Density Functional Theory

In the solid state physics, we are solving the following many-body problem

$$H = \int d\mathbf{r}\Psi^\dagger(\mathbf{r})[-\frac{1}{2}\nabla^2 + V_{ext}(\mathbf{r})]\Psi(\mathbf{r}) + \frac{1}{2}\int d\mathbf{r}d\mathbf{r}'\Psi^\dagger(\mathbf{r})\Psi^\dagger(\mathbf{r}')v_c(\mathbf{r}-\mathbf{r}')\Psi(\mathbf{r}')\Psi(\mathbf{r}) \quad (1)$$

where $\Psi(\mathbf{r})$ is the field operator of electron, $V_{ext}(\mathbf{r})$ is the potential of nucleous and $v_c(\mathbf{r}-\mathbf{r}')$ is the Coulomb interaction $1/|\mathbf{r}-\mathbf{r}'|$. We assumed Born-Oppenheimer approximation for nuclei motion (freezing them since their kinetic energy is of the order of m_e/M_{nuclei}).

The fundamental tenet of DFT: **Any property of the system of interacting particles can be viewed as a functional of the ground state density $n(\mathbf{r})$!**

One scalar function of position (3 variables) $n(\mathbf{r})$ describes the system completely instead of (complicated) N-particle wave function $\Phi(\mathbf{r}_1, \mathbf{r}_2, ...\mathbf{r}_N)$. Here, the ground state electron density is $n(\mathbf{r}) = \langle\Phi|\Psi^\dagger(\mathbf{r})\Psi(\mathbf{r})|\Phi\rangle$.

The existance proof was given by *Hohenber and Kohn* (Phys. Rev. 136, B864-B871 (1964)).

- They proved that H is a unique functional of ground state electron density $n(\mathbf{r})$. In another words, there can not be two different external potentials V_{ext}^1 and V_{ext}^2 giving rise to the same ground state electron density. Hence, if we know $n(\mathbf{r})$, we know $V_{ext}(\mathbf{r})$ and we know H and therefore all the properties of the system of interacting electrons Eq. (1).

- They also proved that the kinetic part+electron-electron interaction part of H is a universal functional of density (The same for any V_{ext}). The total energy of the interacting many-body system can therefore be expressed by

$$E[n] = T[n] + E_{int}[n] + \int d\mathbf{r} V_{ext}(\mathbf{r}) n(\mathbf{r}) \tag{2}$$

  where $E = \langle\Phi|H|\Phi\rangle$ is the ground state energy and $T[n] + E_{int}[n]$ is the same functional for any system of interacting particles.

The second important step was made by *Kohn and Sham* (KS) in 1965 replacing the original many-body problem by an auxiliary independent particle problem. *The exact ground-state density of the interacting system* $n(\mathbf{r})$ *is equal to that of some choosen non-interacting system* $\rightarrow$ simple because one particle problem is exactly solvable, but difficult because the exchange-correlation functional is not known.

With KS ansatz, the total energy of the system is expressed by

$$E_{KS}[n] = \sum_{i\sigma} \int d\mathbf{r} \psi^*_{i\sigma}(\mathbf{r})(-\frac{1}{2}\nabla^2)\psi_{i\sigma} + \int d\mathbf{r} V_{ext}(\mathbf{r}) n(\mathbf{r}) + E_H[n] + E_{xc}[n] \quad (3)$$

where

$$n(\mathbf{r}) = \sum_{i\sigma \in occupied} |\psi_{i\sigma}(\mathbf{r})|^2, \quad (4)$$

$$E_H[n] = \frac{1}{2} \int d\mathbf{r} d\mathbf{r}' \frac{n(\mathbf{r}) n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \quad (5)$$

and E_{xc} is the unknown functional of n.

The variation of the ground-state energy δE vanishes, and with the normalization constrain on the auxiliary functions $\delta\left[E_{KS}[n] - \epsilon_i\left(\int d\mathbf{r}\psi^*_{i\sigma}(\mathbf{r})\psi_{i\sigma}(\mathbf{r}) - 1\right)\right] = 0$, we have

$$\frac{\delta E_{KS}}{\delta\psi^*_{i\sigma}(\mathbf{r})} = -\frac{1}{2}\nabla^2\psi_{i\sigma}(\mathbf{r}) + \left[V_{ext}(\mathbf{r}) + \frac{\delta E_H[n]}{\delta n(\mathbf{r},\sigma)} + \frac{\delta E_{xc}[n]}{\delta n(\mathbf{r},\sigma)}\right]\psi_{i\sigma}(\mathbf{r}) = \epsilon_i\psi_{i\sigma}(\mathbf{r}) \tag{6}$$

since $\frac{\delta n(\mathbf{r},\sigma)}{\delta\psi^*_{i\sigma}(\mathbf{r})} = \psi_{i\sigma}(\mathbf{r})$ and we defined spin density $n(\mathbf{r},\uparrow) + n(\mathbf{r},\downarrow) = n(\mathbf{r})$

The resulting one electron Schroedinger equation can be exactly solved! The problem is that functional E_{xc} is not known (and not simple). May not be invertible for systems with very degenerate ground state (like the paramagnetic Mott insulator with degeneracy 2^N).

Local Density Approximation (LDA)

For many existing materials, the Local Density Approximation (LDA) is remarkably successful approximation. In this method, the unknown functional is expressed by

$$E_{xc}[n] = \int d\mathbf{r} n(\mathbf{r}) \varepsilon_{xc}(n(\mathbf{r})) \tag{7}$$

where $\varepsilon_{xc}(n(\mathbf{r}))$ is the energy per electron at point $\mathbf{r}$ that depends only upon the density of electrons at the same point. Since $\varepsilon_{xc}(n(\mathbf{r}))$ is unique functional, it can be calculated for the uniform electron gas (Jellium model).

Density Functional Theory (DFT)

The question arises

For any ground state density of an interacting electron system, is it possible to reproduce the density exactly as the ground state density of the non-interacting electron system?

It turns out that this is not necessary the case ! There are counterexamples. However, it works reasonably well for many systems. Counterexample: Mott insulator with 2^n degeneracy per site.

The Local Density Approximation is very successful for *"wide-band"* systems with open s and p orbitals and sometimes also for d systems. It works also for band insulators but tends to fails in *strongly correlated systems* (with active d and f orbitals). The reason is that the DFT functional becomes almost degenerate and non-invertible. With the available approximate functionals (LDA,GGA) the ill-post problem of invertability becomes hard to overcome.

In this case it might not be a good idea to look at the functional of the density only, but of the energy dependent density, i.e., the Green's function. The Luttinger-Ward functional is one such generalization of the functional to the functional of the Green's function (rather than density).

One such method that can improve on LDA locality is the **Dynamical Mean Field Theory**

(DMFT) and its combination with LDA, dubbed **LDA+DMFT** method. The idea is that the free energy functional of the system can be expressed by the *local Green's function* $G_\omega(\mathbf{r})$ rather by the *density* $n(\mathbf{r})$ being the static analog $n(\mathbf{r}) = \mathrm{Tr}_{i\omega}[G_{i\omega}(\mathbf{r})]$. The local ansatz in this case includes important retardation effects and can be used also for the systems with strong correlations.

The problem is that the auxiliary system remains a *many-body* problem (although simplified in a form of a quantum impurity) and is much more involved than the non-interacting *Kohn-Sham* problem. For more information see Rev. Mod. Phys., Kotliar *et.al.* 2006.

Which properties can be exactly calculated in KS formulation provided the XC functional is known?

ground state propertis among others

- ground state energy
- ground state density
- ground state spin density
- static charge and spin susceptibility

The properties which can not be exactly calculated within KS formulation even if the XC functional is known

- the Fermi surface and therefore Mott insulator
- excitation energies and density of states
- the size of the gaps (of insulators and semiconductors)

Surprisingly, many times even these properties are in reasonable agreement with experiments.

Details of an LDA implementation

To solve the Kohn-Sham equations, we need the Hartree potential, the Exchange and Correlation potential.

The hartree energy density and potential is given by

$$\varepsilon_H(\mathbf{r}) = \frac{1}{2}\int d\mathbf{r}' \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} \tag{8}$$

$$V_H(\mathbf{r}) = \int d\mathbf{r}' \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} \tag{9}$$

The exchange potential comes from the familiar Fock term (see lectures on Hartree-Fock), but only the local part is kept in LDA (therefore the issue of self-interaction). From the Jellium model we know that the local part of the exchange is given by

$$\varepsilon_x = -\frac{3}{4}\left(\frac{3}{\pi}(n_\uparrow + n_\downarrow)\right)^{1/3} = -\frac{3}{4}\left(\frac{3}{2\pi}\right)^{2/3}\frac{1}{r_s} \tag{10}$$

$$V_x = \frac{4}{3}\varepsilon_x = -\left(\frac{3}{2\pi}\right)^{2/3}\frac{1}{r_s}. \tag{11}$$

Here we used the electron ”radius” r_s defined by $4\pi r_s^3/3 = 1/n$.

The correlation potential can be expressed by the energy density. From Eq. (7) we see

$$V_c(\mathbf{r},\sigma) = \frac{\delta E_c[n]}{\delta n(\mathbf{r},\sigma)} = \varepsilon_c[n(\mathbf{r})] + n(\mathbf{r})\frac{\delta \varepsilon_c[n(\mathbf{r})]}{\delta n(\mathbf{r},\sigma)} \tag{12}$$

The most accurate formulae for the exchange-corelation functional were obtained by fitting the QMC results for the Jellium model. Various parametrizations are available. We will use one of the most popular choices due to Vosko-Wilk (PRB 22, 3812 (1980))

$$\varepsilon_c = \frac{A}{2}\left\{\log\left(\frac{x^2}{X(x)}\right) + 2\frac{b}{Q}\arctan\left(\frac{Q}{2x+b}\right) - \frac{bx_0}{X(x_0)}\left[\log\left(\frac{(x-x_0)^2}{X(x)}\right) + \frac{2(b+2x_0)}{Q}\arctan\left(\frac{Q}{2x+b}\right)\right]\right\} \tag{13}$$

with $x = \sqrt{r_s}$, $X(x) = x^2 + bx + c$, and $Q = \sqrt{4c - b^2}$. Parameters are $A = 0.0621814$, $x0 = -0.10498$, $b = 3.72744$ and $c = 12.9352$. Finally, the potential is given by

$$V_c = \varepsilon_c - \frac{1}{6}A\frac{c(x-x_0) - bxx_0}{(x-x_0)(x^2+bx+c)}. \tag{14}$$

Now we can sketch the algorithm to solve LDA equations

- Choose initial guess for electron density $n(\mathbf{r})$.
- Calculate the Kohn-Sham potential $V_{KS} = V_{ext}(\mathbf{r}) + V_H + V_x + V_c$ using the above formulae
- Solve the Schroedinger equation

$$-\frac{1}{2}\nabla^2\psi_{i\sigma}(\mathbf{r}) + V_{KS}(\mathbf{r})\psi_{i\sigma}(\mathbf{r}) = \epsilon_i\psi_{i\sigma}(\mathbf{r}) \tag{15}$$

- Calculate the new electron density by $n(\mathbf{r}) = \sum_{i\sigma\in occupied} |\psi_{i\sigma}(\mathbf{r})|^2$ and the total energy on output density

$$E_{KS} = \sum_{i\sigma}\int d\mathbf{r}\psi^*_{i\sigma}(\mathbf{r})(-\frac{1}{2}\nabla^2)\psi_{i\sigma} + \int d\mathbf{r}\,(V_{ext}(\mathbf{r}) + \varepsilon_H[n] + \varepsilon_x[n] + \varepsilon_c[n])\,n(\mathbf{r}) \tag{16}$$

  where sum over $i\sigma$ runs only over occupied states $\psi_{i\sigma}$.

- Admix the new density to the old electron density and check for the total energy difference. Finish when accuracy achieved.
- Calculate physical properties using the converged KS functions

Now, we will develop the LDA program for atoms. This is simpler than applications to solids and is often used as a first step in applications to solids, in which atomic cores act like inert atoms.

We need to solve the Schroedinger equation numerically. The one-particle states are expanded in spherical harmonics, while the radial part is solved by numerical integration using Numerov algorithm.

The one particle states can be expressed by

$$\psi_{nlm}(\mathbf{r}) = Y_{lm}(\theta,\phi)r^{-1}u_{nl}(r) \tag{17}$$

where Y_{lm} are usual spheric harmonics and u_{nl} are radial functions to be determined numerically.

The atoms must have spherically symmetric charge density, because there is no prefered orientation for a single atom. (This is not necessary the case for atomic cores.) Since the charge is spherically symmetric, the potential is also spherically symmetric. The solutions of the Schroedinger equations are not symmetric, but a set, which can restore the symmetry, must be degenerate.

The Schroedinger equation for the atom has thus only the radial part of the form

$$-\frac{1}{2}\frac{d^2}{dr^2}u_{nl}(r) + \left[\frac{l(l+1)}{2r^2} + V_{KS}(r) - \epsilon_{nl}\right]u_{nl}(r) = 0 \tag{18}$$

while the angle-part is taken care of by spherical harmonics.

Numerov algorithm

The radial equation is usually solved with Numerov algorithm which is designed for the second order linear differential equation (DE) of the form

$$x''(t) = f(t)x(t) + u(t) \tag{19}$$

Due to a special structure of the DE, the fourth order error cancels and leads to sixth order algorithm using second order integration scheme. If we expand x(t) to some higher power and take into account the time reversal symmetry of the equation, all odd term cancel

$$x(h) = x(0) + hx'(0) + \frac{1}{2}h^2x''(0) + \frac{1}{3!}h^3x^{(3)}(0) + \frac{1}{4!}h^4x^{(4)}(0) + \frac{1}{5!}h^5x^{(5)}(0) + ...$$

$$x(-h) = x(0) - hx'(0) + \frac{1}{2}h^2x''(0) - \frac{1}{3!}h^3x^{(3)}(0) + \frac{1}{4!}h^4x^{(4)}(0) - \frac{1}{5!}h^5x^{(5)}(0) + ...$$

$$x(h) + x(-h) = 2x(0) + h^2(f(0)x(0) + u(0)) + \frac{2}{4!}h^4 x^{(4)}(0) + O(h^6) \tag{22}$$

If we are happy with $O(h^4)$ algorithm, we can neglect $x^{(4)}$ term and get the following recursion relation

$$x_{i+1} - 2x_i + x_{i-1} = h^2(f_i x_i + u_i). \tag{23}$$

But we know from the differential equation that

$$x^{(4)} = \frac{d^2 x''(t)}{dt^2} = \frac{d^2}{dt^2}(f(t)x(t) + u(t)) \tag{24}$$

which can be approximated by

$$x^{(4)} \sim \frac{f_{i+1}x_{i+1} + u_{i+1} - 2f_i x_i - 2u_i + f_{i-1}x_{i-1} + u_{i-1}}{h^2} \tag{25}$$

Inserting the fourth order derivative in the equation (22), we get

$$x_{i+1} - 2x_i + x_{i-1} = h^2(f_i x_i + u_i) + \frac{h^2}{12}(f_{i+1}x_{i+1} + u_{i+1} - 2f_i x_i - 2u_i + f_{i-1}x_{i-1} + u_{i-1}) \tag{26}$$

If we switch to a new variable $w_i = x_i(1 - \frac{h^2}{12} f_i) - \frac{h^2}{12} u_i$ we are left with the following equation

$$w_{i+1} - 2w_i + w_{i-1} = h^2(f_i x_i + u_i) + O(h^6) \tag{27}$$

The variable x needs to be recomputed at each step with $x_i = (w_i + \frac{h^2}{12}u_i)/(1 - \frac{h^2}{12}f_i)$.

The algorithm is surprisingly simple to implement as one needs only few lines of code. Here is the example for $u = 0$ (usual Schroedinger equation):

```
template <class funct>
void Numerov(funct& F, int Nmax, double x0, double dx, std::vector<double>& Solution)
{// Numerov algorithm for integrating the SODE of the form  x''(t)=F(t)x(t)
  // Solution[0] and Solution[1] need to be set (starting points)
  double h2 = dx*dx;  //square of step size
  double h12 = h2/12;  // defined for speed
  double w0 = (1-h12*F(x0))*Solution[0]; // first value of w
  double x = x0+dx;
  double Fx = F(x);
  double w1 = (1-h12*Fx)*Solution[1];    // second value of w
  double X = Solution[1];
  double w2;
  for (int i=2; i<Nmax; i++){
    w2 = 2*w1 - w0 + h2*X*Fx; // new value of w
    w0 = w1;
    w1 = w2;
    x += dx;
    Fx = F(x);                   // only one evaluation of F per step
    X = w2/(1-h12*Fx);         // new solution
    Solution[i] = X;
  }
}
```

One needs to solve Schroedinger eqution for the bound states ϵ_{nl}. We start solving Schroedinger equation far from the nucleus to avoid errors due to possible oscilations of $u_{nl}(v)$ close to nucleous. A good starting guess comes from the solution of hydrogen atom $u(R_{max}) = R_{max}e^{-ZR_{max}}$ and
$u(R_{max} - h) = (R_{max} - h)e^{-Z(R_{max}-h)}$.
The bound states must satisfy $u(0) = 0$. This is so-called two point boundary problem which is most efficiently solved by "shooting". Using numerov algorithm, the value of the solution $u(r)$ at nucleous can be determined for any choosen energy ϵ. For values of ϵ which give $u(0) = 0$, the solution is a bound state with the eigenvalue ϵ. To find all bound states, one needs to start searching at low enough energy and proceed wth certain small step to bracket all zeros in a choosen interval. We need as many zeros as the number of electrons in the atom (actually $N/2$). Than we need to use root-finding routine to determine solution precisely. The eigenfunctions need to be normalized $\int u^2(r)dr = 1$.

Once the eigenvalues and eigenfunctions u are known, we can find the chemical potential

$$N = \sum_{nl} 2(2l+1)f(\epsilon_{nl} - \mu) \tag{28}$$

and construct the new density

$$n(r) = n_\uparrow + n_\downarrow = \sum_{nl} 2(2l+1)\frac{|u_{nl}(r)|^2}{r^2} f(\epsilon_{nl}-\mu) \tag{29}$$

Than we proceed to calculate total energy using Eq. (8), (10) and (13)

$$E = \sum_{nl} 2(2l+1) f(\epsilon_{nl}-\mu)\epsilon_{nl} + \int dr 4\pi r^2 n(r)[-Z/r - V_{KS}(r) + \varepsilon_H(r) + \varepsilon_x(r) + \varepsilon_c(r)] \tag{30}$$

The new density should be admixed to the old density (with linear or Broyden-like mixing) and the new Kohn-Sham potential should be calculated. The exchange and correlation potentials are readily obtained from Eq. (11) and (14). The Hartree potential could be calculated by Eq. (9) but instead we rather solve the Poisson equation for V_H.

From classical electrostatic we remember $\nabla^2 \frac{1}{|\mathbf{r}|} = -4\pi\delta(\mathbf{r})$, therefore the Hartree potential

$$V_H(\mathbf{r}) = \int d\mathbf{r}' \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} \tag{31}$$

can be obtained by solving the Poisson differential equation

$$\nabla^2 V_H = -4\pi n(\mathbf{r}) \tag{32}$$

In spherical symmetric situation, the substitution $U(r) = V(r)r$ leads to a simple second order DE

$$U''(r) = -4\pi r n(r) \tag{33}$$

The boundary conditions are straightforward:

- $V_H(r = 0) < \infty$ therefore $U(r = 0) = 0$
- Taylor expansion of Eq (31) gives $V_H(r) \sim \frac{1}{r}\int d\mathbf{r}' n(\mathbf{r}')$ therefore $U(R_{max}) = N$ for large enough R_{max}.

Again we have two point boundary value problem which could be determined by shooting. But since we know the solution of the homogeneous equation $U''(r) = 0$, we don't need to shoot. It is better to solve for the initial conditions $U(0) = 0$ and $U'(0) = 1$ and than add homogenous solution (which is αr) so that U satisfies the desired two point boundary condition. The desired function U which satisfies boundary conditions is $U^{right}(r) = U(r) + r(N - U(R_{max}))/R_{max}$.

The Poisson equations can again be solved by Numerov algorithm. Setting f to zero and keeping u in Eq. 19, we see that variable w has to be choosen in the following way $w(t) = x(t) - \frac{h^2}{12} f(t)$, and the itteration step takes the form
$w(h) = 2w(0) - w(-h) + h^2 f(0)$

Now we are ready to sketch LDA algorithm for closed-shell atom

- Choose initial guess for electron density $n(r)$.
- Solve the Poisson equation $U''_H(r) = -4\pi r n(r)$ with conditions $U_H(0) = 0$ and $U_H(\infty) = N$.
- Calculate the Kohn-Sham potential $V_{KS}(r) = -Z/r + U_H(r)/r + V_x(r) + V_c(r)$ using the formulae for exchange and correlation potential
- Solve the Schroedinger equation $u''_{nl}(r) = 2(V_{KS}(r) + \frac{l(l+1)}{2r^2} - \epsilon_{nl})u_{nl}(r)$ with boundary conditions $u(\infty) = 0$ and $u(0) = 0$.
- Calculate the chemical potential requiring $N = \sum_{nl} 2(2l+1) f(\epsilon_{nl} - \mu)$ and construct the new density $n(r) = n_\uparrow + n_\downarrow = \sum_{nl} 2(2l+1) \frac{|u_{nl}(r)|^2}{r^2} f(\epsilon_{nl} - \mu)$

Density Functional Theory (DFT)

- Evaluate total energy on output density by

$$E = \sum_{nl} 2(2l+1)\epsilon_{nl} f(\epsilon_{nl}-\mu) + \int 4\pi r^2 n(r)[V_{ext} + \varepsilon_H + \varepsilon_{xc} - V_{KS}],$$

or equivalently,

$$E = \sum_{nl} 2(2l+1)\epsilon_{nl} f(\epsilon_{nl}-\mu) + \int 4\pi r^2 n(r)[-\frac{1}{2}\frac{U_H(r)}{r} + (\varepsilon_{xc} - V_{xc})]$$

- Admix the new density to the old electron density and check for the total energy difference. Finish when accuracy achieved.
- Calculate physical properties using the converged KS functions

The program is surprisingly simple of only 300 lines and can calculate ground state of many atoms very precisely. There are several limitations of the program:

- The radial grid should not be equidistant because the solution is very concentrated around nucleus where more points are required. Most of commercial programs implement logarithmic grid. The extension is rather straightforward, but numerov can not be used...

- Treats only the spherical symmetric part of the potential. This is correct for atoms, but not correct for solids. At least for Hartree term, the extension to non-spherical density is simple since it can be expressed by Gaunt coefficients.

  To evaluate Hartree potential, one can use the following identity

$$\frac{1}{|\mathbf{r}_1-\mathbf{r}_2|}=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{4\pi}{2l+1}\frac{r_<^l}{r_>^{l+1}}Y_{lm}(\hat{r}_1)Y^*_{lm}(\hat{r}_2) \tag{34}$$

  where $r_< = \min(r_1,r_2)$ and $r_> = \max(r_1,r_2)$. Further, one can use the Gaunt coefficients,

$$\langle Y_{l_1m_1}|Y_{l_2m_2}|Y_{l_3m_3}\rangle = (-1)^{m_1}\left[\frac{(2l_1+1)(2l_2+1)(2l_3+1)}{4\pi}\right]^{1/2} \tag{35}$$

$$\times \begin{pmatrix} l_1 & l_2 & l_3 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} l_1 & l_2 & l_3 \\ -m_1 & m_2 & m_3 \end{pmatrix} \qquad (36)$$

They can be evaluated using 3j symbols which are given standard textbooks on angular momentum (Racah formulas).

The precision of the results are however suprisingly good. We will compare them to published results at
`http://physics.nist.gov/PhysRefData/DFTdata/Tables/ptable.htr`

Using 10000 radial points with maximum radius 10 and dr=0.001 (which needs few second on PC for any atom), we get for He total energy E=-2.834836 Hartree which is the same as the published results. The kinetic energy is also the same 2.767922 Hartree.

The more complicated example is Krypton with 36 electrons and therefore filled shell. We get E=-2750.1453 while the published results is E=-2750.147940. The results can be improved by taking more dense mesh or implement logarithmic mesh.

Finally, we can test how good this approximatio is for open shell atoms (with partially filled shells). Lets take C as an example. Its electronic configuration is $C : [He]2s^2 2p^2$. The two outer p electrons should fill the Hund's coupling and should align in the same direction

$S = 1$. Further, they should be combined it total maximal $L = 1$. Finally the total angular momentum J is 0. But we did not take into account the spin-orbit coupling to get good quantum number J nor we implemented the $L = 1$ asymmetric ground state. Our estimation for total energy is -37.425743 and kinetic energy 37.190463, while published result is -37.425749 and 37.190391, respectively.

For oxygen $O : [He]2s^2 2p^4$ ($S = 1$,$L = 1$,$J = 2$) we get Etot=-74.473076 and Ekin=74.116882 while published results are Etot=-74.473077 and Ekin=74.116881.

Below are some main pieces of the code.

This class is used to solve the radial wave function and to find the bound states. Namely, the operator() is given to root-finding routine.

```
class RadialWave{
  int N;
  vector<double> R;        // Radial mesh but reversed R[0]=Rmax and R.last=0
  vector<double> Solution; // Non-normalized solution of Schroedinger equation
  vector<double> rhs;      // Right-hand-site in solving Schroedinger equation
  vector<double> Veff;     // Effective KS potential with centrifugal part
  vector<double> Veff0;    // Effective KS potential without centrifugal part
public:
  RadialWave(const vector<double>& Rmesh) : N(Rmesh.size()), Solution(N), R(N), rhs(N), Veff(N), Veff0(N)
  {
    for (int i=0; i<N; i++) R[i] = Rmesh[N-1-i]; // The mesh is reversed
    Solution[0] = R[0]*exp(-R[0]); // Boundary (starting) points of integration by Numerov
    Solution[1] = R[1]*exp(-R[1]); // Boundary (starting) points of integration by Numerov
  }
  // This function-operator is used to find bound states. It is given to root-finding routine
  double operator()(double E)
  {
    double h = R[1]-R[0];
    for (int i=0; i<N-1; i++) rhs[i] = 2*(Veff[i]-E); // The RHS of the SCHR-equation for choosen energy E
    rhs[R.size()-1]=0;                                 // This is the zero frequency
    Numerov(rhs, Solution.size()-1, h, Solution);      // Solving the radial SCH-equation
    int last = Solution.size()-1;                      // The last point at zero frequency needs extrapolation
    Solution[last] = Solution[last-1]*(2+h*h*rhs[last-1])-Solution[last-2];
    return Solution[last];                             // Value at zero frequency
  }
  // This function return the density of electrons (up+down per volume) of one (nl) state.
  void Density(vector<double>& rho)
  {
    rho.resize(Solution.size());
    int N = Solution.size();
    for (int i=0; i<Solution.size(); i++) rho[i] = Solution[N-1-i]*Solution[N-1-i]; // The mesh outside this class is rev
    double norm = 1./integrate4<double>(rho, R[0]-R[1], rho.size());                // Normalization constant
    for (int i=1; i<rho.size(); i++) rho[i] = rho[i]*norm/(4*M_PI*sqr(R[N-i-1]));   // rho_{nl}=u^2/(4*Pi*r^2)
    rho[0] = 2*rho[1]-rho[2];                                                       // extrapolation to zero frequency
  }
  // This function sets KS potential without centrifugal part
  void SetVeff0(const vector<double>& Uhartree, const vector<double>& Vxc, int Z)
  { for (int i=0; i<R.size()-1; i++) Veff0[i] = (-Z+Uhartree[N-1-i])/R[i]+Vxc[N-1-i];}
  void AddCentrifugal(int l)
  { for (int i=0; i<R.size()-1; i++) Veff[i] = Veff0[i] + 0.5*l*(l+1)/sqr(R[i]);}
  void SetVeff(const vector<double>& Uhartree, const vector<double>& Vxc, int l, int Z)
  { SetVeff0(Uhartree,Vxc,Z);
    AddCentrifugal(l); }
  double V_KS0(int i){return Veff0[N-1-i];}
};
```

The bound state can be found by the followng command

```
RadialWave wave(Rmesh); // Basic class for solving radial Schroedinger equation
double zero = zeroin(x0,x1,wave,1e-10); // Root-finder locates bound state very precisely
```

The Poisson equation is solved with the following few lines of code

```
void SolvePoisson(int Zq, const vector<double>& Rmesh, const vector<double>& rho, vector<double>& Uhartree)
{// Given the input density rho, calculates the Hartree potential
  static vector<double> RHS(Rmesh.size());
  for (int i=0; i<Rmesh.size(); i++) RHS[i] = -4*M_PI*Rmesh[i]*rho[i];
  Uhartree[0]=0;  Uhartree[1]=(Rmesh[1]-Rmesh[0]);// Boundary condition for U_H=V_H/r
  NumerovInhom(RHS, RHS.size(), Rmesh[1]-Rmesh[0], Uhartree); // Solving the 2nd order differential equation
  // adding homogeneous solution to satisfay boundary conditions: U(0)=0, U(infinity)=Z
  int ilast = Uhartree.size()-1;
  double U_last = Uhartree[ilast];
  double alpha = (Zq - U_last)/Rmesh[ilast];
  for (int i=0; i<Rmesh.size(); i++) Uhartree[i] += alpha*Rmesh[i];
}
```

and all the necessary bound state are calculated with

```
void FindBoundStates(int n0, int Z, double dEz, RadialWave& wave, vector<BState>& states)
{// Searches for bound state with given n,l. They are stored in vector<BState>
  vector<vector<double> > zeros(n0);            // Two dimensional function for staoring bound-state energies [n,l]
  for (int i=0; i<zeros.size(); i++) zeros[i].resize(n0); //2D needs to be resized, unfortunatrly. Better to used derived
  for (int i=0; i<states.size(); i++) states[i].E=100;// Sets some high energy not to mix with bound states
  int j=0;
  for (int n=0; n<n0; n++){
    for (int l=0; l<=n; l++){
      wave.AddCentrifugal(l);                        // Adds centrifugal part to effective KS potential
      double x = (n<=l) ? -0.5*Z*Z/sqr(l+1)-3. : zeros[n-1][l]+dEz; //Starts looking for zero
      double v0 = wave(x), v1=v0;
      while(x<10.){                                 // Looks for zero even at positive frequencies somethimes
        x+=dEz;                                     // Proceeding in small steps to bracket all zeros
        v1 = wave(x);                               // New value of radial function at origin
        if (v0*v1<0) {                              // Changs sign?
          double zero = zeroin(x-dEz,x,wave,1e-10); // Root-finder locates bound state very precisely
          zeros[n][l] = zero;                       // Stores into two places, just to remember where is "n-1"-bound stat
          states[j++] = BState(n,l,zero);           // Stores solution
          clog<<"Found solutin for n="<<n<<" l="<<l<<" at "<<zeros[n][l]<<endl;
          break;
        }
      }
    }
  }
}
```

The wave functions are than sorted with increasing energy and new density is calculated

```
    Cmp cmp;
    sort(states.begin(),states.end(),cmp); // Sort states
    Eb = BuildNewRho(Z, states, wave, nrho);// Gives new density and sum of eigenvalues
```

with this part of the code

```
double BuildNewRho(int Z, const vector<BState>& states, RadialWave& wave, vector<double>& nrho)
{// Knowing the energies of eigenstates, finds chemical potential and new charge density
  static vector<double> drho(nrho.size());
  for (int k=0; k<nrho.size(); k++) nrho[k]=0;
  double Eb=0;         // Sum of eigenvalues
  int Nt=0;     // number of electrons added
  for (int k=0; k<states.size(); k++){
    int l = states[k].l;
    int dN = 2*(2*l+1);  // degeneracy of each radial wave level
    double alpha = Nt+dN<=Z  ? 1 : (Z-Nt)/(2.*(2.*l+1)); // if shell is not fully-filled, take only part of charge
    wave.AddCentrifugal(states[k].l);
    wave(states[k].E);
    wave.Density(drho);
    for (int om=0; om<nrho.size(); om++) nrho[om] += drho[om]*2*(2*l+1)*alpha;
    Eb += 2*(2*l+1)*alpha*states[k].E; // Sum of eigenvalues times degeneracy
    Nt += dN;
    clog<<"Adding orbital n="<<states[k].n<<" l="<<states[k].l<<" E="<<states[k].E<<" "<<Nt<<endl;
    if (Nt>=Z) break; // Finish when enough electrons added
  }
  return Eb;
}
```

Results of LDA calculation

- LDA was intended for systems with slowly varying electronic density. Should fail for real materials.
- But: LDA is applied to all kind of systems and in general, LDA calculations lead to very good agreement with experiment, except for strongly correlated materials (with open f and sometimes d shells).
- Geometries
  - The accuracy of geometry is better than 0.1Å
  - Lattice constants are obtained within 4% or better.
- Energies
  - Accuracy for energies usually better than 0.2eV/atom in special cases even better than 0.01 eV/atom. (Factor of 2 worse than best quantum chemical calculations).
  - Atomization energies in simple molecules: up to 4kcal/mol ($\approx$ 0.2eV/atom)

Density Functional Theory (DFT)

Binding energy [eV]

System	LDA	HF	Exp
N_2	7.8	5.3	9.9
C0	9.6	7.9	11.2

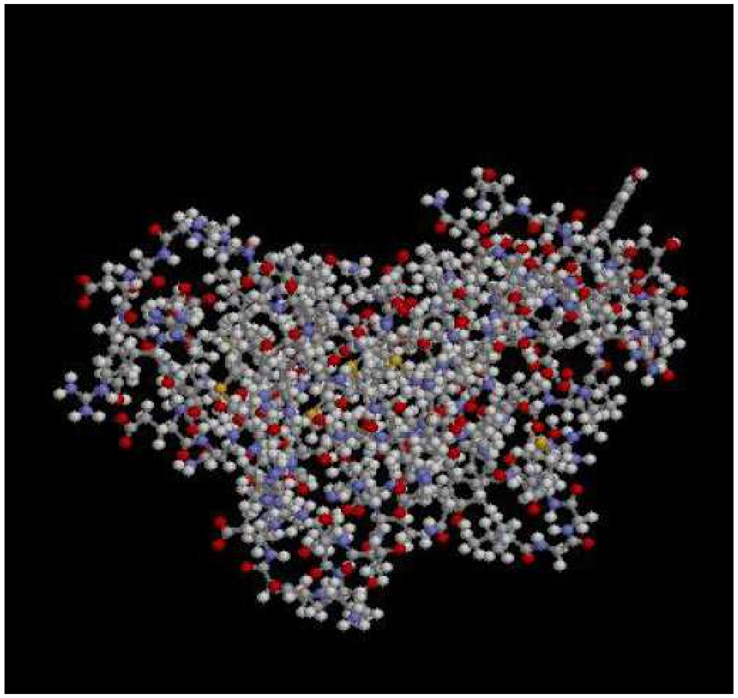
Bond length [Bohr]

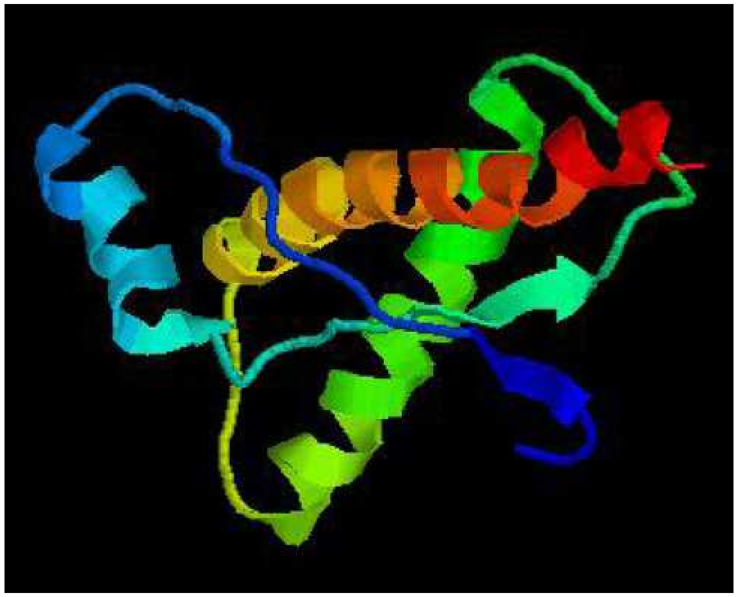
System	LDA	HF	Exp
N_2	2.16	2.01	2.07
CO	2.22	2.08	2.13

Atomization energy [kcal/mol]

System	E^{LSD}	E^{GGA}	Exp
H_2	113	105	109
LiH	60	52	58
Li_2	23	19	24
C_2H_4	633	571	563
CO	299	269	259
HCN	361	326	312

Application to biology





Left: atomic structure BSE:α-helix versus β-sheet $\rightarrow$ total energy problem

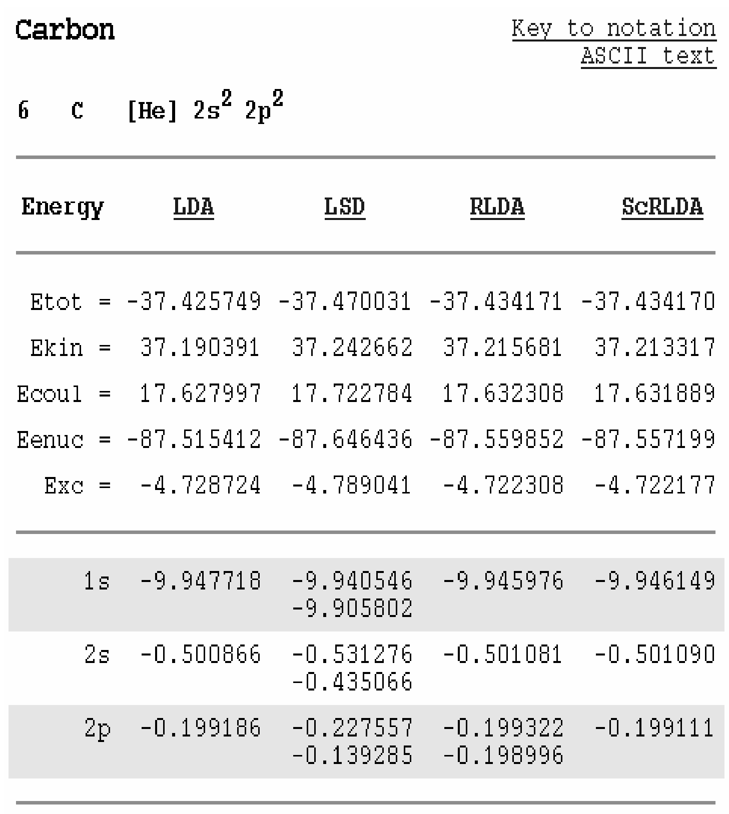
Right: Secondary structure

From Joerg Neugebauer; Protein Data Bank, http://www.rcsb.org/

Homeworks

- Implement the C++ code to compute the ground state energy of an atom.
- Test the code first on H and He.
- Check how precise is your results for heavier elements (like Carbon). Compare your results to http://physics.nist.gov/PhysRefData/DFTdata/Tables/ptable.html.

  For carbon you should get the following results:



Carbon

Key to notation
ASCII text

6 C [He] $2s^2$ $2p^2$

Energy	LDA	LSD	RLDA	ScRLDA
Etot =	-37.425749	-37.470031	-37.434171	-37.434170
Ekin =	37.190391	37.242662	37.215681	37.213317
Ecoul =	17.627997	17.722784	17.632308	17.631889
Eenuc =	-87.515412	-87.646436	-87.559852	-87.557199
Exc =	-4.728724	-4.789041	-4.722308	-4.722177
1s	-9.947718	-9.940546 -9.905802	-9.945976	-9.946149
2s	-0.500866	-0.531276 -0.435066	-0.501081	-0.501090
2p	-0.199186	-0.227557 -0.139285	-0.199322 -0.198996	-0.199111

- (optional) : Implement one of the above mentioned extensions
  - logarithmic radial grid (see details in Richard M. Martin's book and at http://physics.nist.gov/PhysRefData/DFTdata/radialgrids.html).
  - breaking the spin symmetry (check the original articles how correlation energy and potential change)
  - breaking of spheric symmetry (only for Hartree term but not for XC term. The implementation should be through gaunt coefficients- see again Richard M. Martin or any web page on gaunt coefficients).