DFT on an Atom

Jasnoor Singh[1]
Undergraduate Department, Indian Institute of Science, Bangalore.
(*Electronic mail: jasnoorsingh@iisc.ac.in)

(Dated: 18 April 2024)

I. INTRODUCTION

The Schrödinger wave equation, $\hat{H}|\Psi\rangle = E|\Psi\rangle$, represents a foundational achievement in quantum mechanics, however, its practical application is confined predominantly to simple systems due to the computational challenges posed by the many-body problem. The introduction of the Slater determinant and the mean-field approximation through the Hartree–Fock (HF) method somewhat reduced these computational burdens while maintaining an ab initio approach, but, HF methods are generally applicable only to small systems comprising a few tens of atoms. I will discuss the core ideas and operational principles of DFT and how it reduces the problem of finding ground state energy.

In this term paper I focus on the application of DFT to single atoms by employing the Local Density Approximation (LDA) for the exchange correlational functional. I will detail the implementation specifics and the resultant computations, including all the energy components. The results shall be compared with benchmark database maintained by NIST[1]. These ab initio calculations not only underscore DFT's robustness but also help in the generation of effective pseudopotentials for each atom, which are used when extending DFT to more complex systems.

II. THEORY[2][3][4]

A. Electron Density

The solution to Schrodinger equation $\hat{H}_e|\Psi_e\rangle = E_e|\Psi_e\rangle$ is an N-electron multibody wavefunction which depends on 3N position coordinates and N spin coordinates. Even though it is possible to find any observable using this multibody wavefunction, most operators depend on coordinates of only one or two electrons. Therefore calculating the multibody wavefunction is an overkill for most problems. Instead the problem is simplified by calculating total density of electrons, which is a function of just 3 coordinates instead of 3N coordinates. The 2 fairly simple Hohenberg and Kohn theorems show how it is possible to calculate any ground state property using electron density directly. The proofs can be seen in any computational quantum chemistry textbook:

- Theorem 1: For a given density of the ground state of a particular system, we can not have two different external potentials V_{ext}, i.e. that part of Hamiltonian is uniquely specified. Hence, any ground-state property is a functional of electron density.
- Theorem 2: It states that among all the possible electron densities, the density corresponding to the ground state is the one for which energy is minimized as a functional of density.

Therefore we obtain a variational approach to solve the problem of ground state energy by varying the densities. Applying the variational principle we get the following equations where μ is the lagrange multiplier:

$$\delta\left\{E_e[\rho(\mathbf{r})] - \mu\left[\int \rho(\mathbf{r})d\mathbf{r} - N\right]\right\} = 0,$$

$$E_e = T[\rho(\mathbf{r})] + \int V_{\text{ext}}(\mathbf{r})\rho(\mathbf{r})d\mathbf{r} + \int V_c(\mathbf{r})\rho(\mathbf{r})d\mathbf{r} + E'_{xc}[\rho(\mathbf{r})],$$

$$\frac{\delta E_e}{\delta\rho(\mathbf{r})} = \frac{\delta T}{\delta\rho(\mathbf{r})} + \frac{\delta}{\delta\rho(\mathbf{r})}\int V_{\text{ext}}(\mathbf{r}')\rho(\mathbf{r}')d\mathbf{r}' + \frac{1}{2}\int\int \frac{\rho(\mathbf{r}_1)\rho(\mathbf{r}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|}d\mathbf{r}_1 d\mathbf{r}_2 + \frac{\delta E_{xc}}{\delta\rho(\mathbf{r})},$$

$$\frac{\delta T}{\delta\rho} + V_{\text{ext}}(\mathbf{r}) + V_C(\mathbf{r}) + \frac{\delta E'_{xc}}{\delta\rho} = \mu,$$

B. Kohn-Sham Approach

The Hohenberg-Kohn theorems provided proof for correctness, but do not explain how this variational approach has to be applied to get ground-state properties from electron density. To solve this, the trick used by Kohn and Sham is to consider a fictitious, auxillary system of non-interacting particles which have the same density and energy as the orignal system.

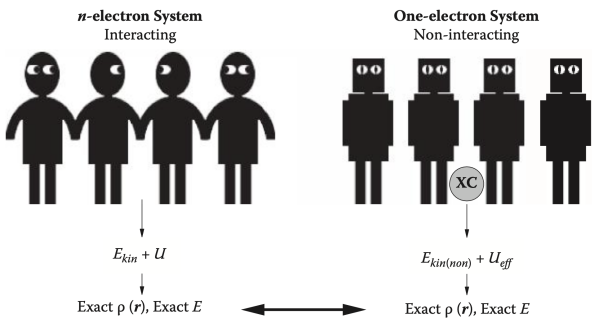

FIG. 1. Kohn-Sham Ansatz[2]

Since the particles in this system are non-interacting, the total energy expression becomes:

$$E_e = T_0[\rho(\mathbf{r})] + \int V_{\text{eff}}(\mathbf{r})\rho(\mathbf{r})d\mathbf{r},$$

where V_{eff} is given by:

$$V_{\text{eff}}(\mathbf{r}) = \frac{\delta T}{\delta \rho} - \frac{\delta T_0}{\delta \rho} + V_{\text{ext}}(\mathbf{r}) + V_C(\mathbf{r}) + \frac{\delta E'_{xc}}{\delta \rho},$$

We don't know the exact form of this effective potential, specifically the terms which are absorbed into E_{xc}:

$$\frac{\delta T}{\delta \rho} - \frac{\delta T_0}{\delta \rho} + \frac{\delta E'_{xc}}{\delta \rho} = \frac{\delta E_{xc}}{\delta \rho} = V_{xc}(\mathbf{r}),$$

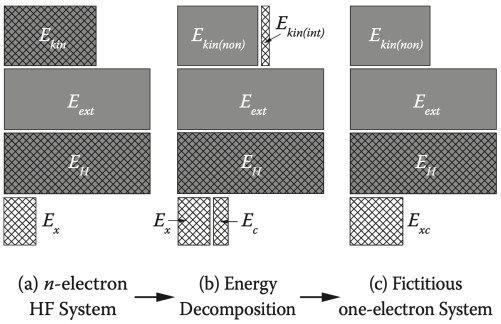

FIG. 2. Schematic of DFT approach[2]

With this approach, we solve the Kohn-Sham equations:

$$\left[\hat{h} + V_C(\mathbf{r}) + V_{xc}(\mathbf{r})\right]\phi_i(\mathbf{r}) = \varepsilon_i\phi_i(\mathbf{r}),$$

where Total Energy is given by:

$$E = \sum_{i=1}^{N}\langle\phi_i|\hat{h}|\phi_i\rangle + \frac{1}{2}\int V_C(\mathbf{r})\rho(\mathbf{r})d\mathbf{r} + E_{xc}[\rho(\mathbf{r})].$$

C. Approximations for E_{xc}

Our problem is reduced to finding the expression of E_{xc} as a functional of density. The exact expression has not been found, so we have to make approximation. The simplest approximation that can be applied is Local Density Approximation (LDA), where we use the potential V_{xc} at a point $\mathbf{r}$ as it would be for a homogenous gas with density $\rho(\mathbf{r})$. This approximation ignores the spatial variation in density. The LDA exchange-correlation functional of Vosko, Wilk, and Nusair (VWN)[5] has been used in the calculations below. As a next step, the dependencies on spatial variations can be introduced by making V_{xc} depend upon: $\vec{\nabla}\mathbf{r}$, $\nabla^2\mathbf{r}$ and so on. Finding such exchange-correlational functionals is a key area of research in DFT.

D. Self Consistency Field

Once we know V_{xc} (or approximately), the Kohn-Sham equations cannot be solved like Schrödinger equation, because the effective potential V_{ext} itself depends on the density, which itself depends on the solutions of the Kohn-Sham equations. Therefore this has to be solved self-consistently, meaning we have to find an effective potential, which leads to a charge density creating the same effective potential. This is done by an iterative approach. Initially the Hartree, and exchange correlation potential are initialised as 0 everywhere, then the Kohn-sham equations are solved to obtain eigenvalues and eigenfunctions, which are used to find the effective potential. These potentials are substituted back in the equations to solve again. The process is iterated until the total energy doesn't change above a threshold. This process involves a mixing parameter β: $V_{ext} = \beta V_{ext}^{new} + (1-\beta)V_{ext}^{old}$, which denotes the amount of mixing between the old potential and the new potential. $\beta = 1$ corresponds to no mixing, and old potential is discarded. Higher β can leads to unstable energies which never converge, because the potentials keep oscillating without approaching the correct value.

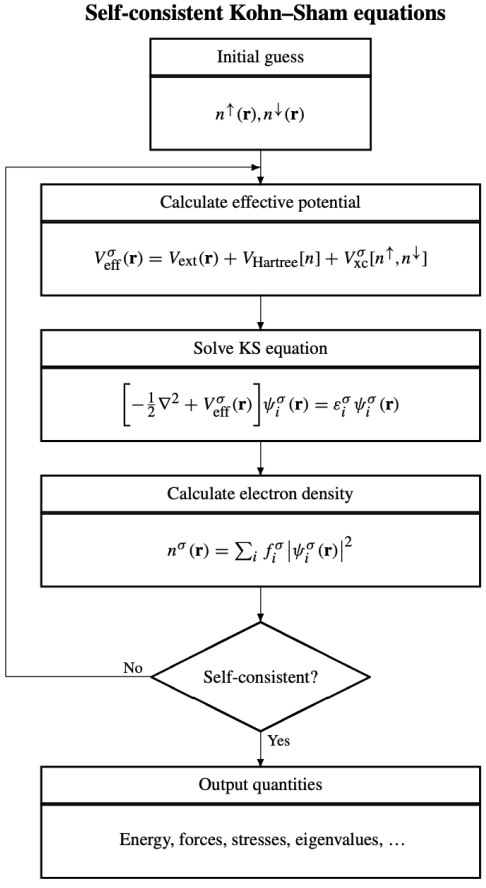

FIG. 3. Schematic representation of the self-consistent loop for solutions of the Kohn-Sham equations[4]

III. PROCEDURE AND IMPLEMENTATION[6]

A. Procedure

The calculations are carried out using Kohn-Sham approach to DFT. I have done calculations on the first 12 elements of the periodic table, and their respective cations. In case of non spherically symmetric subshells ($m \neq 0$), spherical averaging of orbitals is done to calculate hartree, exchange and correlation energies. In DFT, we solve the Kohn-sham equations:

$$\left[-\frac{1}{2}\nabla^2 + v(\mathbf{r})\right]\psi_i(\mathbf{r}) = \varepsilon_i\psi_i(\mathbf{r}),$$

where ψ_i are Kohn-Sham orbitals in our auxillary system and,

$$v(\mathbf{r}) = v_{\text{ext}}(\mathbf{r}) + v_{\text{h}}(\mathbf{r}) + v_{\text{xc}}(\mathbf{r}),$$

here $v_{\text{ext}}(\mathbf{r})$ is the external (nuclear) potential, $v_{\text{h}}(\mathbf{r})$ is the hartree potential, and $v_{\text{xc}}(\mathbf{r})$ is the exchange-correlation potential. The components of total energy are calculated as follows:

$$T = -\frac{1}{2}\sum_i \int d\mathbf{r}\,\psi_i^*(\mathbf{r})\nabla^2\psi_i(\mathbf{r}),$$

$$E_{\text{enuc}} = \int d\mathbf{r}\,\rho(\mathbf{r})v_{\text{nuc}}(\mathbf{r}),$$

$$E_{\text{coul}} = \frac{1}{2}\int d\mathbf{r}\int d\mathbf{r}'\,\frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|},$$

$$E_{\text{xc}} = \int d\mathbf{r}\,\rho(\mathbf{r})\varepsilon_{\text{xc}}(\rho),$$

I have used exchange-correlation energy functional of Vosko, Wilk, and Nusair (VWN)[5]. The exchange energy which is a functional of electron density is calculated as follows, where $n_\uparrow$ and $n_\downarrow$ are upspin and downspin electron densities and n is the total electron density :

$$r_s = \left(\frac{3}{4\pi n}\right)^{1/3},$$

$$\xi = \frac{(n_\uparrow - n_\downarrow)}{n},$$

$$\varepsilon_{\text{xc}}^{\text{P}}(r_s) = 2^{-1/3}\varepsilon_{\text{xc}}^{\text{F}}(r_s) = -3\left(\frac{9}{32\pi^2}\right)^{1/3} r_s^{-1},$$

Similarly the correlation energy is calculated as follows:

$$F(r_s) = A\left[\ln\left(\frac{x^2}{X(x)}\right) + \frac{2b}{Q}\tan^{-1}\left(\frac{Q}{2x+b}\right) - \frac{bx_0}{X(x_0)} \times \ln\left(\frac{(x-x_0)^2}{X(x)}\right) + \frac{2(b+2x_0)}{Q}\tan^{-1}\left(\frac{Q}{2x+b}\right)\right],$$

$$f(\xi) = \frac{(1+\xi)^{4/3} + (1-\xi)^{4/3} - 2}{2(2^{1/3}-1)},$$

$$\varepsilon_c(r_s,\xi) = \left[1 - f(\xi)\varepsilon_c^4\right]F_p + f(\xi)(1-\xi^4)F_a/f''(0) + f(\xi)\varepsilon_c^4 F_f,$$

where: $x = r_s^{1/2}$, $X(x) = x^2 + bx + c$, $Q = (4c-b^2)^{1/2}$ and three instances of F are created using the parameters as follows in Table I.

TABLE I. Parameters for VWN functional

Parameter	Paramagnetic F_p	Ferromagnetic F_f	Spin stiffness F_a
A	0.0310907	0.01554535	$-1/(6\pi^2)$
x_0	−0.10498	−0.325	−0.0047584
b	3.72744	7.06042	1.13107
c	12.9352	18.0578	13.0045

The exchange-correlation potential is obtained using the relation:

$$V_{\text{xc}}(n) = \frac{dn(\varepsilon_x + \varepsilon_c)}{dn}.$$

B. Implementation[7]

Since the atom is spherically symmetric, the problem of solving Kohn-Sham equations is reduced to solving the radial part. Therefore, a suitable choice of radial grid is needed. I choose an exponentially increasing grid, since the wavefunction changes much faster near the nucleus as compared to further away. The n^{th} point on this grid is given by the relation:

$$r_n = r_{\min}\left(\frac{r_{\max}}{r_{\min}}\right)^{\frac{n}{N}},$$

The parameters chosen were: $N = 15788$, $r_{min} = 10^{-6}$, $r_{max} = 50$, since these were the parameters used to find energies in the NIST Atomic Reference Database.

Considering only the radial part of Kohn-Sham orbitals, the Kohn-Sham equations reduce to:

$$-\frac{1}{2}\frac{d^2u_i}{dr^2} + \left[v(r) + \frac{\ell(\ell+1)}{2r^2}\right]u_i = \varepsilon u_i,$$

where l is the angular momentum quantum number and $u(r) = rR(r)$. Calculation of hartree potential is reduced to solving the poisson equation in this case due to spherical symmetry, with the boundary conditions that $U(0) = 0$ and $U(r_{max}) = q_{max}$, where q_{max} is the total charge of electrons:

$$v_h''(r) = -\frac{u^2(r)}{r}.$$

The exchange correlation potentials are simply calculated using the formulae mentioned before where, $n(r) = \frac{u^2}{4\pi r^2}$. Using the above parameterisation, calculation of energies are simplified as:

$$E_h = \frac{1}{2}\int V_h(r)u^2(r)\,dr$$

$$E_{xc} = \int \varepsilon_{xc}(r) u^2(r)\, dr$$

$$E_{\text{ext}} = \int -Z u^2(r) r\, dr$$

$$E_{kin} = \frac{1}{2} \int \frac{d^2u(r)}{dr^2} u(r)\, dr + \frac{l(l+1)}{2} \int \frac{u^2(r)}{r^2}\, dr$$

The Kohn-Sham equations are solved using self-consistent approach. Initially the hartree, and exchange correlation potentials are initialised as 0 everywhere, then the Kohn-sham equations are solved to obtain eigenvalues and eigenfunctions, which are used to find the hartree and exchange correlation potential. These potentials are substituted back in the equation to solve again. The process is iterated until the total energy doesn't change above a threshold. This process involves a mixing parameter β, which denotes the amount of mixing between old potential and the new potential. $\beta = 1$ corresponds to no mixing, and old potential is discarded. $\beta = 1$ sometimes leads to unstable energies which never converge, because the potentials keep oscillating without approaching the correct value. I used $\beta = 0.5$ when running the program, which ensured convergance within 25 iterations.

C. Routines

Here is the explanation of snippets of the main routines used in the code. The following code is used to find the eigenvalue of the Kohn-Sham equation, between E_{min} and E_{max} using bracketing method along with a differential equation solving routine. The differential equation is solved from $r = r_{max}$ to $r = 0$ with initial condition $u(r_{max}) = 0$, and the value of E for which $u(0) = 0$ is returned, since it satisfies the boundary conditions.

```
def find_eigenvalue(self, l, E_min, E_max,
    precision):
    find_eigenroot = lambda E: self.
    differential_solver(E, l)[-1]
    n = 4*self.Z
    alpha = (E_min-E_max)/n**2
    arr = [alpha*i**2+E_max for i in np.arange(n
    , 0, -1)]
    for i in range(len(arr)-1):
        try:
            a = find_eigenroot(arr[i])
            b = find_eigenroot(arr[i+1])
            if a*b<=0:
                E = toms748(find_eigenroot, arr[
    i], arr[i+1], xtol=precision)
                Y = self.differential_solver(E,
    l)
                return Y[0][::-1], E
        except ValueError:
            continue
    return None
```

The next code routine is used to find the energies associated with all the filled Kohn-Sham orbitals, and the resultant density of up-spin and down-spin electrons. The algorithm automatically adjusts the value of E_{min} and E_{max} to be used for calculating the energy eigenvalues of the orbitals. For instance, the energy of 2s orbital should be more than 1s, so the E_{min} for 2s is kept as a little more than the energy eigenvalue of 1s orbital. Similarly E_{min} for 2p is kept slightly more than energy of 2s.

```
def find_e_nup_ndown_T(self, Vh, Vx, Vc):
    n_up = np.zeros_like(self.r)
    n_down = np.zeros_like(self.r)
    eigenvalues = dict()
    T = 0
    E_e = 0
    for n, l, up, down in self.table:
        occupancy = up+down
        a = orbital_eigenvalue_solver(self.Z,
    self.r, Vh, Vx, Vc, self.qmax)
        if (n-l)==1:
            if (n==1 and l==0):
                u, epsilon = a.find_eigenvalue(l
    , self.E_min-0.001, self.E_max, self.tol)
            else:
                emin = eigenvalues[f"{n} {l-1}"]
                u, epsilon = a.find_eigenvalue(l
    , emin-0.001, self.E_max, self.tol)
        else:
            emin = eigenvalues[f"{n-1} {l}"]
            u, epsilon = a.find_eigenvalue(l,
    emin+0.001, self.E_max, self.tol)
        T += -occupancy*simpson((
    double_derivative(u,self.r)*u), self.r)/2
        if l!=0:
            T += occupancy*l*(l+1)*simpson(u**2/
    self.r**2, self.r)/2
        eigenvalues[f"{n} {l}"] = epsilon
        n_up += up*u**2/self.r**2/4/np.pi
        E_e += occupancy*epsilon
        n_down += down*u**2/self.r**2/4/np.pi

    return eigenvalues, n_up, n_down, T, E_e
```

The following code is used to find the exchange-correlation potential and energies. The calculation of correlation functional is computationally intensive since it is done by substituting in a very long sympy expression and takes time, however it can be done parallely by multiple cores, since the array containing the densities can be split and assigned to different cores to calculate the potential. The code sets up a routine for carrying out this parallel computation.

```
def setup_parallel_computation(expression):
    def parallel_computation(n, digits):
        num_cores = multiprocessing.cpu_count()
        num_processes = max(1, num_cores-1)
        chunk_size = int(np.ceil(len(n) /
    num_processes))

        split_arrays = [n[i:i + chunk_size] for
    i in range(0, len(n), chunk_size)]

        results = []
        with ProcessPoolExecutor(max_workers=
    num_processes) as executor:
            future_results = [executor.submit(
    process_chunk, chunk, expression, digits)
    for chunk in split_arrays]

            for future in future_results:
                results.extend(future.result())

        return np.array(results)
    return parallel_computation
```

The next routine uses the parallel computation function defined above to calculate the correlation potential and energy. The function returns the value using the formula defined in the procedure section. Similar function is defined for exchange potential and energies as well.

```
def find_ec_Vc(self):
    n = self.n
    Vcp = np.nan_to_num(parallel_compute_vcp(n, self.digits))
    ecp = np.nan_to_num(parallel_compute_ecp(n, self.digits))
    if np.all(self.z)==0 or self.xc.lower() == "lda":
        return ecp, Vcp
    if self.xc.lower()=="lsd":
        Vcf = np.nan_to_num(parallel_compute_vcf(n, self.digits))
        ecf = np.nan_to_num(parallel_compute_ecf(n, self.digits))
        Vca = np.nan_to_num(parallel_compute_vca(n, self.digits))
        eca = np.nan_to_num(parallel_compute_eca(n, self.digits))
        Vc = Vcp*(1-self.fz*self.z4) + self.fz*(1-self.z4)*Vca/fpp_0 + self.fz*self.z4*Vcf
        ec = ecp*(1-self.fz*self.z4) + self.fz*(1-self.z4)*eca/fpp_0 + self.fz*self.z4*ecf
        return ec, Vc
    return None
```

The following code is the main code of the dft solver class which solves the Kohn-Sham equations self-consistently. Initially the hartree, and exchange correlation potentials are initialised as 0 everywhere, then the Kohn-sham equations are solved to obtain eigenvalues and eigenfunctions, which are used to find the hartree and exchange correlation potential. These potentials are substituted back in the equation to solve again. The values of all the component energies are printed each iteration so that the convergence of these values can be seen.

```
    def solve(self, iterations):
        a, b, E_new, E_e, E_h, E_x, E_c, E_enuc, T, E = np.zeros(10)
        Vh = np.zeros_like(self.r)
        Vx = np.zeros_like(self.r)
        Vc = np.zeros_like(self.r)
        beta_arr=np.linspace(self.beta, 1, iterations)
        for i in range(iterations):
            beta = beta_arr[i]
            print(f"Iteration {i}")
            print("Energies:")
            print("E_tot: {:.6f},E_kin: {:.6f}, E_coul: {:.6f},E_enuc: {:.6f},E_xc: {:.6f},e: {:.6f}".replace(",", "\n").format(E_new, T, E_h, E_enuc, E_x+E_c, E_e))
            E = E_new
            eigenvalues, n_up, n_down, T, E_e = self.find_e_nup_ndown_T(Vh, Vx, Vc)
            n = n_up+n_down
            print("Eigenvalues:")
            tablel={"0": "s", "1": "p", "2": "d", "3": "f"}
            for key, value in eigenvalues.items():
                l = tablel[key[-1]]
                print("{:1}{:1}: {:.6f}". format(key[0], l, value))
            b = poisson_solver(self.r, n_up, n_down, self.qmax)
            xc = xc_calculator(n_up, n_down, self.tol, self.xc)
            Vh = beta*b.find_Vh() + Vh*(1-beta)
            ex, Vxn = xc.find_ex_Vx()
            ec, Vcn = xc.find_ec_Vc()
            Vx = Vxn*beta+(1-beta)*Vx
            Vc = Vcn*beta+(1-beta)*Vc
            E_h = 4*np.pi*simpson(Vh*n*self.r**2, self.r)/2
            E_x = 4*np.pi*simpson(ex*n*self.r**2, self.r)
            E_c = 4*np.pi*simpson(ec*n*self.r**2, self.r)
            E_enuc = 4*np.pi*simpson(-self.Z*n*self.r, self.r)
            E_new = T+E_enuc+E_h+E_x+E_c
        print("Converged values:")
        print("E_tot: {:.6f},E_kin: {:.6f}, E_coul: {:.6f},E_enuc: {:.6f},E_xc: {:.6f},e: {:.6f}".replace(",", "\n").format(E_new, T, E_h, E_enuc, E_x+E_c, E_e))
        print("\n")
        return  E_new
```

Sample main.py file to run DFT calculation on helium atom. "1s": "1 1" denotes 1 electron each in up spin and down spin in 1s orbital. N is the number of points in the radial grid, Z is the atomic number, beta is the mixing parameter, xc is the exchange-correlation functional used.

```
from dft import dft_solver
if __name__=="__main__":
    ec = {"1s": "1 1"}
    He = dft_solver(ec, N = 15788, Z=2, beta=1/2, xc="LDA-VWN", r_min=1e-6, r_max=50, tol=1e-6)
    He.solve(20)
```

IV. RESULTS

A. Energies

The energies are calculated for the first the first 12 element atoms from H to Mg and their singly positive cations. These energy values are saved in .xlsx file attached. The energies calculated for elements He, B, and Na by my code are shown in the table below. The table also compares the standard reference values maintained by NIST[1] which uses the same exchange correlation functional. We can see that the total energy values is accurate upto $1 \mu hartrees$, and the individual component energies are accurate upto $8 \mu hartrees$, which is the standard set by the NIST in the four independent codes they used for this calculation. The error is a little higher in case of B and Na, because the value did not converge upto $1 \mu hartrees$ in the fixed number of iterations. The various components of total energy with their absolute value contribution is also shown as a pie chart for the case of Helium. We can see that the exchange correlation energy has the lowest contribution, but we still need to calculate it to know the physics. This is

true for complex systems as well, where the exchange correlation energy is the one which determines the physics of a material. The trend of total energies as a function of atomic number from H to Mg is also shown.

Energies	Code	Reference
E_{tot}	−2.834 836	−2.834 836
E_{kin}	2.767 924	2.767 922
E_{coul}	1.996 121	1.996 120
E_{enuc}	−6.625 567	−6.625 564
E_{xc}	−0.973 314	−0.973 314
1s	−0.570 424	−0.570 425

TABLE II. He

Energies	Code	Reference
E_{tot}	−24.344 201	−24.344 198
E_{kin}	24.161 045	24.161 047
E_{coul}	11.503 005	11.503 002
E_{enuc}	−56.484 592	−56.484 587
E_{xc}	−3.523 660	−3.523 660
1s	−6.564 346	−6.564 347
2s	−0.344 699	−0.344 701
2p	−0.136 601	−0.136 603

TABLE III. B

Energies	Code	Reference
E_{tot}	−161.440 062	−161.440 060
E_{kin}	160.896 489	160.896 571
E_{coul}	79.732 481	79.732 497
E_{enuc}	−388.546 280	−388.546 372
E_{xc}	−13.522 751	−13.522 757
1s	−37.719 969	−37.719 975
2s	−2.063 393	−2.063 401
2p	−1.060 629	−1.060 636
3s	−0.103 413	−0.103 415

TABLE IV. Na

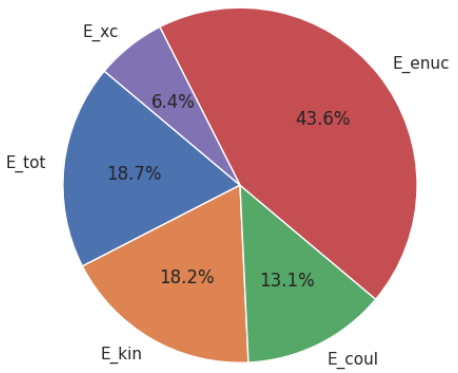

FIG. 4. Components of total energy in He

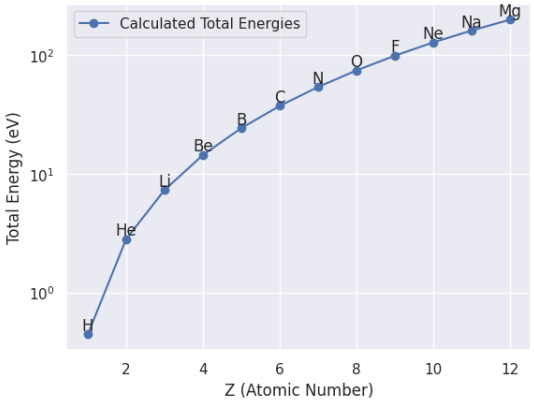

FIG. 5. Trend of Total Energies

B. Ionisation Energies

The energies of cations calculated for atoms He+, B+, and Na+ by my code are shown in the table below. The table also compares the standard reference values maintained by NIST[1] which uses the same exchange correlation functional.

Energies	Code	Reference
E_{tot}	−1.861 238	−1.861 237
E_{kin}	1.828 738	1.828 737
E_{coul}	0.592 267	0.592 268 0
E_{enuc}	−3.823 854	−3.823 853
E_{xc}	−0.458 390	−0.458 389
1s	−1.410 933	−1.410 933

TABLE V. He+

Energies	Code	Reference
E_{tot}	−24.038 280	−24.038 275
E_{kin}	23.877 682	23.877 686
E_{coul}	9.557 871	9.557 870
E_{enuc}	−54.209 500	−54.209 497
E_{xc}	−3.264 334	−3.264 334
1s	−7.044 295	−7.044 297
2s	−0.713 194	−0.713 196

TABLE VI. B+

Energies	Code	Reference
E_{tot}	−161.250 340	−161.250 337
E_{kin}	160.718 955	160.718 894
E_{coul}	76.774 839	76.774 852
E_{enuc}	−385.344 451	−385.344 403
E_{xc}	−13.399 684	−13.399 680
1s	−38.005 004	−38.004 998
2s	−2.347 376	−2.347 368
2p	−1.343 362	−1.343 353

TABLE VII. Na+

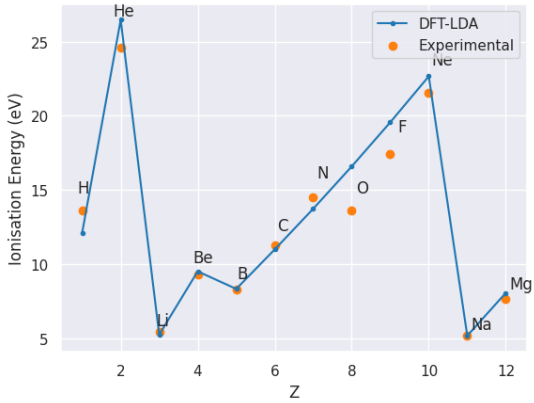

FIG. 6. Trend of Ionisation Energies

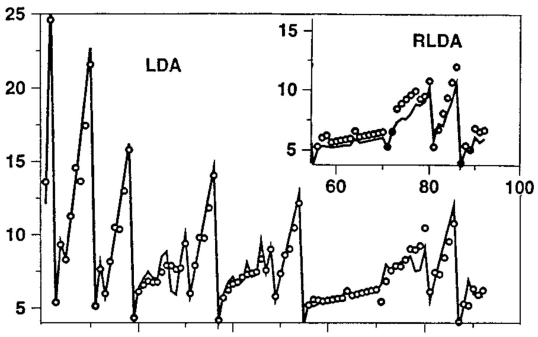

FIG. 7. Ionisation Energies for first 95 elements reference[5]

Above figure shows the calculated as well as experimental[8] ionisation energies of elements as function of atomic number. The ionisation energy is determined by subtracting the energy of neutral atom from the energy of the cation. As expected, the ionisation energy peaks at noble gas, followed by a sharp dip at alkali metal. The ionisation energy experiment values seem to be predicted very well by DFT, with an exception of Oxygen, which has considerable deviation (20%) from the actual experimental value. We can in principle, also calculate the energies of anion, but such a calculation doesn't have a standard database maintained by NIST. However the electron gain enthalpies obtained by it, can be verified by the experimental values.

C. Miscellaneous

The trend of eigenvalue energies of various orbitals can also be observed from our data.

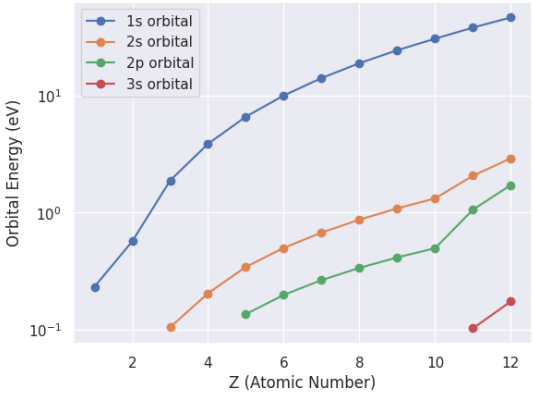

FIG. 8. Trend of Eigenvalue Energies of Orbitals

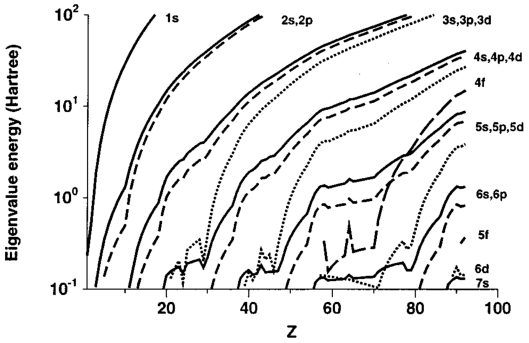

FIG. 9. Eigenvalue Energies of Orbitals reference[5]

V. APPENDIXES

Appendix A: Code

The code written for finding the energies is available at: Github Repository

Appendix B: Data

The benchmark atomic reference data published by NIST[1] can be found here: Atomic Reference Data. The data generated by the code is stored in an excel sheet: Sheet

Appendix C: Graphs

The jupyter notebook used for generating the graphs is here: Jupyter Notebook

Appendix D: Credits

I thank Aman Goyal (amangoyal@iisc.ac.in) for helping me in debugging the code and simplifying equations.

[1]"Atomic reference data for electronic structure calculations," .
[2]J. G. Lee, *Computational Materials Science: An Introduction*.
[3]M. Springborg, *Methods of Electronic-structure Calculations*.
[4]R. M. Martin, *Electronic Structure*.
[5]S. H. Vosko and L. Wilk, "Influence of an improved local-spin-density correlation-energy functional on the cohesive energy of alkali metals," Phys. Rev. B **22**, 3812–3815 (1980).
[6]S. Kotochigova, Z. H. Levine, E. L. Shirley, M. D. Stiles, and C. W. Clark, "Local-density-functional calculations of the energy of atoms," Phys. Rev. A **55**, 191–199 (1997).
[7]J. Thijssen, *Computational Physics*.
[8]"Ionisation energy in the periodic table of elements," .