1-Dimentional Kohn-Sham Density functional theory

Cankut Tasci

CUNY Graduate Center

December 19, 2022

Schrödinger equation

$$\hat{H}\psi(\vec{r}) = E\psi(\vec{r}) \tag{1}$$

$$\hat{H} = \hat{T} + \hat{W}_{ee} + \hat{V} \tag{2}$$

$$\hat{T} = -\frac{1}{2}\sum_{i=1}^{N} \nabla_{r_i}^2 \tag{3}$$

$$\hat{W}_{ee} = \sum_{i<j}^{N} \frac{1}{|r_i - r_j|} \tag{4}$$

$$\hat{V} = \sum_{i=1}^{N} v(r_i) \tag{5}$$

$$v(r) = -\sum_{A}^{nuclei} \frac{Z_A}{|r - R_A|} \tag{6}$$

Non interacting electrons

$$\hat{W}_{ee} \to 0 \tag{7}$$

$$\left(\hat{T} + \hat{V}\right)\psi_i(r_i) = \epsilon_i \psi_i(r_i) \tag{8}$$

$$\psi(r) = \prod_i \psi_i(r_i) \tag{9}$$

$$E = \sum_i \epsilon_i \tag{10}$$

$$\tag{11}$$

Mapping interacting problem onto non-interacting problem

- Is it possible to extract the exact(interacting)ground-state energy from a non-interacting system?
- One way to establish a connection between interacting and non-interacting worlds is to use the electron density as basic variable (instead of the wavefunction)
- Electron density for a non-interacting system:
  $n_{non}(r) = \sum_{\sigma=\pm 1/2} \sum_{i=1}^{N} \left|\psi_i(r,\sigma)\right|^2$
- Electron density for interacting system
  $n(r) = N \sum_{\sigma=\pm 1/2} \int dx_2 ... \int dx_N \left|\psi(r,\sigma,x_2,...,x_N)\right|^2$

Kohn-Sham DFT

- $n_{non}(r) \leftrightarrow n(r)$
- We may assume that it is possible to adjust the local potential in the non-interacting system such that the two densities become equal.
- $\hat{W}_{ee} \rightarrow 0$
- $v(r) \rightarrow v(r)^{KS}$
- $n_{non}(r) = n(r)$

Kohn-Sham DFT

- First Honenberg-Kohn theorem: The ground state electron density fully determines the local potential v. Therefore, we have $n(r) \to v(r) \to \psi_0(r) \to E_0$

Kohn-Sham DFT

- Second Honenberg-Kohn theorem: Exact energy density of $\hat{H}$ minimizes the energy density functional:
  $E[n] = F[n] + \int v_{ne}(r)n(r)dr$
  $F[n] = \langle \psi[n] | \hat{T} + \hat{W}_{ee} | \psi[n] \rangle$

Kohn-Sham DFT

- $\hat{W}_{ee} \to 0$
- $v[n](r) \to v[n]^{KS}(r)$
- $\psi[n] \to \psi^{KS}[n]$
- $F[n] \to T_s[n] = \left\langle \psi^{KS}[n] \right| \hat{T} \left| \psi^{KS}[n] \right\rangle$
- KS decomposition: $F[n] = T_s[n] + E_{Hxc}[n]$
  $E_{Hxc}[n] = \frac{1}{2} \int \int \frac{n(r)n(r')}{|r-r'|} drdr' + E_{xc}$
  $E_{xc}[n] = E_{exchange}[n] + E_{correlation}[n]$
  $v[n]^{KS} = v_{ne} + \frac{\delta E_{xc}[n]}{\delta n(r)} = v_{ne} + v_{exchange} + v_{correlation}$
- $(\hat{T} + v[n]^{KS})\psi_i^{KS} = E_i^{KS}\psi_i^{KS}$

1-D Kohn Sham DFT

- $\hat{H}\psi(x) = E\psi(x)$
- $\hat{H} = -\frac{1}{2}\frac{d^2}{dx^2} + V_{Har}(x) + V_X(x) + V_{external}$
- $\int \left|\psi(x)\right|^2 dx = 1$
- $n(x) = \sum_n f_n \left|\psi_n(x)\right|^2$
- $E_X^{LDA} = -\frac{3}{4}(\frac{3}{\pi})^{1/3} \int n^{4/3}(x)dx$
- $V_X^{LDA}(x) = -(\frac{3}{\pi})^{1/3} n^{1/3}(x)$
- $E_{Har} = \frac{1}{2}\int\int \frac{n(x)n(x')}{\sqrt{(x-x')^2+\epsilon}} dxdx'$
- $V_{Har}(x) = \frac{\partial E_{Ha}}{\partial n(x)} = \int \frac{n(x')dx'}{\sqrt{(x-x')^2+\epsilon}}$

Algorithm for solving the system Self-Consistently

- Choose a proper grid for x values
- Check the selection with known potentials
- Represent second derivative as a matrix operator
- Start with initial selection of $n_0(x)$ for first iteration
- Calculate V_{Har} and V_X^{LDA} values
- Represent Hamiltonian as a Matrix and solve for eigenvalues and eigenvectors.
- Calculate total energy
- Update the density with $n(x) = \sum_n f_n \left|\psi_n(x)\right|^2$
- Follow the same steps
- If the energy difference between each iteration becomes small enough, break the process

1-D Kohn-Sham DFT

- x=np.linspace(-5,5,grid)
- $(\frac{dy}{dx})_i = D_{ij} y_j$
- $D_{ij} = \frac{\delta_{i+1,j} - \delta_{i,j}}{h}$
- $D^2_{ij} = -D_{ik} D_{kj}$

Code

```
#Grid testing with known potentials
grid=200
x=np.linspace(-5,5,grid)
y=np.sin(x)
h=x[1]-x[0]
D=-np.eye(grid)+np.diagflat(np.ones(grid-1),1)
D = D / h
     #Second order derivative
D2=D.dot(-D.T)
     #Getting rid of the ends
D2[-1,-1]=D2[0,0]
pothar=x*x
pot=np.full_like(x,1e10)
pot[np.logical_and(x>-2,x<2)]=0
H=-D2/2+np.diagflat(pothar)
energy, psi= np.linalg.eigh(H)
for i in range(5):
    plt.plot(x,psi[:,i],label=f'Psi {i}')
    plt.legend()

#Firt order derivative

def KSDFT(
```

Figure 1: Step 1-3

Grid testing

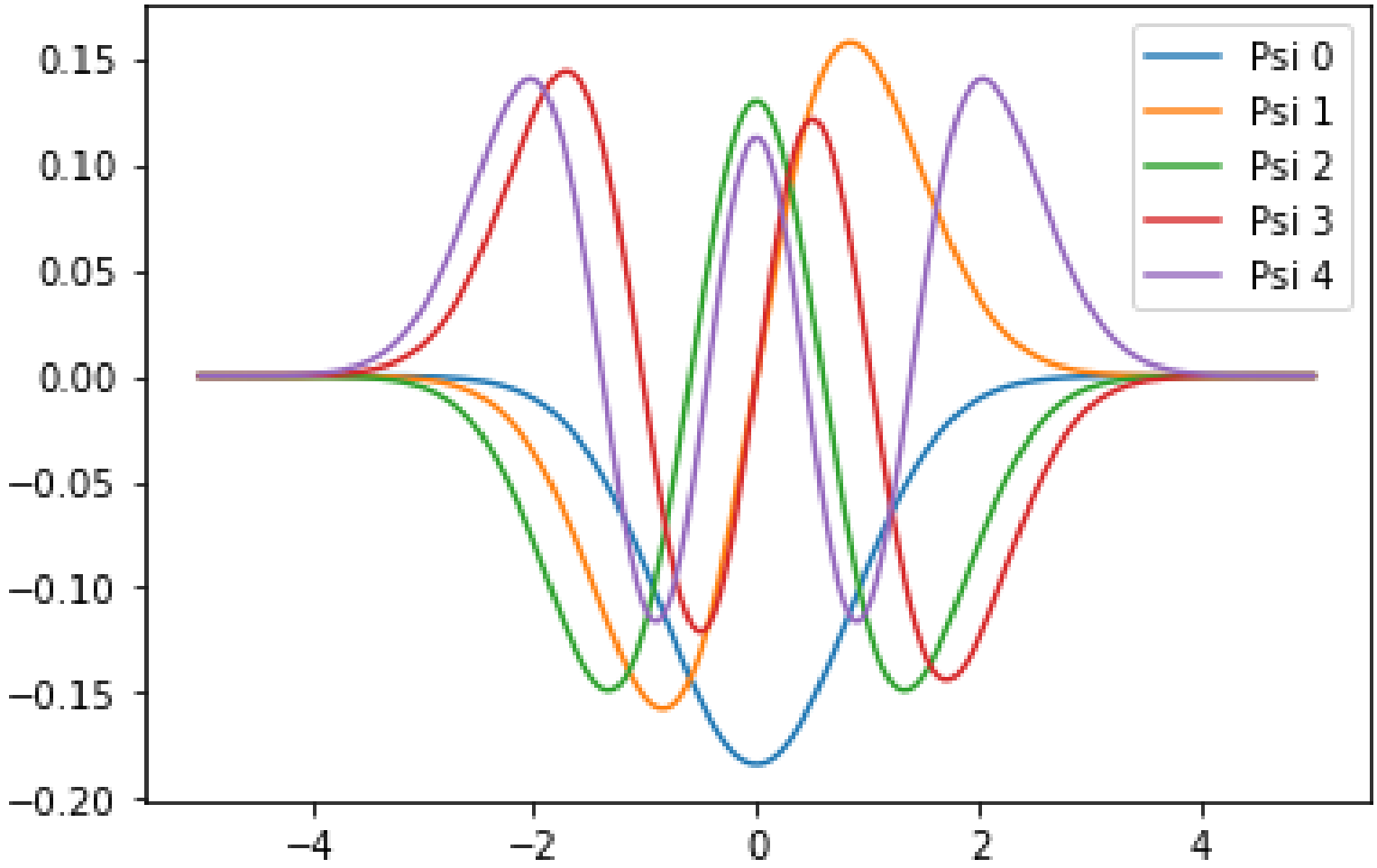

Figure 2: Grid test with harmonic oscillator

1-D Kohn-Sham DFT

```
def KSDFT(ne,grid,pot,x):
     h=x[1]-x[0]
     D=-np.eye(grid)+np.diagflat(np.ones(grid-1),1)
     D = D / h
     #Second order derivative
     D2=D.dot(-D.T)
     #Getting rid of the ends
     D2[-1,-1]=D2[0,0]
     #Density
     # integral
     # integral(Mean value method)
     def integral(x,y):
         a=x[0]
         b=x[-1]
         N=len(x)
         sum1=0
         for i in range(N):
             sum1+=y[i]
         return (b-a)*sum1/N

     def density(ne, psi, x):
         # normalization

         C=integral(x,psi**2)
         normed_psi=psi/np.sqrt(C)
         oc=[]
         for i in range(ne//2):
              oc.append(2)
         if ne%2:
              oc.append(1)
         n=np.zeros(grid)
         if len(oc)>=grid:
           for j in range(grid):
               n+=oc[j]*normed_psi.T[j,:]**2
         else:
           for j in range(len(oc)):
               n+=oc[j]*normed_psi.T[j,:]**2
         #print(n)
         return n
```

1-D Kohn-Sham DFT

```
def Exchange(n,x):
    Eex=-3/4*(3/np.pi)**(1/3)*integral(x,n**(4/3))
    Vex=-(3/np.pi)**(1/3)*n**(1/3)
    return Eex, Vex
def Hatree(n,x, eps=1e-1):
    h=x[1]-x[0]
    Eh=0
    for i in range(len(n)):
            for j in range(len(n)):
                Eh+=1/2*(n[i]*n[j]*h**2)/(np.sqrt((x[i]-x[j])**2+eps))
    Vh=[]
    for i in range(len(n)):
        V0=0
        for j in range(len(n)):
            V0+=n[j]*h/(np.sqrt((x[i]-x[j])**2+eps))
        Vh.append(V0)
    Vh=np.array(Vh)
    return Eh, Vh
```

Figure 4: Exchange and Hartree energy (potential)

1-D Kohn-Sham DFT

```
93      def print_log(i,log):
94           print(f"step: {i:<5} energy: {log['energy'][-1]:<10.4f} energy_diff: {log['energy_diff'][-1]:.10f}")
95
96      max_iter=1000
97      energy_tolerance=1e-5
98      log={"energy":[float("inf")], "energy_diff":[float("inf")]}
99      n=np.zeros(grid)
100     for i in range(max_iter):
101             ex_energy, ex_potential=Exchange(n,x)
102             ha_energy, ha_potential=Hatree(n,x)
103          # Hamiltonian
104             H=-D2/2+np.diagflat(ex_potential+ha_potential+pot)
105             energy, psi= np.linalg.eigh(H)
106           # log
107             log["energy"].append(energy[0])
108             energy_diff=energy[0]-log["energy"][-2]
109             log["energy_diff"].append(energy_diff)
110             print_log(i,log)
111
112             # convergence
113             if abs(energy_diff) < energy_tolerance:
114                 print("converged!")
115                 break
116
117             n=density(ne,psi,x)
118
119     else:
120                print("not converged")
121 KSDFT(ne,grid,pot,x)
122
```

Figure 5: Self-Consistent Iteration

Results

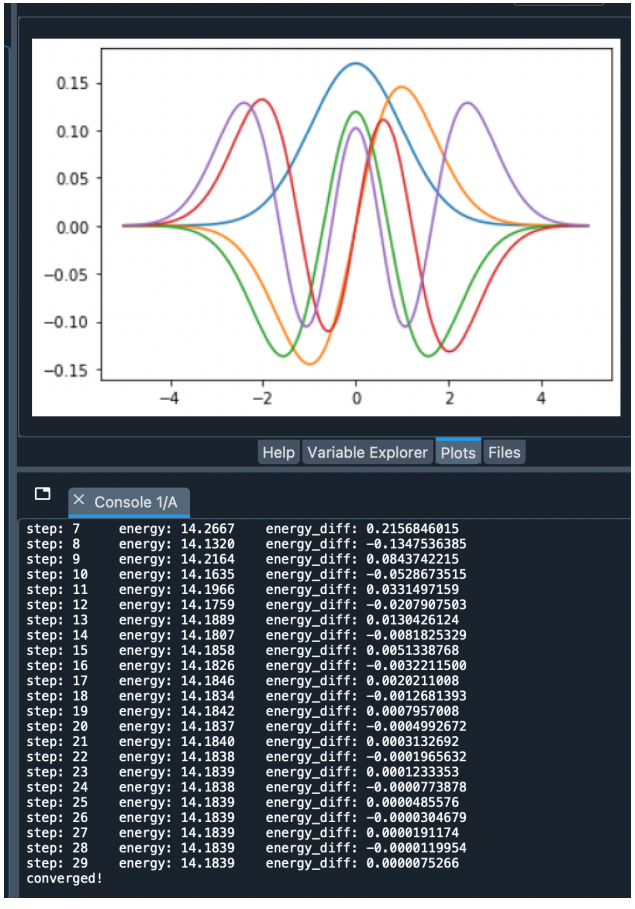

Figure 6: Ne=16, $V = x^2$

Results

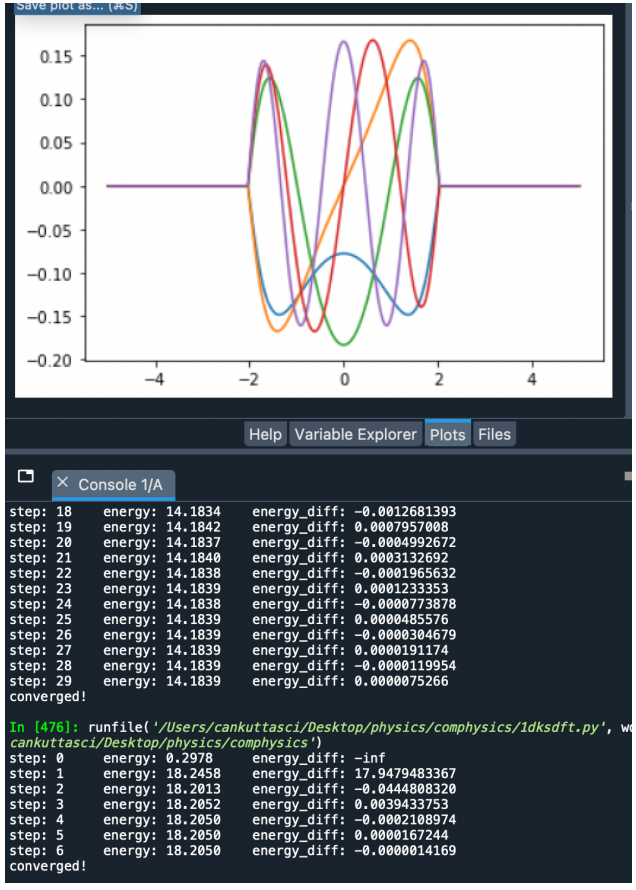

Figure 7: Ne=16, Infinite square well potential $|x| < 2$

References

- H. Larsen, K. Lyon, "Write a simple DFT code in python", 2018