**Numerical Algorithms Applied to Computational Quantum Chemistry
Homework 6: Build Your Own Density Functional Theory Program**

Prof. Martin Head-Gordon, with Zhe Wang (C279) and Xiao Liu (C179)

Due May 2, 2024

1 INTRODUCTION

In this problem set, you will build code to calculate the electronic energy of atoms and small molecules by solving the Kohn-Sham equations within the framework of a simplified exchange-only version of the Local Density Approximation (LDA) functional. This simplified LDA model is actually quite famous: it was called the $X-\alpha$ method, and was the first "modern" DFT approach to be widely used in chemistry and physics even before the Kohn-Sham theory was invented. Maybe it helped inspire KS-DFT!

As you know, it can be a lot of work to write a DFT code, so we have made some unusual choices to simplify the process. The most important one is making the matrix elements straightforward to evaluate compared to production DFT codes. For this purpose we want you to use choose particle-in-a-box (PB) basis functions. These are the functions that solve the Schrodinger equation for a particle trapped inside a cubic 3-D box of length L. As a function of $\mathbf{r} = (x, y, z)$, where x, y, z each can range from $[-\frac{L}{2}, \frac{L}{2}]$, the PB wavefunctions are given by

$$\omega_{\mathbf{n}}(\mathbf{r}) = \left(\frac{2}{L}\right)^{3/2} \sin\left(\frac{n_x \pi x}{L} + \frac{n_x \pi}{2}\right) \sin\left(\frac{n_y \pi y}{L} + \frac{n_y \pi}{2}\right) \sin\left(\frac{n_z \pi z}{L} + \frac{n_z \pi}{2}\right) \tag{1.1}$$

Here each PB function is labelled by $\mathbf{n} = (n_x, n_y, n_z)$, which is a set of 3 quantum numbers corresponding to each dimension x, y, z, which are allowed to be any positive integer value $(1, 2, 3, ...)$. The factor $\left(\frac{2}{L}\right)^{\frac{3}{2}}$ ensures that the wavefunction is normalized. As you can prove yourself, it is very convenient that the PB basis functions are naturally orthonormal:

$$S_{\mathbf{nn'}} = \int d\mathbf{r}\, \omega_{\mathbf{n}}^*(\mathbf{r}) \omega_{\mathbf{n'}}(\mathbf{r}) = \delta_{n_x n_x'} \delta_{n_y n_y'} \delta_{n_z n_z'} \tag{1.2}$$

Hence $\mathbf{S} = \mathbf{1}$ and so there is no need to orthogonalize the basis, unlike with contracted Gaussians!

In your code it may be convenient to form a list of the chosen PB functions, so that they can be labelled by a single index μ where each μ maps to its own particular set of quantum numbers: $\mu \leftrightarrow \mathbf{n}_\mu = (n_{\mu x}, n_{\mu y}, n_{\mu z})$. Since each μ has its own unique set of quantum numbers $\mathbf{n}_\mu$ we can then simply write:

$$S_{\mu\nu} = \delta_{\mu\nu} \tag{1.3}$$

These PB basis functions are suitable for describing a "molecule-in-a-box" because the wavefunction is guaranteed to vanish at the box boundary, so the entire electron density will be enclosed in the box. This is called the Dirichlet boundary condition, and it will ensure our program is well-behaved. These simple PB functions, which are just products of sines, are very closely related to plane waves because:

$$\sin\left(\frac{n_x \pi x}{L}\right) = \frac{1}{2i}\left[\exp\left(\frac{n_x \pi x}{L}\right) - \exp\left(-\frac{n_x \pi x}{L}\right)\right] \tag{1.4}$$

Thus our chosen PB basis functions are "linear combinations of plane-waves", with equal amplitudes for the left-going wave and the right-going wave in each of the three dimensions. Each PB basis function is a linear combination of 8 plane waves. Plane waves are widely used in condensed matter physics to model solids, using periodic boundary conditions (PBCs) where the orbitals should *not* vanish at the boundary.

How should we choose how many PB basis functions to use? The answer is we should set a maximum value (i.e. a cutoff) for the kinetic energy (KE) of our PB wavefunctions. Let us call this energy E_{cutoff}. Choosing E_{cutoff} sets the highest kinetic energy (and thus the smallest wavelength) PB functions that will be included in the calculations. This cut-off needs to be carefully chosen to balance computational cost and accuracy. It is quite easy to show that the matrix of the KE of the PB wavefunctions (in atomic units) is diagonal and given by:

$$T_{\mu\nu} = \int d\mathbf{r}\,\omega_\mu(\mathbf{r})\left(-\tfrac{1}{2}\nabla^2\right)\omega_\nu(\mathbf{r}) = \frac{\pi^2 \mathbf{n}_\nu^2}{2L^2}\delta_{\mu\nu} \tag{1.5}$$

Our chosen PB basis functions will then obey:

$$T_{\mu\mu} = \frac{\pi^2 \mathbf{n}^2}{2L^2} \leq E_{\text{cutoff}} \tag{1.6}$$

2 CONSTRUCTING THE INTEGRATION GRID IN REAL SPACE

In Density Functional Theory (DFT), the accurate calculation of electronic properties, such as total energy, electron density, and forces on atoms, necessitates the integration of various functions over the entire volume occupied by the electrons. Since analytic integration is typically impossible due to the fractional powers of the density in the exchange-correlation (XC) functional, numerical integration becomes essential. This process is facilitated by constructing an integration grid in real space, with points $\mathbf{r}_g$ and associated quadrature weights, w_g, to provide a discretized representation of the volume within which the electrons are found.

$$\int_V f(\mathbf{r})\,d\mathbf{r} \quad \rightarrow \quad \sum_g w_g f(\mathbf{r}_g) \tag{2.1}$$

In this exercise, we will develop a uniform grid with uniform weight along each dimension within the box. Specifically, if we allocate N_g equally spaced grid points along each dimension, the total number of grid points will amount to $N_G = N_g^3$. Consequently, the weight for each grid point is the same:

$$w = \frac{V}{N_G} \tag{2.2}$$

For example, as a test of how well your quadrature is working, you can check to see whether you can correctly integrate the electron density:

$$\int_V \rho(\mathbf{r}) \, d\mathbf{r} = n_{\text{elec}} \approx w \sum_g^{N_G} \rho(\mathbf{r}_g) \tag{2.3}$$

In this equation, we index the grid points g from 1 to $N_G = N_g^3$ and note that each grid point has its own coordinates $\mathbf{r}_g$.

3 INPUT AND SETUP

There are several quantities that must be specified:

- The molecular coordinates. The format for the molecule input file will remain consistent with the previous CNDO/2 homework assignments.

- The box size.

- The kinetic energy cutoff.

- The number of quadrature points along each direction.

4 KOHN-SHAM FOCK MATRIX AND THE ELECTRONIC ENERGY

As discussed in lecture, the Kohn-Sham equations can be expressed as:

$$\left(-\frac{1}{2}\nabla^2 + V_H(\mathbf{r}) + V_{xc}(\mathbf{r}) + V_{\text{ext}}(\mathbf{r}) \right) \phi_i(\mathbf{r}) = \epsilon_i \phi_i(\mathbf{r}) \tag{4.1}$$

where:

- $\phi_i(\mathbf{r})$ is the i-th occupied Kohn-Sham molecular orbital (MO).

- ϵ_i is the corresponding eigenvalue, often interpreted as the energy of the i-th Kohn-Sham MO.

- $V_H(\mathbf{r})$ is the Hartree potential, representing the classical electrostatic interaction between electrons. $V_{xc}(\mathbf{r})$ is the exchange-correlation potential, a quantum mechanical correction that includes both exchange and correlation effects among electrons.

- $V_{\text{ext}}(\mathbf{r})$ is the external potential, usually due to the nuclei for isolated molecules.

By representing the MOs as a linear combination of particle-in-a-box wavefunctions, we will optimize the MO coefficients to minimize the electronic energy. This approach mirrors the methodology applied in solving the CNDO/2 energy problem. Accordingly, we will solve the following eigenvalue problem:

$$\mathbf{F}^{\alpha}\mathbf{C}^{\alpha} = \mathbf{C}^{\alpha}\epsilon^{\alpha} \tag{4.2}$$

$$\mathbf{F}^{\beta}\mathbf{C}^{\beta} = \mathbf{C}^{\beta}\epsilon^{\beta} \tag{4.3}$$

Here:

- $\mathbf{F}^{\sigma}$ ($\sigma = \alpha$ or β) corresponds to the Kohn-Sham Fock matrix for electrons with spin σ.

- $\mathbf{C}^{\sigma}$ are the MO coefficients for electrons with spin σ.

- ϵ^{σ} is a diagonal matrix of MO eigenvalues.

Building $\mathbf{F}^{\sigma}$ is the major challenge, and it includes the matrix representation of each of the components of the Kohn-Sham Fock operator (eq. 4.1):

$$\mathbf{F}^{\sigma} = \mathbf{T} + \mathbf{V}_H + \mathbf{V}_{xc}^{\sigma} + \mathbf{V}_{\text{ext}} \tag{4.4}$$

Where:

- $\mathbf{T}$ is the matrix of the kinetic energy operator (see Eq. 1.5).

- $\mathbf{V}_H$ is the Hartree potential, which is the classical electrostatic potential due to the electron density

- $\mathbf{V}_{xc}^{\sigma}$ includes both exchange and correlation effects not captured by the Hartree potential.

- $\mathbf{V}_{\text{ext}}$ corresponds to the potential due to external forces, typically the attraction between electrons and nuclei.

For the potential energy terms, the matrix entries can be computed through grid integration:

$$V_{\mu\nu} = \int_V \omega_\mu(\mathbf{r}) \, V(\mathbf{r}) \, \omega_\nu(\mathbf{r}) \, d\mathbf{r} = w \sum_g^{N_G} \omega_\mu(\mathbf{r}_g) \, V(\mathbf{r}_g) \, \omega_\nu(\mathbf{r}_g) \tag{4.5}$$

where the total potential has 3 contributions:

$$V(\mathbf{r}) = V_H(\mathbf{r}) + V_{xc}(\mathbf{r}) + V_{\text{ext}}(\mathbf{r} \tag{4.6}$$

The external potential can be easily calculated on each grid point:

$$V_{\text{ext}}(\mathbf{r}_g) = \sum_A \frac{-Z_A}{|\mathbf{r}_g - \mathbf{R}_A|} \tag{4.7}$$

where Z_A and $\mathbf{R}_A$ correspond to the nuclear charge and the nuclear position respectively. However, note that there is a potential problem if a grid point is positioned on a nucleus! It

will be important to make sure that your code does not suffer from this problem. In fact this is one limitation of our approach which affects the rate of convergence of our results with N_G.

Turning to the exchange-correlation term, we will keep only the exchange term of the LSDA functional for V_{xc}^σ for simplicity (that's what defines the X_α method). Therefore, $V_{xc}^\sigma(\mathbf{r})$ can also be fairly easily calculated on each grid point:

$$V_{xc}^\sigma(\mathbf{r}_g) = V_x^{\text{LDA}}[\rho^\sigma](\mathbf{r}_g) = -\left(\frac{3}{\pi}\right)^{\frac{1}{3}} \rho^\sigma(\mathbf{r}_g)^{\frac{1}{3}} \tag{4.8}$$

Here the electron spin $\sigma = \alpha$ or β, and $\rho^\sigma(\mathbf{r}_g)$ is the electron density of the corresponding spin at that grid point.

The last of the three potential contributions is the Hartree potential:

$$V_H(\mathbf{r}) = \int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}\, d\mathbf{r}' \tag{4.9}$$

$V_H(\mathbf{r}_g)$ can be obtained by solving the Poisson equation (a second order elliptic partial differential equation in 3 variables) with Dirichlet boundary conditions (i.e. the potential becomes zero at the boundary). The Poisson equation is:

$$\nabla^2 V_H(\mathbf{r}) = -4\pi\rho(\mathbf{r}) \tag{4.10}$$

Here, $\rho(\mathbf{r}) = \rho^\alpha(\mathbf{r}) + \rho^\beta(\mathbf{r})$.

You will use a finite difference method to solve the Poisson equation, where we write:

$$\nabla^2 V_{i,j,k} \approx \frac{V_{i+1,j,k} + V_{i-1,j,k} + V_{i,j+1,k} + V_{i,j-1,k} + V_{i,j,k+1} + V_{i,j,k-1} - 6V_{i,j,k}}{h^2} \tag{4.11}$$

The discretized Poisson equation is therefore:

$$\frac{V_{i+1,j,k} + V_{i-1,j,k} + V_{i,j+1,k} + V_{i,j-1,k} + V_{i,j,k+1} + V_{i,j,k-1} - 6V_{i,j,k}}{h^2} = -4\pi\rho_{i,j,k} \tag{4.12}$$

Here h is the spacing of the grid points, and $V_{i,j,k}$ and $\rho_{i,j,k}$ are the Hartree potential and electronic density on each grid point respectively. The discretized Poisson equation, eq. 4.12 is a set of linear equations, where $\rho_{i,j,k}$ is the known right-hand side, and $V_{i,j,k}$ are the unknowns.

There are many ways to solve the discretized Poisson equation. We should certainly not do it directly because the coefficient matrix is both enormous ($N_G \times N_G$) and enormously sparse (only about $7N_G$ non-zeros). So we should solve it iteratively, and amongst the simplest iterative solver is Jacobi iteration with Dirichlet boundary condition (that is to say, $V_{i,j,k} = 0$ for grid points on the boundary) to solve for $V_H(\mathbf{r}_g)$. This iteration uses the diagonal contribution for the update on the $(n+1)$-st iteration, starting from a zero initial guess ($V_{i,j,k}^{(0)} = 0$).

$$V_{i,j,k}^{(n+1)} = \frac{1}{6}\left(V_{i+1,j,k}^{(n)} + V_{i-1,j,k}^{(n)} + V_{i,j+1,k}^{(n)} + V_{i,j-1,k}^{(n)} + V_{i,j,k+1}^{(n)} + V_{i,j,k-1}^{(n)} + 4\pi h^2 \rho_{i,j,k}\right) \tag{4.13}$$

Stop your Poisson iterations when the maximum change in $V_{i,j,k}$ is suitably small, and thus your Hartree potential has converged.

After assembling the Kohn-Sham Fock matrix, we can directly solve the eigenvalue problem to obtain the updated molecular coefficients and the corresponding electron density:

$$\rho^\alpha(\mathbf{r}) = \sum_i^{\text{occ}_\alpha} |\phi_i^\alpha(\mathbf{r})|^2 = \sum_i^{\text{occ}_\alpha} \sum_\mu^{\text{AO}} |C_{\mu i}^\alpha|^2 |\omega_\mu(\mathbf{r})|^2 \tag{4.14}$$

$$\rho^\beta(\mathbf{r}) = \sum_i^{\text{occ}_\beta} |\phi_i^\beta(\mathbf{r})|^2 = \sum_i^{\text{occ}_\beta} \sum_\mu^{\text{AO}} |C_{\mu i}^\beta|^2 |\omega_\mu(\mathbf{r})|^2 \tag{4.15}$$

Here, occ_α and occ_β correspond to the number of electrons with α and β spin respectively. Given that the Fock matrix depends on the electron density, we must solve the eigenproblem through a self-consistent approach.

After the density/energy converge (oh, what a happy day!), we can calculate the total energy as:

$$E_{\text{total}} = \text{Tr}(\mathbf{PT}) + \text{Tr}(\mathbf{PV}_{\text{ext}}) + \frac{1}{2}\text{Tr}(\mathbf{PV}_{\mathbf{H}}) + E_{xc}^{\text{tot}} \tag{4.16}$$

where

$$E_{xc}^{\text{tot}} = E_{\text{exchange}}^{LDA}[\rho^\alpha] + E_{\text{exchange}}^{LDA}[\rho^\beta] = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{\frac{1}{3}} \int \left[\rho^\alpha(\mathbf{r})^{\frac{4}{3}} + \rho^\beta(\mathbf{r})^{\frac{4}{3}}\right] d\mathbf{r} \tag{4.17}$$

$$\mathbf{P} = \mathbf{P}^\alpha + \mathbf{P}^\beta \tag{4.18}$$

$$P_{\mu\nu}^\alpha = \sum_i^{\text{occ}_\alpha} C_{\mu i}^\alpha C_{\nu i}^\alpha \tag{4.19}$$

$$P_{\mu\nu}^\beta = \sum_i^{\text{occ}_\beta} C_{\mu i}^\beta C_{\nu i}^\beta \tag{4.20}$$

Therefore, the whole process will be

1. Input Processing: Read in the molecular information from sample inputs. Build the basis set by setting a suitable cutoff energy. Generate a discrete grid within the cubic box to perform numerical integration accurately. Please make sure there are no grid points on atoms!

2. Make an initial guess for the electron density distribution or the density matrix for different spins. Zero can work!

3. Assemble the Kohn-Sham Fock matrix $\mathbf{F}^\alpha$ and $\mathbf{F}^\beta$, incorporating contributions from the kinetic energy, Hartree potential, external potential, and the LDA exchange-correlation functional. This requires solving the Poisson equation on each iteration.

4. Solve the Kohn-Sham equations, $\mathbf{FC} = \mathbf{C}\epsilon$ for each spin, to obtain new electron density and energy values.

5. Evaluate the convergence of the energy and density. If the system has not yet converged to the desired accuracy, return to step 3 and iterate.

6. After convergence, calculate the electronic energy of the system.

In this problem set, we ask you to

1. Implement your own DFT code using particle in a box basis functions!

2. Debug this code in easy stages, starting from the hydrogen atom, for which it is fine to set $V(\mathbf{r}) = V_{\text{ext}}(\mathbf{r})$ and assume $V_H(\mathbf{r}) = -V_{xc}(\mathbf{r})$ so you do not need to have your Poisson solver working yet. Note that this is in fact a Hartree-Fock code, at least for this system! Your GSI will provide sample outputs for hydrogen and helium. Note for these cases (and H$^-$, Be etc) when an atom is at the center of the box and only s-type orbitals are occupied, it is fine to include only odd values of n_x, n_y, n_z because the occupied orbitals are all centrosymmetric. This reduces your number of basis functions by another factor of 8 as well as helping convergence!

3. Plot the occupied orbitals along the x, y, and z axis of some simple systems such as He and the stretched hydrogen molecule. You can change the parameters (cutoff energy, box size, and number of grid points in each dimension) and compare the plots. Discuss what you find about these plots.

4. Feel free to imagine or experiment with other improvements or changes to the code!