Write a simple 1D DFT code in Python

Ask Hjorth Larsen, asklarsen@gmail.com
Keenan Lyon, lyon.keenan@gmail.com

September 15, 2018

Overview

Our goal is to write our own Kohn–Sham (KS) density functional theory (DFT) code.

A full-featured DFT code is very complex, so we limit our ambitions to the simplest possible model that is still interesting: We will iteratively solve the Kohn–Sham equations for a harmonic oscillator including electronic kinetic energy, electrostatic repulsion between the electrons, and the local density approximation for electronic interactions, ignoring correlation.

This gives us the full Hamiltonian including the potential $v(x)$ which we write as

$$\hat{H} = -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + v(x) = -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + v_{\mathrm{Ha}}(x) + v_{\mathrm{X}}^{\mathrm{LDA}}(x) + x^2. \tag{1}$$

We must be able to calculate the KS wavefunctions, the density, and each of the potentials required to represent the Hamiltonian. We must also represent the Hamiltonian somehow, including the kinetic operator. But one thing at a time.

Python

Python is a dynamically typed language. Python programs are executed by the Python interpreter. There are two main versions: `python2` (also called just `python`) and `python3`. Either is fine, but we use `python3`. Open a terminal and run the interactive Python interpreter:

```
askhl@hagen:~$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('hello world!')
hello world!
>>> 2+2
4
```

```
>>> items = [1, 2, 3, 'hello']
>>> for obj in items:
...     print(obj)
...
1
2
3
hello
>>>
```

Note that the scope of the `for` loop is controlled by indentation. The loop ends once the code is no longer indented. It is conventional to indent by 4 spaces in Python.

Alternatively we can write a script, opening our favourite text editor (gedit is the standard one in Ubuntu) and typing:

```
print('hello world!')
```

Then save the script as `hello.py` and run it with `python3 hello.py`.

We always want to write *scripts*, but the interactive interpreter is an excellent tool to play around and test things before adding them to the script.

Use `help(obj)` to see the documentation for any object (including functions and modules).

If you are entirely unfamiliar with Python, be sure to have a look at the official Python tutorial. You will need to know basic things like printing and writing loops.

Grids

The simplest way to represent a real function $f(x)$, with $a \leq x \leq b$, is to sample it on a real-space grid of points $\{x_i\}$ from a to b with some uniform spacing h. The function is then represented by the vector of values $\{f(x_i)\}$.
We can use matplotlib to plot $\sin(x)$ on a suitable grid:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-5, 5, 200)  # define grid
y = np.sin(x)
plt.plot(x, y)
plt.show()
```

I. How would you approximate the first and second derivatives of a function from its representation $\{f(x_i)\}$ on the grid using finite differences? Calculate and verify using matplotlib that your numerical derivatives are correct.

Hint: The derivatives may not be well defined at the *ends* of the grid; exclude the first and/or last grid points to suit your needs.
Here are some examples of how to work with Numpy arrays:

```
import numpy as np

array = np.zeros(10)  # new array with 10 zeros
array[3] = 5  # Assign to an element
array[5:8] += 2  # In-place add to several elements
print(array[5])  # Print an element
array2 = 2 * array  # Multiply by 2, creating new array
array *= 2  # Multiply all elements in-place

array2D = np.random.rand(4, 8)  # 4 x 8 random numbers
print(array2D)  # Print whole array
print(array2D[:, 0])  # Print first column
print(array2D[:, 7])  # Print last column
print(array2D[0, :])  # Print first row
```

Free non-interacting electrons

To get the Kohn–Sham wavefunctions we must diagonalize the Hamiltonian, i.e., calculate its eigenvectors. We will establish a matrix representation of the Hamiltonian starting with the kinetic operator $\hat{T}$.

If our grid has N points, then linear operators on that space are represented by $N \times N$ matrices. An operator O is applied to a state $\psi(x)$ by left-multiplying it onto the state: `Opsi = np.dot(O, psi)`.

> II. How can we represent the kinetic operator ($-1/2$ times the second derivative) as a matrix? Use the expression for the second derivative derived earlier.

Now we have a matrix representation of the kinetic operator; this is the Hamiltonian of non-interacting free particles in a box given by the size of our grid:

$$\hat{H} = \hat{T} = -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} \tag{2}$$

The Kohn–Sham states and their energies are the eigenvectors and eigenvalues of that matrix. The matrix is real and symmetric, so both eigenvalues and eigenvectors are real and we can use `np.linalg.eigh` which is for Hermitian matrices:

```
epsilon_n, psi_gn = np.linalg.eigh(T)
```

Above, we use `_n` as a hint that the array has an element for each state, and `_gn` means a 2D array indexed first by grid point, then state. This tends to improve readability of code with many arrays. Thus, `psi_gn[:, 0]` is the whole state

vector of the first state, and `psi_gn[:, :5]` is a 2D matrix including only the first five.

> III. Choose a grid and calculate the free-particle energies and wavefunctions. What are the eigenvalues? Plot the wavefunctions $\psi_n(x)$ with the lowest 5 energies.

Harmonic oscillator

Now we include the external potential $v_{\text{ext}}(x) = x^2$ in the Hamiltonian:

$$\hat{H} = \hat{T} = -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + x^2. \tag{3}$$

This is the harmonic oscillator for non-interacting particles. How can you represent x^2 as a matrix?

> IV. Calculate the Hamiltonian and plot the 5 states with lowest energy, making sure that your grid is adequate.

Density

We will want to include both the Coulomb or Hartree interaction as well as LDA exchange, both of which are density functionals. Hence we need to calculate the electron density. Each state should be normalized so it integrates to one (having the capacity to contain one electron):

$$\int |\psi(x)|^2 \,\mathrm{d}x = 1. \tag{4}$$

The normalization from `np.linalg.eigh` will be different.

> V. How do you calculate the integral of a function on the grid? Normalize the states so they integrate to 1.

Then the electron density in DFT is given by

$$n(x) = \sum_n f_n |\psi_n(x)|^2, \tag{5}$$

where f_n are the *occupation numbers*.

In DFT we calculate the ground state: The electrons will occupy the states with lowest energy. Each state fits up to two electrons: one with spin up, and one with spin down. A state can therfore be occupied by 0, 1, or 2 electrons.

Let us say that we have 6 electrons. Then the three lowest states will have occupation number $f = 2$ and all others $f = 0$.

> VI. Calculate and plot the electron density for 6 electrons in the harmonic potential. Verify that the density integrates to 6 electrons.

Exchange energy

The exchange–correlation functional is a correction to the electronic energy that approximates the effect of electron interactions.[1] To keep life simple, we ignore correlation because its expression is more tedious than interesting to implement. Consider therefore the exchange functional in the local density approximation (LDA) which is

$$E_{\mathrm{X}}^{\mathrm{LDA}}[n] = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3} \int n^{4/3}\,\mathrm{d}x \tag{6}$$

In the derivation of the Kohn–Sham equations, the potential corresponding to each energy term is defined as the derivative of the energy with respect to the density. The *exchange potential* is therefore given by the derivative of the exchange energy with respect to the density:

$$v_{\mathrm{X}}^{\mathrm{LDA}}(x) = \frac{\partial E_{\mathrm{X}}^{\mathrm{LDA}}[n]}{\partial n(x)} = -\left(\frac{3}{\pi}\right)^{1/3} n^{1/3}(x) \tag{7}$$

> VII. Calculate the exchange potential and energy from the previously calculated density.

It is useful to define a function to do this calculation:

```
def calculate_exchange(density):
    energy = ...  # Calculate
    potential = ...  # Calculate
    return energy, potential

mydensity = np.random.rand(10)
energy, potential = calculate_exchange(mydensity)
```

> VIII. Write the calculation of exchange as a function.

[1] The Hartree interaction is the interaction between the electron density and itself. The exchange interaction includes a correction to the Hartree interaction, so in a sense it is unphysical to implement exchange *before* Hartree interaction. We do it anyway because the LDA exchange expression is simpler than the Hartree interaction.

Coulomb potential

(If short of time, consider skipping this exercise for now and implement the self-consistency loop first, ignoring the Coulomb potential)

The electrostatic energy or Hartree energy is given by

$$E_{\mathrm{Ha}}^{\mathrm{3D}} = \frac{1}{2} \iint \frac{n(\mathbf{r}) n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \, \mathrm{d}\mathbf{r} \, \mathrm{d}\mathbf{r}'. \tag{8}$$

This expression converges in 3D, but not in 1D. Hence we cheat and use a modified, "softened" form:

$$E_{\mathrm{Ha}} = \frac{1}{2} \iint \frac{n(x) n(x')}{\sqrt{(x - x')^2 + 1}} \, \mathrm{d}x \, \mathrm{d}x'. \tag{9}$$

Again, the potential is the derivative of the energy with respect to the density:

$$v_{\mathrm{Ha}}(x) = \frac{\partial E_{\mathrm{Ha}}}{\partial n(x)} = \int \frac{n(x')}{\sqrt{(x - x')^2 + 1}} \, \mathrm{d}x' \tag{10}$$

IX. Write a function to calculate the Coulomb energy and potential, then plot the potential for the previous density.

Self-consistency loop

The self-consistency loop repeats the following steps, calculating:

- Exchange energy and potential from the density (you can start with a constant density, e.g. 0)
- Coulomb energy and potential from the density.
- Hamiltonian from kinetic operator and potentials
- Wavefunctions and their energies from the Hamiltonian
- Density from the normalized wavefunctions and occupations

X. Implement the self-consistency loop and iterate enough times to converge and plot the density.

You may wish to print interesting quantities like the energy contributions from within the self-consistency loop. When the density does not change anymore, the calculated quantities are *self-consistent*, and the calculation is done.

Total energy

The kinetic energy is

$$T = \int \psi^*(x) \left(-\frac{1}{2} \frac{\mathrm{d}^2}{\mathrm{d}x^2} \right) \psi(x) \, \mathrm{d}x = \sum_n f_n \epsilon_n - \int n(x) v(x) \, \mathrm{d}x, \tag{11}$$

where $v(x) = v_{\mathrm{Ha}}(x) + v_{\mathrm{X}} + x^2$ is the total potential.

XI. Calculate the kinetic energy and each of the other energy contributions to get the final total energy:

$$E = \sum_n f_n \epsilon_n - \int n(x) v(x) \, \mathrm{d}x + E_{\mathrm{Ha}}[n] + E_{\mathrm{X}}[n] + \int n(x) v_{\mathrm{exc}}(x) \, \mathrm{d}x. \tag{12}$$

The end

The DFT code is now complete! If you want, you can wrap it all in a single function whose parameters are the grid, the external potential, and the number of electrons. Then it can be easily called to perform different calculations.