ex05_hf_solution

March 15, 2025

Computational Quantum Physics - PHYS 463

Lecturer: Prof. G. Carleo

Assistants: alessandro.sinibaldi@epfl.ch, linda.mauron@epfl.ch, lorenzo.fioroni@epfl.ch

0.1 Solutions 5.1 - Molecular energies using PySCF

[1]:
```
import numpy as np
from pyscf import gto,scf,ao2mo,mp,cc,fci,tools
import matplotlib.pyplot as plt
```

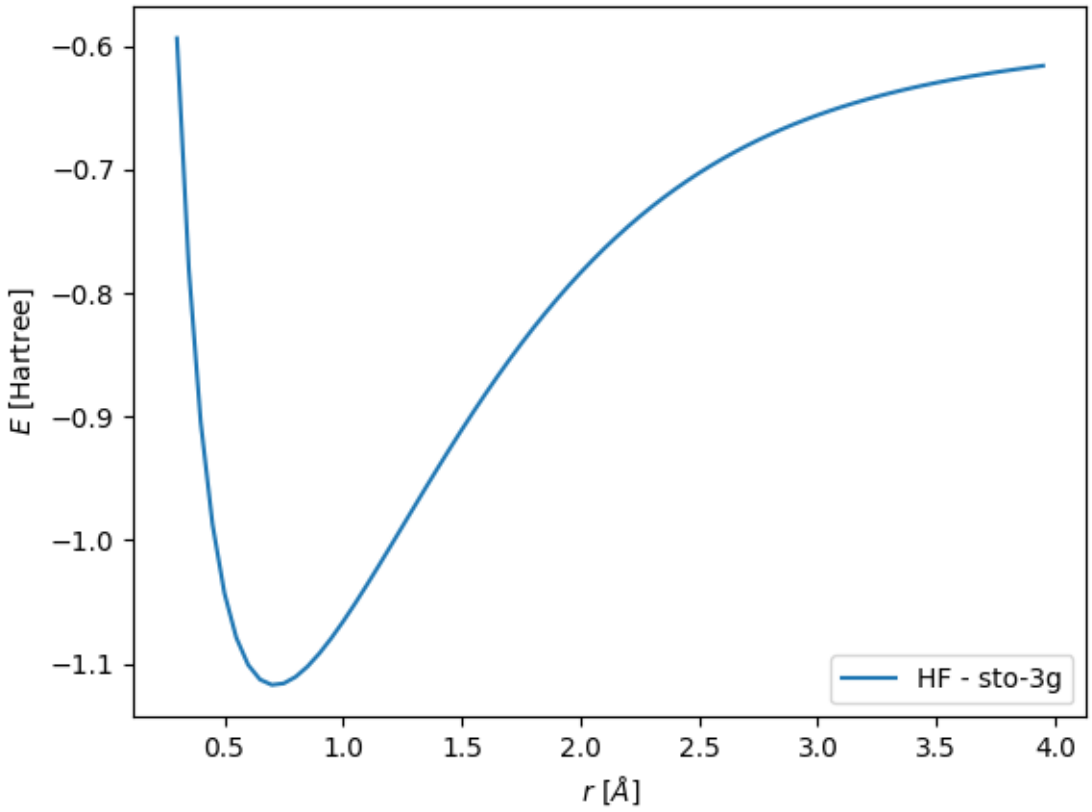
0.2 b) Hydrogen dissociation curve

[2]:
```
## Define a set of distances between the two atoms
distances = np.arange(0.3, 4, .05)
## and a basis in which the calculations will be made
basis = 'sto-3g'
## Define a dictionary containing the energies derived with different methods
energies = {}
```

0.2.1 The Hartree-Fock (HF) calculation

[3]:
```
energies["HF_"+basis] = []

for (i,r) in enumerate(distances):
    geometry = "H .0 .0 .0; H .0 .0 "+str(r)
    mol = gto.M(atom=geometry, charge=0, spin=0, basis=basis, symmetry=True,␣
 ↪verbose=0)
    mf  = scf.RHF(mol)
    Ehf = mf.kernel()

    energies["HF_"+basis].append(Ehf)
```

[4]:
```
plt.plot(distances,energies["HF_"+basis],label="HF - "+basis)
plt.xlabel(r"$r$ [$\AA$]")
plt.ylabel(r"$E$ [Hartree]")
plt.legend()
plt.show()
```



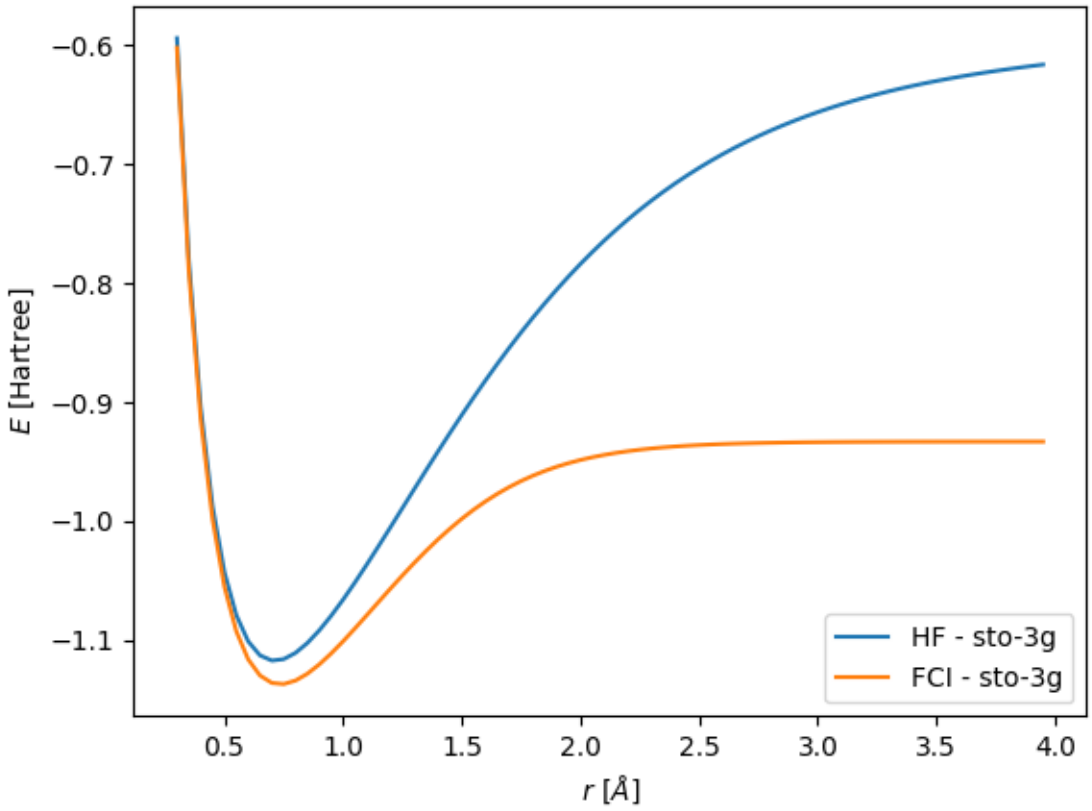
We can see that is pretty simple and requires only a bunch of lines to obtain a dissociation curve for a diatomic molecule.
However, the curve doesn't tell us very much about the quality of our simulation.

Let's compare HF to other methods.

0.2.2 The Full Configuration Interaction (FCI) method

[5]:
```
energies["FCI_"+basis] = []

for (i,r) in enumerate(distances):
    geometry = "H .0 .0 .0; H .0 .0 "+str(r)
    mol = gto.
 ↪M(atom=geometry,charge=0,spin=0,basis=basis,symmetry=True,verbose=0)
    mf  = scf.RHF(mol).run() # Here using run is important, since we are not␣
 ↪calling the HF kernel
    fci_h2 = fci.FCI(mf)
    Efci = fci_h2.kernel()[0]

    energies["FCI_"+basis].append(Efci)
```

[6]:
```
## Now plot the result

plt.plot(distances,energies["HF_"+basis],label="HF - "+basis)
plt.plot(distances,energies["FCI_"+basis],label="FCI - "+basis)
plt.xlabel(r"$r$ [$\AA$]")
plt.ylabel(r"$E$ [Hartree]")
plt.legend()
plt.show()
```



The two dissociation curves are qualitatively different!

Except for the different values of absolute energies, which are quantitative information, we must keep an eye on two important factors:

- the equilibrium position r_{eq}, corresponding to the r where the minimum of the energy is, has shifted, which means that the molceule is different from what HF predicted
- the dissociation energy, which is $E_{+\infty} - E_{\min} \sim E_{r>>r_{eq}} - E_{\min}$ varied significantly. In particular, there is a big difference for $r > 1.5\,\mathring{A}$, where the wavefunction is multireferential, meaning that can no longer be approximated reliably with a single Slater determinant, making HF method fail.

0.3 c) Comparison of different basis sets

Now we repeat the calculation with different basis sets and compare the results.

[7]:
```
basis_list = ['sto-3g', 'sto-6g','6-31g','cc-pvdz','cc-pvqz']
```

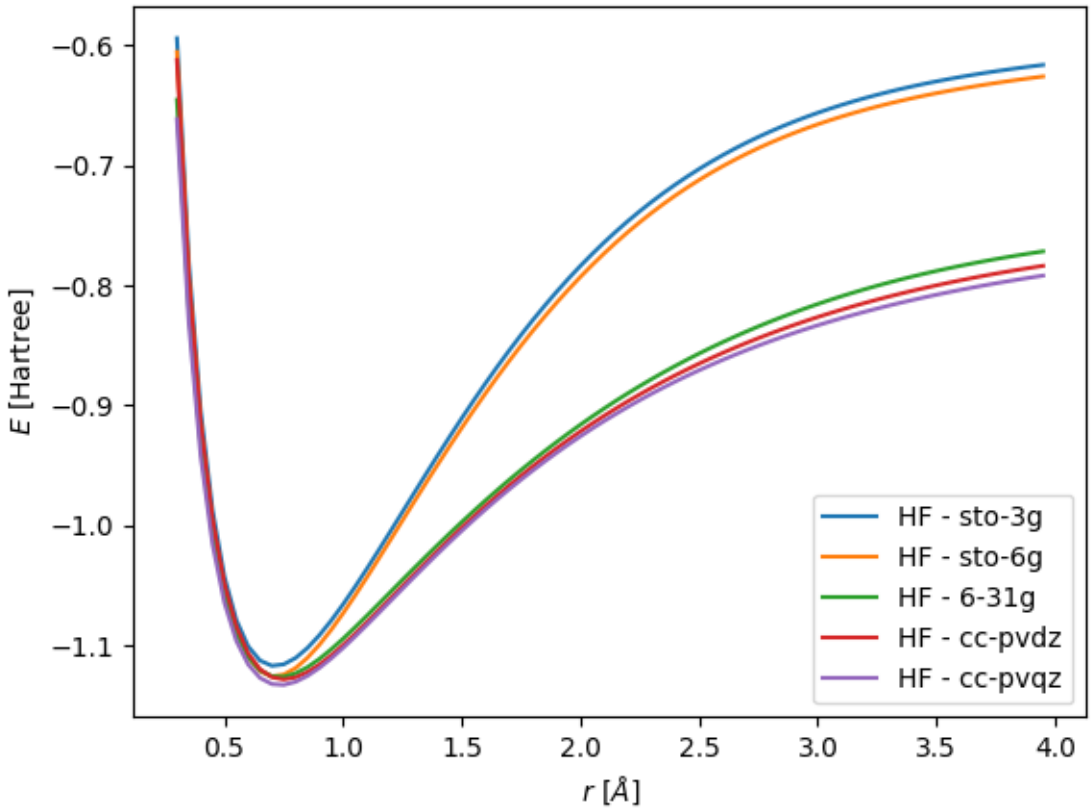
[8]:
```
for b in basis_list:
    print(b)

    energies["HF_"+b]  = []
    energies["FCI_"+b] = []

    for (i,r) in enumerate(distances):
        geometry = "H .0 .0 .0; H .0 .0 "+str(r)
        mol = gto.M(atom=geometry, charge=0, spin=0, basis=b, symmetry=True,␣
↪verbose=0)
        mf  = scf.RHF(mol)
        Ehf = mf.kernel()
        fci_h2 = fci.FCI(mf)
        Efci = fci_h2.kernel()[0]

        energies["HF_"+b].append(Ehf)
        energies["FCI_"+b].append(Efci)
```

```
sto-3g
sto-6g
6-31g
cc-pvdz
cc-pvqz
```

Now plot the comparison:

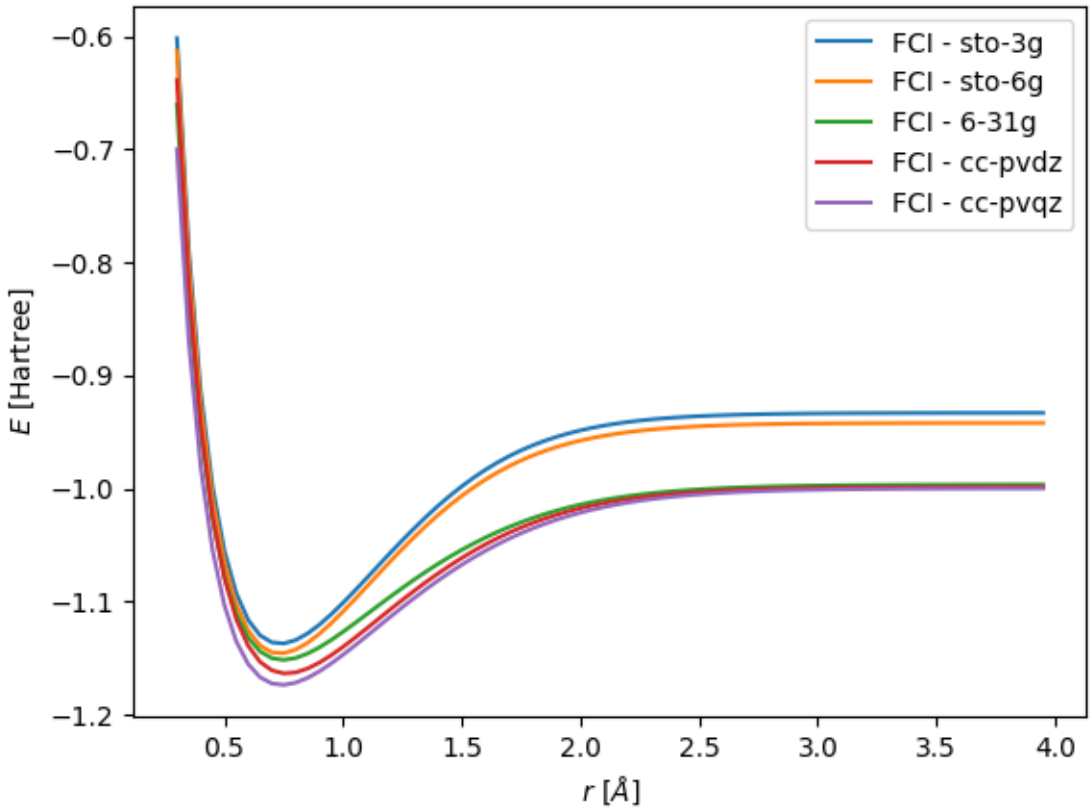
- HF

[9]:
```
for (i,base) in enumerate(basis_list):
    plt.plot(distances,energies["HF_"+base],label="HF - "+base,color="C"+str(i))

plt.xlabel(r"$r$ [$\AA$]")
plt.ylabel(r"$E$ [Hartree]")
plt.legend()
plt.show()
```



- FCI methods

[10]:
```
for (i,base) in enumerate(basis_list):
    plt.plot(distances,energies["FCI_"+base],label="FCI -␣
 ↪"+base,color="C"+str(i))

plt.xlabel(r"$r$ [$\AA$]")
plt.ylabel(r"$E$ [Hartree]")
plt.legend()
plt.show()
```



We can see that the FCI calculation in a minimal basis set is different from the other proposed solution.

In particular, this solution in particularly off in predicting the correct dissociation energy and how rapidly the energy changes when moving from a equilibrium configuration to a dissociated one.

As we consider a bigger and bigger set of orbitals, the FCI solution converge to the true solution *in the Born-Oppenheimer approximation.*

0.4 Exercise 5.2 - Restricted Hartree-Fock (RHF)

[11]:
```
from pyscf import gto,scf
import numpy as np
import scipy
```

In this exercise we find the mean-field solution of the Hamiltonian

$$\hat{H} = \sum_{ij\sigma} t_{ij}\, c^{\dagger}_{i\sigma}\hat{c}_{j\sigma} + \frac{1}{2}\sum_{ijkl\sigma\sigma'} V_{ijkl}\, \hat{c}^{\dagger}_{i\sigma}\hat{c}^{\dagger}_{k\sigma'}c_{l\sigma'}c_{j\sigma} + E_{nuc}.$$

a) Compute Integrals with PySCF We use PySCF to compute the coefficients of the one-body term

$$t_{\alpha\beta} = \int f_\alpha^\star(\mathbf{r}) \left(-\frac{\hbar^2}{2m}\nabla^2 - e^2 \sum_i \frac{Z_i}{|\mathbf{R}_i - \mathbf{r}|} \right) f_\beta(\mathbf{r})\, d\mathbf{r}$$

the coefficients of the two-body term (in the convention used by PySCF)

$$V_{\alpha\beta\gamma\delta} = \iint f_\alpha^\star(\mathbf{r}) f_\beta(\mathbf{r}) \frac{e^2}{|\mathbf{r} - \mathbf{r}'|} f_\gamma^\star(\mathbf{r}') f_\delta(\mathbf{r}')\, d\mathbf{r}\, d\mathbf{r}'$$

the nuclear repulsion energy

$$E_{nuc} = e^2 \sum_{i \neq j} \frac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j|}$$

and the overlap matrix

$$S_{\alpha\beta} = \langle f_\alpha | f_\beta \rangle = \int f_\alpha^\star(\mathbf{r}) f_\beta(\mathbf{r})\, d\mathbf{r}.$$

We choose the `cc-pVTZ` basis set for the basis functions f_α, which is already built into PySCF.

[12]:
```
# define the molecule object
mol = gto.M(
    atom = """
            H        0.000000    0.755453   -0.471161
            H        0.000000   -0.755453   -0.471161
            O        0.000000    0.000000    0.117790
            """,
    basis = 'cc-pVTZ',
)

N = int(mol.nelectron)
M = mol.nao
E_nuc = mol.energy_nuc()

S = mol.intor('int1e_ovlp')
t = mol.intor('int1e_nuc') + mol.intor('int1e_kin')
V = mol.intor('int2e') # (PQ|RS) = PR|QS
```

b) Solve the Roothan-Hall equations

[13]:
```
# random initial guess
C = np.random.normal(size=(M, M))
```

For the implementation of the self-consistent field method we need the density matrix defined as

$$P_{\alpha\beta} = 2 \sum_{i=1}^{N/2} C_{\alpha i}^\star C_{\beta i},$$

where N is the number of electrons, as well as Fock matrix given by (using the convention for $V_{\alpha\beta\gamma\delta}$ above)

$$F_{\alpha\beta} = t_{\alpha\beta} + \sum_{\gamma\delta} P_{\gamma\delta} (V_{\alpha\beta\gamma\delta} - \frac{1}{2} V_{\alpha\gamma\beta\delta}).$$

Then we have to solve generalized eigenvalue problem given by the Roothan-Hall Equation

$$\sum_{\beta}(F_{\alpha\beta} - \epsilon_k S_{\alpha\beta})C_{\beta k} = 0$$

for which we can use the generalized eigenvalue solver from scipy.

It finds a new set of orbitals

$$|\phi_k\rangle = \sum_{\alpha} C_{\alpha k}|f_\alpha\rangle$$

which are orthonormal:

$$\langle\phi_k|\phi_l\rangle = \sum_{\alpha\beta} C^{\star}_{\alpha k} C_{\beta l} S_{\alpha\beta} = \delta_{kl}$$

Furthermore we can compute the Hartree-Fock energy as

$$E_0 = \frac{1}{2}\sum_{\alpha\beta}(t_{\alpha\beta} + F_{\alpha\beta})P_{\alpha\beta} + E_{nuc}.$$

[14]:
```
def hf_step(t, V, E_nuc, S, N, C):
    P = 2 * np.einsum('ai,bi->ab', C[:, :N//2].conj(), C[:, :N//2])
    F = t + np.einsum('gd,abgd->ab', P, V-0.5*np.transpose(V,(0,2,1,3)))
    E0 = 0.5 * ((t + F)*P).sum() + E_nuc

     , C = scipy.linalg.eigh(F, S)
    # check eigenvectors and eigenvalues
    np.testing.assert_allclose(F@C-S@C@np.diag( ), 0, atol=1e-11)
    # check orthogonality
    np.testing.assert_allclose(np.einsum("ak,bl,ab->kl", C.conj(), C, S), np.
 ↪eye(C.shape[0]), atol=1e-12)
    return C, E0
```

c) Run the SCF procedure

[15]:
```
E0 = 0
for i in range(50):
    C, E0 = hf_step(t, V, E_nuc, S, N, C)
    print(i, f"{E0:0.4f}")
```

```
0 76911.8578
1 -0.0164
2 -61.4545
3 -66.7201
4 -71.7599

5 -70.2123
6 -73.1047
7 -72.9083
8 -74.3118
9 -75.1280
```

```
10 -75.6498
11 -75.8796
12 -75.9855
13 -76.0281
14 -76.0456

15 -76.0525
16 -76.0553
17 -76.0564
18 -76.0568
19 -76.0570

20 -76.0571
21 -76.0571
22 -76.0571
23 -76.0571
24 -76.0571

25 -76.0571
26 -76.0571
27 -76.0571
28 -76.0571
29 -76.0571

30 -76.0571
31 -76.0571
32 -76.0571
33 -76.0571
34 -76.0571

35 -76.0571
36 -76.0571
37 -76.0571
38 -76.0571
39 -76.0571

40 -76.0571
41 -76.0571
42 -76.0571
43 -76.0571
44 -76.0571

45 -76.0571
46 -76.0571
47 -76.0571
48 -76.0571
49 -76.0571
```

d) Compare the result with PySCF

[16]:
```
# check with pyscf
mf  = scf.RHF(mol)
E_hf = mf.kernel()
print(f"{float(E_hf):0.4f}")
```

```
converged SCF energy = -76.0570982356602
```

```
-76.0571
```

[ ]:
```

```