R.conj().T)
```

```
[[ 1.00000000e+00 -3.33066907e-16]
 [-3.33066907e-16  1.00000000e+00]]
```

MO: $|B_j\rangle = \sum_i \beta_{ij}|g_i\rangle$

IAO: $|A_j\rangle = \sum_i \alpha_{ij}|g_i\rangle$

$$\langle A_i|B_j\rangle = \sum_{kl} \alpha^*_{ki}\beta_{lj}\langle g_k|g_l\rangle$$

The overlap matrix between the gaussian basis (mf.get_ovlp())

$$R = A^\dagger S B$$

mf.mo_coeff

Obtain the tight-binding Hamiltonian

5. Rotate the Hamiltonian in MO basis (diagonal) into the IAO basis, using R: $H_{\mathrm{IAO}} = R H_{\mathrm{MO}} R^{\dagger}$

```
# rotate the Hamiltonian
H0 = np.array([[mf.mo_energy[0],0],[0,mf.mo_energy[1]]])

H_tb = np.einsum("ij,jk,kl->il", R, H0, R.conj().T)
print('tb model:\n', H_tb)
```

```
tb model:

      [[-0.21078147 -0.17493142]
      [-0.17493142 -0.21078147]]
```

Check the eigenvalues are the same:

```
eigvals, eigvecs = np.linalg.eigh(H_tb)

print('tb model eigenvalues:', eigvals)
print('original eigenvalues:', mf.mo_energy[:2])
```

```
tb model eigenvalues: [-0.38571289 -0.03585005]
original eigenvalues: [-0.38571289 -0.03585005]
```

Evaluate the 1RDM in IAO basis

6. Rotate the rdm1 in MO basis (diagonal) into the IAO basis, using R: $\rho_{\mathrm{IAO}} = R\rho_{\mathrm{MO}}R^{\dagger}$

Compute the rdm1 in MO basis

```
rdm1_ao = mf.make_rdm1()
B = mf.get_ovlp() @ mf.mo_coeff
rdm1_mo = np.einsum('ai, ab, bj->ij', B, rdm1_ao, B)

# compute the rotated rdm1
rdm1_iao = np.einsum("ij,jk,kl->il", R, rdm1_mo[:2,:2], R.conj().T)
print(rdm1_iao)
```

```
[[1. 1.]
 [1. 1.]]
```