Université de Bourgogne Franche-Comté

Quantum chemistry and density functional theory: survival guide

Authors:
Christophe RAMSEYER
Noah PERREAU

April 19, 2022

Contents

1 Introduction **3**

1.1 How to use this guide 3

2 The minimum theoretical knowledge **4**

2.1 The fundamental problem: nuclei and electrons 4

2.2 The hydrogen atom 5

2.3 Spin 8

2.4 Multi-electron systems and electron correlation 9

2.4.1 The heart of quantum chemistry: approximations 11

2.5 Density functional theory 13

2.5.1 Born-Oppenheimer approximation 13

2.5.2 Slater determinants to enforce antisymmetry 14

2.5.3 The foundation of DFT: Hohenberg-Kohn theorems 15

2.5.4 The Kohn-Sham equations implement the Hohenberg-Kohn theorems 16

2.5.5 The exchange-correlation functional 17

2.5.6 Discretization by choosing a basis set 17

2.5.7 Self-consistent field (SCF) iterations and convergence . . . 19

2.5.8 Pseudopotentials 20

2.5.9 Moving atoms: geometry optimization 21

2.5.10 Also moving atoms: ab initio molecular dynamics 22

3 Parallel programming and supercomputers **23**

3.1 The hardware 23

3.2 The software 24

4 Interacting with UNIX systems **25**

4.1 The terminal and some useful commands 26

4.1.1 `pwd`: *program working directory* and filesystem paths . . . 26

4.1.2 `cd`: *change directory* 27

4.1.3 `ls`: listing the contents of a directory 28

4.1.4 `cp`: copy files and directories 29

4.1.5 `mv`: move files and directories 29
4.1.6 `mkdir`: creating directories 29
4.1.7 `rmdir`: creating directories 30
4.1.8 `rm`: remove files, directories, and wreak havoc 30
4.1.9 `man`: print documentation about a command 30
4.1.10 Interrupting a command 31
4.1.11 Wildcards 31
4.1.12 Command history and autocompletion 31
4.2 Working with remote computers: SSH and X forwarding 32
4.2.1 The *Secure Shell* (SSH) 32
4.2.2 `scp`: secure copy 32

1 Introduction

Quantum chemistry is the application of quantum mechanics to atoms, molecules, solids, and possibly some more exotic states of matter. In its simplest form, it is the application of the Schrödinger equation to a set of charged particles, which include a given number of nuclei and a given number of electrons. Solving this equation gives all the knowledge about the chemistry of a specific system.

Its applications are virtually limitless, since chemistry and materials science are at the center of our society. What is the best alloy for a given application? Can we make it lighter and stronger by introducing such and such impurities, or make it brittle by introducing others? Can we increase its conductivity? Is it possible to tune its Fermi level? In which region of the spectrum is this or that material the most transparent? Can we oxidize hydrogen at low temperatures to prevent explosions? How efficiently can we convert methanol to dimethyl ether, a candidate green-fuel? And so on.

These few questions, and the extensive research carried out in these domains, show that there is a great incentive to solving quantum chemical problems on computers. Similar to other kinds of scientific computation, we want to be able to predict the properties of systems of interest without having to build or get near them, especially in the cases where is it impractical or dangereous to do so. If the quantum chemical problem could be solved trivially, we would be done, and chemistry would be completed; but just like many other types of scientific computation, *reality is subtly more complicated.* Finding solutions to this problem is incredibly hard, in both theoretical and practical ways, even with extremely powerful supercomputers.

In fact, quantum chemistry is much more about finding approximate and efficient methods rather than the original, full quantum chemical problem. On such methods, which has enjoyed wide testing and use, is *density functional theory* (DFT). This document, along with the pratical work, will show you the basics of a DFT computation, the typical results you can obtain, and suggest more complex studies it can be used for.

1.1 How to use this guide

You are not expected to read and understand everything in this guide. You certainly are encouraged to try to read and understand as much as you can, but will not be asked to explain some fine technical or theoretical point. But who knows, you might very well find a new interest in a field with plenty more to discover. It might also be the case that it definitely repels you away from it, though we hope not.

In any case, this guide provides a slight overview of the vast subject of com-

putational chemistry which you can use to understand a bit more about what is hapenning in the computations you will launch during the practicals. It is sometimes tempting to think that those programs we use, can be used as black-boxes: input some data, let it crunch the numbers, and get the results as output. This is far from the truth: a clever human operator is needed to select reasonable input data, and assess the value of the output. As we say in scientific computing, *garbage in, garbage out*.

2 The minimum theoretical knowledge

2.1 The fundamental problem: nuclei and electrons

It is known experimentally that matter is made of atoms, which themselves are composed of positively charged nuclei and negatively charged electrons. It is also known that nuclei are in fact made of smaller particles, protons and neutrons, which are in turn made of even smaller particles: quarks. Quantum chemistry is only concerned with nuclei and electrons, as dimensionless points with mass and electric charge (and commonly also *spin*, on which a few reminders will be given later).

The typical quantum chemical problem consists in finding the ground state structure of such a system of nuclei and electrons, without an external field. Such a system is described by the *time-independent Schrödinger* equation (TISE),

$$\hat{H}\Psi = E\Psi,$$

which is an eigenvalue problem, in which $\hat{H}$ is the hamiltonian of the system, Ψ is the ground-state wavefunction and E the associated energy. Solving this equation amounts to finding E and ψ; E gives information on the stability of the system relative to the conventional 0 level, corresponding to completely independent nuclei and electrons (the *unbounded* state), while ψ is a probability amplitude. The Born interpretation of quantum mechanics postulates that ψ^2 is the probability density for the quantum system to be in a given state. It depends on all the coordinates the system, which in quantum chemistry would be the nuclei's and electron's coordinates, as well as their spin if required.

The problem is completely specified by $\hat{H}$. In quantum chemistry, it is obtained by simply listing the kinetic energies of all particles, and the potentials due to the interaction between every possible pair of particles. Using atomic units, we can drop the electron's mass and charge, as well as Planck's constant, yielding the following hamiltonian:

$$\hat{H} = -\sum_{A=1}^{N_n} \frac{1}{2m_A}\nabla^2_{\vec{r_A}} - \sum_{i=1}^{N_e} \frac{1}{2}\nabla^2_{\vec{r_i}} + \sum_{A=1}^{N_n}\sum_{B>A}^{N_n} \frac{Z_A Z_B}{r_{AB}} - \sum_{A=1}^{N_n}\sum_{i=1}^{N_e} \frac{Z_A}{r_{iA}} + \sum_{i=1}^{N_e}\sum_{j>i}^{N_e} \frac{1}{r_{ij}} \qquad (1)$$

where N_N is the number of nuclei, N_e is the number of electrons, Z_A is number of protons in nucleus A (providing its chemical identity) and m_A its mass in units of an electron's mass.

It is often presented in the more manageable form:

$$\hat{H} = \hat{K}_N + \hat{K}_e + \hat{U}_{NN} + \hat{U}_{Ne} + \hat{U}_{ee}, \tag{2}$$

where $\hat{K}_N$ and $\hat{K}_e$ are the kinetic operators for the nuclei and the electrons, respectively, and the $\hat{U}_{xy}$ operators represent the potential for each pair of particles.

2.2 The hydrogen atom

For the hydrogen atom, we have a single nucleus (containing a single proton, if not considering other isotopes) and a single electron, so $N_N = 1$ and $N_e = 1$. The full hamiltonian simplifies to:

$$\hat{H} = -\frac{1}{2m}\nabla^2_{\vec{R}} - \frac{1}{2}\nabla^2_{\vec{r}} - \frac{1}{\|\vec{R} - \vec{r}\|} \tag{3}$$

where we now use $\vec{R}$ for the single nucleus' position and $\vec{r}$ for the single electron's position. Notice that there is no $\hat{U}_{ee}$ nor $\hat{U}_{NN}$ term, since there are no pairs of electrons or nuclei. Stated differently, in the hydrogen atom the only interaction we account for is the Coulomb attraction between the proton of the nucleus and the electron.

The hamiltonian can be simplified further if we take the nucleus as the origin of our reference frame. In this case, we always have $\vec{R} = 0$ and the kinetic energy term for the nucleus vanishes, since the nucleus doesn't move in this frame. Now we only have to consider the electron and its position, so we might as well simplify the notation by completely dropping any indices distinguishing between nuclei and electrons:

$$\hat{H} = -\frac{1}{2}\nabla^2 - \frac{1}{\|\vec{r}\|} = -\frac{1}{2}\nabla^2 - \frac{1}{r} \tag{4}$$

The radial symmetry from the Coulomb potential suggests using spherical coordinates. Indeed, if you write the hamiltonian using cartesian coordinates,

$$\hat{H} = -\frac{1}{2}\nabla^2 - \frac{1}{\sqrt{x^2 + y^2 + z^2}},$$

you can see that x, y, and z are hardly separable. Moving to spherical coordinates, the hamiltonian remains the same as in equation (4), but the wavefunction becomes a function of the spherical coordinates: $\Psi(r, \theta, \phi)$.

Now things become a bit more tedious, albeit not significantly more complicated. You probably do not remember the expression of the laplacian in spherical coordinates (nobody does), so here it is:

$$\nabla^2 = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial}{\partial r}\right) + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial}{\partial \theta}\right) + \frac{1}{r^2 \sin^2\theta}\frac{\partial^2}{\partial \phi^2} \tag{5}$$

The only useful thing to remember is that this complicated expression comes from the usual way you change from one coordinate frame to another, using the Jacobian matrix. This again is a very tedious derivation, which you may have done once in a calculus class, but once you've done it you earn the right to copy-paste it anytime you need.

The full Schrödinger equation for the hydrogen atom, using spherical coordinates, can now be written:

$$\begin{aligned} -\frac{1}{2}\left(\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial \Psi}{\partial r}\right)+\frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial\Psi}{\partial\theta}\right)+\frac{1}{r^2\sin^2\theta}\frac{\partial^2\Psi}{\partial\phi^2}\right) \\ -\frac{1}{r}\Psi(r,\theta,\phi)=E\Psi(r,\theta,\phi) \end{aligned} \tag{6}$$

The $1/r^2$ terms are commonly a source of problems, so we should multiply the whole equation by $2r^2$ and reorganize it a bit:

$$\begin{aligned} -\frac{\partial}{\partial r}\left(r^2\frac{\partial\Psi}{\partial r}\right)-\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial\Psi}{\partial\theta}\right) \\ -\frac{1}{\sin^2\theta}\frac{\partial^2\Psi}{\partial^2\phi}+2r^2\left(\frac{1}{r}-E\right)\Psi(r,\theta,\phi)=0 \end{aligned} \tag{7}$$

The second and third terms commonly arise when working in spherical coordinates: they are linked to the angular momentum of a particle on a circular trajectory. Another occasion you come across them is when modeling the rotational-vibrational spectra of diatomic molecules. So these terms define a new operator, $\hat{L}^2$:

$$\hat{L}^2=-\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right)-\frac{1}{\sin^2\theta}\frac{\partial^2}{\partial^2\phi}$$

We use $\hat{L}$ by analogy with classical angular momentum, which is usually denoted by $\vec{l}$. It can be verified by the principle of correspondance that indeed this represents the square of angular momentum. Angular momentum is of critical importance to quantum chemistry, which makes sense since we are interested in electrons *orbiting* around nuclei. The theory of angular momentum in quantum mechanics can be quite tricky to grasp, but here we will only consider the simple case of the hydrogen atom.

We can now introduce $\hat{L}^2$ into the Scrödinger equation and reorganize again:

$$-\frac{\partial}{\partial r}\left(r^2\frac{\partial\Psi}{\partial r}\right)+2r^2\left(\frac{1}{r}-E\right)\Psi+\hat{L}^2\Psi=0 \tag{8}$$

The Schrödinger equation is now separable: the operators acting on r are separated from the operators acting on θ and ϕ. Thus the resulting wavefunction is a product of an r-dependent part, called the *radial* part, and a (θ,ϕ)-dependent part, called the *angular* part. To keep things simple, we will admit that the eigenproblem for the $\hat{L}^2$ operator,

$$\hat{L}^2 f(\theta,\phi)=\lambda f(\theta,\phi),$$

is solved by a discrete set of functions f called the *spherical harmonics*. Spherical harmonics are usually denoted $Y_l^{m_l}$, where l is a positive integer and m_l is an integer ranging from $-l$ to $+l$. The associated eigenvalues are $l(l+1)$; this might seem to come out of nowhere, but we can assure you that if you work through the full theory, it will seem much simpler. Those two numbers, l and m_l are two of the four *quantum numbers* describing the state of an electron in the hydrogen atom; l is the *azimuthal quantul number* and m_l the associated *magnetic quantum number*.

Since we know the angular part of the hydrogen wavefunction are the spherical harmonics, we can now write the full wavefunction:

$$\Psi = R(r)\, Y_l^{m_l}(\theta, \phi)$$

where R is the radial part. Using separation of variables and the eigenpairs of $\hat{L}^2$, the Schrödinger equation becomes:

$$\left(-\frac{\partial}{\partial r}\left(r^2\frac{\partial R}{\partial r}\right)\right) Y_l^{m_l} + 2r^2\left(\frac{1}{r} - E\right) RY_l^{m_l} + l(l+1)\, RY_l^{m_l} = 0$$
$$\Rightarrow \frac{1}{R}\left(-\frac{\partial}{\partial r}\left(r^2\frac{\partial R}{\partial r}\right)\right) + 2r^2\left(\frac{1}{r} - E\right) + l(l+1) = 0$$

Rearranging everything neatly finally yields:

$$\left(-\frac{\partial}{\partial r}\left(r^2\frac{\partial R}{\partial r}\right)\right) + \left(2r^2\left(\frac{1}{r} - E\right) + l(l+1)\right) R(r) = 0$$

Or

$$-\frac{1}{2r^2}\left(\frac{\partial}{\partial r}\left(r^2\frac{\partial R}{\partial r}\right)\right) + \left(\left(\frac{1}{r} - E\right) + \frac{l(l+1)}{2r^2}\right) R(r) = 0$$

Here again, solving the equation is not particularly complicated, but it is really long. We will again admit that given enough time and effort, we can derive the solution, which the product of three main terms: r^l, $exp(-r/(na_0))$ and a function called the *associated Laguerre polynomial*. Imposing normalization adds very long constant in front of those three functions, but this is just a technical matter. Solving the equation naturally introduces a third quantum number, which is in fact the first: n, the principal quantum number. It is principal because all the other numbers depend on it. Also, in this model it is the only number on which the energy of the orbital depends:

$$E_n = -\frac{1}{n^2}$$

which is what we expect from experience and Rydberg's formula.

This closes this very short overview of the hydrogen atom theory. Any book on quantum mechanics goes into much more depth than we have done here. A particularly readable exposition, alongside a more general introduction to quantum chemistry, can be found in McQuarrie's *Quantum Chemistry* (second edition).

2.3 Spin

Spectroscopy is responsible for most of the early developments of quantum chemistry. Atoms can absorb and emit electromagnetic radiation in integer multiples of their energy levels. For instance, the eigenvalues of the hydrogen orbitals obtained in the previous sections are almost *exactly* those measured in experiments, if you do not look to closely. The hydrogen theory developed in the previous section was thought to be complete, until an odd experiment proved it slightly wrong. If you take any electron from the universe, say obtained by ionizing the atoms of a gas, accelerate it at a given speed and subject it to an external magnetic field, the electrons will slam into the measuring screen in exactly two locations. Some of them go up, and some of them go down, and there are about equal amounts of each. This experiment shows that electrons seem to behave like little magnets, some of which have one polarity, and some having another, opposite polarity. This was the discovery of *spin*.

The reason it is called spin is that the electron behaves *as if* it was spinning on an axis going through its center. That name has historically been very misleading. There is no theoretical nor experimental proof that the electron is in fact spinning. The only experimental evidence is that the electron has an *intrinsic magnetic momentum*, which can be thought as being caused by an *intrinsic angular momentum*. It can feel quite arbitrary, but two facts should help you get over this. First, there really is no equivalent to quantum spin in classical mechanics, as would be the case say for spatial coordinates. A particle has a position in classical mechanics and in quantum mechanics, though the way this position is described is different. A particle has an *intrinsinc angular momentum* in quantum mechanics, while it doesn't in classical mechanics, and we call it spin. If you want angular momentum in classical mechanics, you do have to spin an object, but not so in quantum mechanics. Second, while this is completely empirical in non-relativistic quantum mechanics, it can be justified using relativistic quantum mechanics. Spin arises as a natural property, as do some of its consequences, like the Pauli exclusion principle.

In quantum mechanics, we usually keep the empirical picture of spin, because it doesn't change the conclusions from a chemical point of view. Spin just exists, and we can account for it, independently from its origin. Thus, an electron in an atom or molecule has two angular momenta: the *orbital* angular momentum, caused by the orbital motion of the electron around the nuclei, and the *intrinsic* angular momentum, or spin.

To introduce spin into the theoretical picture, we use a similar formalism to that used for orbital momentum. We define an operator $\hat{S}^2$, which has eigenfunctions α and β with associated eigenvalue $s(s+1)$. Hence the α and β states are degenerate, and this is why for small atoms and in the absence of an external field, we cannot distinguish different spectral lines for a given energy level. In addition, we define the z-projection operator $\hat{S}_z$, which projects the intrinsic magnetic momentum

along the z axis (an arbitrary choice, but we like x and y too much to treat them differently). The eigenfunctions of $\hat{S}_z$ are also α and β, but the eigenvalues are now either $+\frac{1}{2}$ and $-\frac{1}{2}$, which we usually call *up* and *down*[1].

Contrary to a spinless wavefunction, we don't associate an expression to α or β; we rather use them as labels, to indicate that an electron has one spin value, or the other. When accounting for spin, atomic orbitals are expanded to a wavefunction including either α or β, and is then called a spin-orbital. That is, $\psi(r,\theta,\phi)$ becomes

$$\alpha\psi(r,\theta,\phi)$$

or

$$\beta\psi(r,\theta,\phi).$$

The existence of two different angular momenta can have subtle consequences on the electronic structure of materials. The fine structure of atom spectra can only be explained by the existence of these magnetic moments, and their interaction with external fields. Also, some phenoma arise because of coupling between orbital and intrinsic spin; this adds a *spin-orbit* term in the molecular hamiltonian, but those are advanced subjects beyond the scope of this practical work.

There are many systems in which spin doesn't play a significant role, and thus we only work with orbitals instead of spin-orbitals. Other systems depend dramatically on spin, the prototypical example being magnetic materials. Neglecting spin divides the amount of computation by two, so it is worth considering if the case applies. A computation considering spin is said to be *polarized*, or *unrestricted*.[2]

2.4 Multi-electron systems and electron correlation

Succeeding at solving the Schrödinger equation for the hydrogen atom, we might try a more complex system. A system which seems simple enough and makes for a natural progression, is the helium atom, which has two electrons around its nucleus. Again adapting the general hamiltonian to this specific case, we obtain:

$$\hat{H} = -\frac{1}{2m}\nabla^2_{\vec{R}} - \frac{1}{2}\nabla^2_{\vec{r_1}} - \frac{1}{2}\nabla^2_{\vec{r_2}} - \frac{Z}{\|\vec{R}-\vec{r_1}\|} - \frac{Z}{\|\vec{R}-\vec{r_2}\|} + \frac{1}{r_{12}} \tag{9}$$

There is still a single nucleus, with position $\vec{R}$ and mass m, but now there are two electrons, with positions $\vec{r_1}$ and $\vec{r_2}$. Having three particles, there are three kinetic energy terms. Having a single nucleus, there are still no nuclei-nuclei Coulomb repulsion terms. There are two electron-nucleus terms, for each electron. The

[1] Of course, it doesn't matter which is which, as long as we distinguish them. If "up" really is going to be "up" on a measuring screen, depends on the orientation of the measuring screen (relatively to the external magnetic field).

[2] There is also an intermediate: *restricted open-shell* methods, and there are both theoretical and technical reasons for choosing one method over another.

most import addition in the electron-electron repulsion term, which wrongfully looks simple.

This simple progression makes the problem immediately more complicated. In fact, it cannot be solved analytically because of coupling between the different variables (*i.e.* the nucleus and electron's positions). There is coupling between the nucleus and the electrons through the nucleus-electron attraction terms, and there is coupling between the electrons because of the electron-electron repulsion term. Coupling is problematic because it prevents using separation of variables to solve the differential equation, as we did for the hydrogen atom, even if we move to spherical coordinates.

We can decide to remain stuck forever, or start making clever approximations by assessing their potential influence on the results. The first approximation we can make is to consider the nuclei to be fixed. For the hydrogen atom, it was just a convention, but here it is an approximation because of the *coupling* between electronic and nuclear coordinates. Being fixed, we might as well choose the nucleus to be at the origin, dropping $\vec{R}$ from the hamiltonian. If we also decide that now $\vec{r_1}$ and $\vec{r_2}$ are expressed in this new reference frame, we obtain the simplified hamiltonian:

$$\hat{H} = -\frac{1}{2}\nabla^2_{\vec{r_1}} - \frac{1}{2}\nabla^2_{\vec{r_2}} - \frac{Z}{r_1} - \frac{Z}{r_2} + \frac{1}{r_{12}} \tag{10}$$

It is useful to notice that the hamiltonian for the helium atom contains the hamiltonian for the hydrogen atom twice, one for each electron, in addition to the electron-electron term. Let us define a partial hydrogen-like hamiltonian for an electron i as

$$\hat{h}_i = -\frac{1}{2}\nabla^2_{\vec{r_i}} - \frac{Z}{r_i} \tag{11}$$

The hamiltonian for the helium atom thus becomes

$$\hat{H} = \hat{h}_1 + \hat{h}_2 + \frac{1}{r_{12}} \tag{12}$$

We have not removed any of the difficulty caused by the electron-electron term, since it will still prevent separation of variables. If we imagine for a moment that electrons do not interact together, the hamiltonian would simplify to

$$\hat{H} = \hat{h}_1 + \hat{h}_2, \tag{13}$$

which is perfectly separable. Since $\hat{h}_i$ is the hamiltonian for a hydrogen-like system, the resulting wavefunction would then just be a product of the hydrogen-like orbitals (which can be conveniently written in spherical coordinates):

$$\Psi(r_1, \theta_1, \phi_1, r_2, \theta_2, \phi_2) = \psi_H(r_1, \theta_1, \phi_1) \cdot \psi_H(r_2, \theta_2, \phi_2)$$

where ψ_H are hydrogen-like atomic orbitals. In fact, *any* system however large would be trivial to solve without electron interaction: it is just a superposition

of the single-electron hydrogen wavefunctions, one for each electron. We would have solved the central problem of quantum chemistry, and we would be able to compute the structure of any molecule, any crystal, any ion or radical; we would be able to predict any chemical reaction, and probably systematically search for the best catalysts for a given reaction.

But electrons do interact.

2.4.1 The heart of quantum chemistry: approximations

The Schrödinger equation becomes distressingly unsolvable, but not completely unusable. One could try a direct numerical approach, for instance by discretizing each operator in each dimension using finite differences. Let us see where it leads in the case of the simple water molecule, which has 3 nuclei and 10 electrons. Each has three degrees of freedom, not counting spin, so there is a total of 39 degrees of freedom. Assuming we place the molecule at the center of a sufficiently large box, we can discretize each of the 39 dimensions of the problem into a number P of points at which the various quantities and operators can be evaluated. If we use for instance $P = 10$, this amounts to 10^{39} points; assuming we store them as 4 bytes, we would need around 3.64×10^{27} terabytes of memory. This is quite a lot, and this is just for memory. The computing time required to solve such a problem is also ridiculously large.

This calls for methods specific to this very problem. Fortunately, people have been working on the subject for nearly as long as quantum chemistry exists, so there is a wide variety of approximation methods. A few of the most import ones are worth mentioning:

- *Perturbation theory* has been used in other quantum mechanical problems, and can be applied here as well. The non-perturbed hamiltonian is the molecular hamiltonian without electron correlation:

  $$\hat{H}_0 = \sum_{i=1}^{N_e} \hat{h}_i$$

  and the electron-electron term is the pertubation, yielding the full hamiltonian:

  $$\hat{H} = \hat{H}_0 + \sum_{i}^{N_e} \sum_{j>i}^{N_e} \frac{1}{r_{ij}}$$

  It will be shown in section 2.5.2 that the $\hat{H}_0$ part is trivial to solve, and the correlation part is handled in the usual way up to a given order. The *Moller-Plesset* (MP) methods are based on perturbation theory, and their name usually indicates to which order they are used: MP2 for second order, MP4 for fourth order, and so on. Increasing the order of perturbation increases the accuracy of the computation, but of course does so at the expense of the amount of computation.

- *Hartree-Fock* (HF) methods were among the first methods developed in quantum chemistry, and alongside DFT are still widely implemented in computer programs nowadays. In fact, HF methods is more a family of methods than a single method on its own. More precisely, one usually talks about *post-HF* methods, which are based on the original HF method but use various mechanisms to make it more accurate. There are *lots* of post-HF methods with a wild variety of acronyms (some of the most common would be CI, CISD, CISD(T), CC, CCSD, MRSCF, ...), each providing a different tradeoff between accuracy and computing resources.

  As HF theory was developed very early in quantum chemistry, it can be considered as one of its foundations. Many other methods use some bits of it, including DFT.

- *Quantum Monte Carlo* (QMC) methods use the framework of Monte Carlo methods, applied to the electronic structure problem. Those methods tend to be expensive, but they are unbiased: it is fairly straightforward to obtain more accurate results, by increasing the number of samples for instance. Thus QMC methods tend to be used to compare and fine-tune parameters in other methods, which themselves can then be used on larger systems.

The important point to remember is no method fits all use cases. There is always a tradeoff between accuracy, precision and computer resources. For instance, the full CI method from the HF family can solve the molecular problem *exactly* at the limit (within numerical precision of course, and given a basis set; more on those in section 2.5.6). But the computing time it requires, though not as large as a direct finite-difference scheme, still grows extremely fast. It can therefore only be used for very small systems, with just a few electrons. Many methods derive from it by trying to reduce drastically the amount of required computations, because many of those computations commonly contribute very little to the final result.

Still, more approximate methods exist to solve problems more efficiently, or work with larger problems. For each method, there is a conventional limit to the size of the problem that can be handled (i.e. the number of electrons or equivalently, the number of atoms, since each atom has a given number of electrons). Beyond this size, the required computer resources become too large. Thanks to advances in computer technology, larger and larger problems could be solved the last fifty years. The first computers used in quantum chemistry could do a few atoms at best!

Nonetheless, as powerful as computers get, we already know we have plenty of complex methods to keep them busy.

2.5 Density functional theory

Density functional theory (DFT) is now one of the most commonly used quantum chemical method. As other numerical methods, it is established through a series of approximations and theorems, and then implementation details. The path to set up the theoretical framework of DFT can seem overwhelming, but the goal of this section is to show it clearly and in order.

2.5.1 Born-Oppenheimer approximation

The first approximation generalizes an idea used earlier for the helium atom: since the electron is three orders of magnitude lighter than even the lightest nucleus, their respective motion can be decoupled. This is the *Born-Oppenheimer approximation*. Under reasonable conditions, the electrons will adapt almost instantaneously to nuclei positions, and so the electrons can be considered to depend only parametrically on the nuclei positions. This amounts to separating the nuclear and electronic coordinates, and so the molecular hamiltonian can be written as

$$\hat{H} = \hat{H}_n + \hat{H}_e \ , \tag{14}$$

where $\hat{H}_e$ contains only electronic operators and $\hat{H}_n$ contains only nuclear operators. The resulting wavefunction can then be written as a product of a nuclear and an electronic part:

$$\Psi(\vec{R}, \vec{r}) = \Psi_n(\vec{R}) \cdot \Psi_e(\vec{R}; \vec{r}) \tag{15}$$

where $\vec{R}$ and $\vec{r}$ informally designate the set of nuclear and electronic coordinates, respectively. It should be stressed that Ψ_e depends only parametrically on the nuclear positions; that is, the electrons move in the external potential created by the nuclei, but there is no coupling with the nuclei's motion.

The nuclear hamiltonian is useful in understanding vibrational and rotational spectroscopy, among other things, but the structural properties of the molecule are obtained from the electronic hamiltonian:

$$\hat{H}_e = \hat{T}_e + \hat{U}_{ne} + \hat{U}_{ee} \tag{16}$$

Be careful about a convention which changes from author to author (and thus is *not* a convention): sometimes the nucleus-nucleus potential is included in the electronic hamiltonian. Since it involves only nuclear coordinates, it is constant when solving for the electronic wavefunction and just shifts the absolute energy values. Also, in quantum chemistry we usually drop the subscripts to distinguish nuclear and electronic coordinates and wavefunctions, unless mentioned otherwise. So from now on, we work on the electronic hamiltonian $\hat{H}$.

2.5.2 Slater determinants to enforce antisymmetry

The Born-Oppenheimer approximation helps in reducing the dimensionality of the problem we are trying to solve, but only does so by a constant factor (proportional to the number of nuclei). We saw that with independent electrons (that is, $\hat{U}_{ee}$), the electronic hamiltonian is a sum of single-electron hamiltonians:

$$\hat{H} = \sum_{i=1}^{N_e} \hat{h}_i$$

and the electronic wavefunction is then a product of atomic orbitals:

$$\Psi(\vec{r_1}, \ldots, \vec{r_{N_e}}) = \psi_1(\vec{r_1}) \ldots \psi_2(\vec{r_{N_e}})$$

where ψ_1 and ψ_2 are atomic orbitals.

Neglecting electron correlation isn't feasible, but we might still *consider* we can write the electronic wavefunction as a simple product of atomic orbitals, even if we do have interacting electrons. The point in such an approximation is to reduce the dimensionality of the problem, from a 3N-space to a 3-space. Remembering the example of the water molecule, it should be motivating enough. It might feel a bit arbitrary; so people call an approximation such as this an *ansatz* instead of *arbitrary*. But when we make the choice of an ansatz like this, we try something and see if we get *somewhere* when anything else is too difficult.

This manner of writing the wavefunction, called a *Hartree product*, did provide interesting early results, especially considering these were done by hand. It has however a major drawback: it is not antisymmetric under coordinate exchange. For instance, if $\Psi(\vec{r_1}, \vec{r_2})$ is the wavefunction for a system with two electrons, it simply means that we have to have

$$\Psi(\vec{r_2}, \vec{r_1}) = -\Psi(\vec{r_1}, \vec{r_2})$$

The origin of this constraint is Pauli's exclusion principle. It can be demonstrated in relativistic quantum mechanics, but in non-relativistic quantum chemistry we just take it as an additional constraint alongside the Schrödinger equation. You can easily convince yourself that the Hartree product doesn't satisfy this condition.

Slater determinants provide a slightly more complicated ansatz which does satisfy wavefunction antisymmetry. Their general for N electrons is

$$\Psi \approx \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_1(\vec{r}_1) & \ldots & \psi_N(\vec{r}_1) \\ \vdots & \ddots & \vdots \\ \psi_1(\vec{r}_N) & \ldots & \psi_N(\vec{r}_N) \end{vmatrix},$$

where the ψ_i are single-electron wavefunctions, and the leading factor is just a normalization constant. Antisymmetry can easily be shown in the 2-electron case:

$$\Psi(\vec{r}_1, \vec{r}_2) = \frac{1}{2} \begin{vmatrix} \psi_1(\vec{r}_1) & \psi_2(\vec{r}_1) \\ \psi_1(\vec{r}_2) & \psi_2(\vec{r}_2) \end{vmatrix} = \frac{1}{2} (\psi_1(\vec{r}_1)\psi_2(\vec{r}_2) - \psi_1(\vec{r}_2)\psi_2(\vec{r}_1))$$

$$\Rightarrow \quad \Psi(\vec{r}_2, \vec{r}_1) = \frac{1}{2} (\psi_1(\vec{r}_2)\psi_2(\vec{r}_1) - \psi_1(\vec{r}_1)\psi_2(\vec{r}_2)) = -\Psi(\vec{r}_1, \vec{r}_2)$$

You may start to mix things up a bit at this point. Remember that the starting point of everything is the *full wavefunction*, depending on all the coordinates of all the electrons in our system. We cannot obtain it easily, and so we look for approximations. One idea is to write it as a product of hydrogen-like wavefunctions, also called *atomic orbitals*, which we could obtain analytically. Slater determinants is a neat way of writing an N-electron wavefunction as a product of N single-electron wavefunctions, i.e. the atomic orbitals of the hydrogen atom. If we treated the subject in a bit more depth, we would also talk about *molecular orbitals*. But suffice it to say anything beyond the hydrogen atom is an approximate wavefunction. And a recurring theme in quantum chemistry is trying to write those approximate wavefunctions with those of the simple hydrogen atom, since we can express them explicitly[3].

2.5.3 The foundation of DFT: Hohenberg-Kohn theorems

The theoretical basis for density functional theory was established by Hohenberg and Kohn in 1964 through the following to theorems:

1. The ground state electronic density uniquely determines all the properties of the system. In particular, the electronic energy is a functional of the electronic density.

2. The actual ground state density is the one which minimizes this energy functional.

The first theorem proves that there is a bijection between the ground state electronic density and the hamiltonian of the system. More precisely, the system-specific part of the hamiltonian and the ground-state are uniquely related, all other parts being common to all electronic problems. It immediately appears to simplify our problem since the wavefunction is a 3N-coordinates function, while the electron density has only 3 space coordinates.

All the observables of interest can be computed as a *functional* of the density, that is, a function which associates a scalar to a function. Commonly, functionals are expressed as integrals and arise in variational calculus, optimization problems or weak formulations of partial differential equations. Because observables are functionals of the electron density, the theory based on the HK theorems has been called *density functional theory* (DFT).

The second theorem is a bit more abstract but also very useful. It essentially states the problem in a *variational formulation*: among all the possible electron densities you can choose, as unrealistic and distorted as they may be, there is one and only one which does minimize the energy functional, and that is the ground

[3] And we don't have much of a choice anyways.

state electron density. Thus things are apparently easy: minimize the functional, and you're done, right?

2.5.4 The Kohn-Sham equations implement the Hohenberg-Kohn theorems

The first HK theorem is an existence proof: it shows that a functional exists, which can provide all the observables only using the ground state density. It doesn't say anything about the functional itself, however. Little is known about it, and this in fact makes sense: the HK theorems are not approximations in themselves, they just formulate the Schrödinger equation for the electronic problem in terms of electronic density instead of the wavefunction. This formulation is just as strong as the original Schrödinger equation, and therefore just as complicated to solve. It would be an extremely powerful tool, mathematically speaking, if one could systematically solve any Schrödinger-like equation with this functional!

Therefore, additional approximations are therefore needed to establish a more useful formulation. The most common way this is done is through the Kohn-Sham (KS) equations. The main idea is to write everything we know how to write, and put everything else in a term we will need to approximate. HF theory provides many of those things: using Slater determinants to express the wavefunction and using the average effect of electrons on each other, among other things. Another ingredient is the use of Lagrange multipliers to formulate everything as an optimization problem, since the second HK theorem says we can do so.

The essential part is that in the end, we obtain N_e single-electron Schrödinger equations, the KS equations, one for each of the N_e orbitals. They look just like any standard Schrödinger equation:

$$\left(-\frac{1}{2}\nabla^2 + V_s\right)\psi_i = \epsilon_i \psi_i$$

where ψ_i is the orbital of electron i as usual, we have the KS hamiltonian on the left term and the KS eigenvalue ϵ_i on the right. The hamiltonian contains a kinetic energy operator for that orbital, but it is *not* the same operator as in the original Schrödinger equation: this kinetic operator applies to a single orbital, while in the original problem it applies to the full wavefunction. Another crucial part is the V_s term, which is the *effective potential*: a potential which approximately reproduces the effect of the full N_e-electron Schrödinger equation, but for that single electron. It contains the terms from Hartree-Fock theory, like the average Coulomb repulsion from the other electrons, but most importantly it also contains an *exchange-correlation* term.

Exchange-correlation is a central topic to DFT. It contains everything we don't have any expression for, nor from HF theory or anything else. It is a completely approximate term, and physicists and chemists think hardly on how to provide expressions for it. Usually one hears about the *exchange-correlation functional*

(XC functional), since DFT formulates everything as a functional of electron density. But in the end, the XC functional is applied to a given electron density and just adds a potential term to the KS equations.

2.5.5 The exchange-correlation functional

There are plenty of XC functionals in common use, and choosing one can be quite troublesome. We will not go too deep in these considerations here, but like most things in scientific computing, the choice is a matter of tradeoffs. Some functionals are accurate given some problems, while some are accurate in others. Some functionals are derived completely theoretically, while some use experimental data. Some functionals provide good results for a diversity of systems compared to others, but are far more expensive to compute.

Fundamentally, choosing a functional requires looking at the literature for suggestions or even benchmarks, and then trying on your own system. Common default choices are PBE and B3LYP, the first for physicists and the second for chemists. In any case, this shows an important aspect of quantum chemistry: one should always try to compare results to other's results, wether numerical or experimental.

2.5.6 Discretization by choosing a basis set

We are almost there; having formulated the HK theorems and the KS equations based on it, we now need to devise a method of solving those equations on a computer. In theory, the orbitals are continuous functions and would therefore need an infinite amount of information to be represented in computer memory. Thus, we *discretize* the orbitals: we choose a set of functions we call a *basis set*, and write the orbitals approximately as a linear combination of those basis functions:

$$\psi_i = \sum_n c_n \chi_n$$

where ψ_i is the orbital for electron i, χ_n is one of the basis functions, and c_n is the associated coefficient in the linear combination.

The basis set is not rigorously a basis since it doesn't span the full space of the orbitals. But, as for other numerical schemes like the finite element method, we can consider it to be a basis to an *approximate* space of orbitals, into which we will look for the closest combination which yields a result close to the actual wavefunction.

The choice and influence of basis sets could easily fill volumes on its own, but for this introduction it is enough to say that there are three main types of basis functions:

- *Gaussian Type Orbitals* (GTO) rule in the kingdom of traditional quantum chemistry, and historically have been used mostly with Hartree-Fock methods. These use nuclei-centered gaussian functions, which makes sense since electrons tend to remain close to the nuclei. Thus, GTOs are very efficient for molecular systems and in reactivity studies, but much less so for extended systems like solids.

- *Plane waves* are traditionally associated with solid-state physics, since the Bloch theorem shows that wavefunctions are periodic in periodic potentials (like crystalline solids). It lends itself to efficient implementation thanks to Fast Fourier Transform (FFT) algorithms. Many things are simpler to compute by going back and forth between direct and reciprocal space. This is typically the case for derivation operators (like the Laplacian in the kinetic energy term), since derivation is equivalent to simple multiplication in reciprocal space. Plane waves and gaussian functions are essentially complementary to each other, and determine to a large extent the type of computation that can be done.

- *Real-space grids* have been developed more recently with the advent of highly-parallel computers. All the functions are expanded on real-space grids, and the operators discretized using simple finite-difference approximations. Despite its simplicity, this approach has a number of advantages: it is unbiased, it is simple to improve accuracy by using a larger basis set, it is simple to implement and trivial to parallelize.

The choice of a basis set is critical both for the scientist using a program, and the implementer of the program. The scientist needs to know which systems can be represented accurately with which basis set, and which associated parameters. The implementer wants to write an efficient program, but it is difficult to be efficient for both plane waves and gaussians at the same type. This is why some programs such as VASP, which uses plane waves, are more popular among material scientists, while programs like Gaussian which uses gaussians[4], are more popular among chemists.

Nonetheless, the choice of a basis set is a technical matter, and *in principle* any system can be handled with any sufficiently large basis set of one kind or the other. It just might be that to reach reasonable accuracy, one would need an unreasonable amount of computating time and memory. Some methods like augmented plane waves (APW) try to mix both gaussians and plane wave, to benefit from the positive aspects of both.

[4] Obviously.

2.5.7 Self-consistent field (SCF) iterations and convergence

Once a basis set has been chosen, the Kohn-Sham hamiltonian can be computed explicitly. There is, however, a slight problem: the KS hamiltonian depends on the electron density, which depends on the orbitals, which we compute by solving the KS equations, which we state by computing the KS hamiltonian... This is a circular definition, and because the KS hamiltonian depends non-trivially on the orbitals, we can't simply put everything to one side and solve a typical eigenproblem. Instead, this is a nonlinear eigenvalue problem, which can be solved *iteratively*.

The idea is the following: choose an initial set of orbitals $\psi_i^{(0)}$. Remembering that each orbital is expanded in terms of the basis set χ_n,

$$\psi_i^{(0)} = \sum_n c_{i,n}^{(0)} \chi_n,$$

this amounts to choosing the initial coefficients $c_{i,n}^{(0)}$ for each orbital ψ_i. It is not simple to make a good initial choice, but this is handled by the program and we rarely have to care about it. One common choice, though, is to start with hydrogen-like atomic orbitals; that is, we choose initial orbitals as if we were solving the molecular problem *without* electron correlation.

Given these orbitals we chose, we can compute the initial electron density $n^{(0)}(\vec{r})$, from which in turn we compute the initial KS hamiltonian $\hat{h}_{KS}^{(0)}$. This defines the following standard eigenproblem:

$$\hat{h}_{KS}^{(0)} \psi_i^{(1)} = \epsilon_i^{(1)} \psi_i^{(1)}$$

in which we solve for the new orbitals $\psi_i^{(1)}$ and the associated eigenvalues $\epsilon_i^{(1)}$. You might see where this is going: the new set of orbitals can be used to build a new electron density $n^{(1)}$, which can be used to build a new KS hamiltonian $\hat{h}_{KS}^{(1)}$, which we can use to solve yet another standard eigenproblem, and so on. We do this until we reach *convergence*: the orbitals have stabilized and don't change very much from one iteration to the next. In more mathematical terms, we reach a *fixed-point*: doing more iterations will not change the solution further, up to numerical precision of course. Yet another way of saying this is that we rearch *self-consistency*: input orbitals define a hamiltonian which is going to yield the same orbitals as output. Thus, this iterative procedure was called the *self-consistent field* (SCF) loop.

It may not seem obvious why this scheme works. Actually there are not many mathematical guarantees of convergence, but in practice we just know that if we are careful, it will often converge. As mentioned, one should pay attention to the initial orbitals: they shouldn't be too far from the final result, otherwise the output orbitals might converge to something else completely.

A trickier problem is that if the output density $n^{(k)}$ is used directly in iteration $n^{(k+1)}$, the SCF iteration is almost guaranteed to oscillate and diverge. *Density*

mixing schemes are used to smooth the variation between successively computed densities, at the cost of longer time to convergence[5]. The simplest method linearly interpolates between the input and the output, using a small and constant parameter α:

$$n_{in}^{n+1} = \alpha n_{out}^{n} + (1-\alpha)\, n_{in}^{n} \tag{17}$$

There are more sophisticated method, such as Broyden mixing which relies on a nonlinear solver itself. Quantum chemistry programs usually provide several mixing schemes, but it is more an art to choose parameters such as these. When studying a new system, one often spends quite a bit of time fiddling with the parameters to find a set which works reasonably well.

2.5.8 Pseudopotentials

The electron-nuclei Coulomb potential is the source of a significant technical problem. As it depends on the distance from the electron to the nucleus, the potential grows to infinity as the electron comes closer and closer to the nucleus. We know from quantum theory that the electron won't fall into the nucleus, but the closer it gets, the stronger the interaction.

In fact, for electrons in the lowest shells (that is, closest to the nucleus), the Coulomb potential is so high that most other terms have a negligible impact, especially electron correlation. One can say that the motion of those *core* electrons is almost independent of anything else happening in the system, wether it is neighbouring electrons or the electrons of more distant atoms. On the contrary, the electrons which are in the outermost shell are much less attracted by the nucleus, because they are further apart, and because the lower-lying electrons have a shielding effect.

It is also known from experimental chemistry that in the huge majority of reactions, only those extern, *valence* electrons actively participate. The core electrons are selfishly almost undisturbed by anything happening anywhere other than their nucleus.

This is not quite a problem when using gaussian type orbitals, because the basis functions are localized anyway and can describe the core electrons well. But it is a problem for plane waves. Higher potentials cause quick oscillations in the orbitals, which in turn requires using plane waves with smaller wavelengths to represent orbitals correctly. So, we have to expand a lot of effort trying to describe correctly orbitals which almost never change!

Pseudopotentials were introduced to mitigate this problem. The orbitals of core electrons are removed, but their effect must remain if we are to simulate the same system. This is done by adding a potential term in the Kohn-Sham hamiltonian, which indeed accounts for the screening due to the core electrons.

[5] But *longer* beats *never*.

The procedure to create such additional potentials has many details, but the idea is the following:

1. For each atom of interest, use a very large set of plane waves in order to compute core orbitals accurately. This is going to be an expensive computation, but it doesn't really matter because it's for a single atom, and we will do it only once to create the pseudopotential.

2. Then, remove the core orbitals and invert the KS equations, starting with the orbitals computed in the previous step. The goal in this case is to find the hamiltonian which *would have* caused these orbitals, with the difference that we removed the core electrons. Thus, we substract all the known parts of the KS hamiltonian, and are left with a term which corrects for the removal of core electrons: the pseudopotential.

This pseudopotential can then be used on other systems using the same atom. It should be noted that as most things, it is a bit more complex: pseudopotentials are developed with specific parameters (especially exchange-correlation functionals and basis sets), and specific chemical environments. One should be careful about using a pseudopotential developed for one kind of system in a completely different kind of system. There is much research in developing *transferable* pseudopotentials, but in the end the best way to obtain them is to generate them with the preceding procedure.

One last note on terminology. Technically speaking, when using pseudopotentials, we should call Ψ the *pseudowavefunction*. Also, we replaced nuclei with particles having the same mass, but a different charge because we account for the screening of core electrons: hence we should call the nuclei *ions* instead. In practice, the terms are used interchangeably.

2.5.9 Moving atoms: geometry optimization

TODO: Less proud of this one, rewrite if time allows.

Ever since we introduced the Born-Oppenheimer approximation, we implicitly considered the nuclei to be fixed in space; and so they are when computing the electronic structure with the SCF iteration. At the end of the SCF iteration, we get the total energy of the system for *that* position of the atoms. But very often, we would like to compare the total energy of slightly different systems, say different crystal structure for the same elements. We're especially interested in the structure with the lowest energy, since it will be the most stable (at absolute zero at least).

In the case of crystalline systems, it is quite simple: just test all the possible (or reasonable) crystal structures, and find the one with the lowest energy. For more general systems, it is less obvious. You could guess for a given molecule which

atoms will bond with which atoms, but in the end, we would like to have a more systematic method of finding optimal structures.

Since we are looking for structures which minimize the total energy, a simple idea would be to compute the gradient of the total energy for various neighbouring points, and then move the atoms in the direction of the greatest energy decrease. This procedure is called *ionic minimization*: move the atoms until the total energy is minimized. This is a 3N-dimensional optimization problem, just like the electronic problem: each atom has 3 coordinates, and you want to vary those coordinates to compute the total energy of the system[6]. In general, it is not possible to guarantee that the computed structure will minimize the energy; however, you can always be sure that it is at least a local minima. Hence you usually want to start from a structure which is not too far from the "real" one. Fortunately, this can very often be done by using experimentally-derived structures, such as from X-ray diffraction.

In addition, it typically makes no sense to try find a global minima. Think of all the stable molecules you can get with just some hydrogen and some carbon: all those stable molecules would be a local minimum which can exist in some conditions. Most of the time we are starting from a system with a known structure, and study its modifications in given conditions.

2.5.10 Also moving atoms: ab initio molecular dynamics

Geometry optimization moves atoms to minize energy, but doesn't compute a physically meaningful trajectory. In particular, it doesn't account for thermal motion and more generally, instantaneous interaction between atoms. Computing the trajectory of atoms for successive, discrete time steps is called *molecular dynamics*. It simply corresponds to integrating Newton's equation for each atom, by considering the forces they exert on each other[7]. The difficult part is actually computing those forces.

In the case of DFT, we know we can compute the orbitals and associated electron density for given nuclei positions. Depending on how they overlap, these orbitals might cause attraction and repulsion between atoms; in other words, the forces exerted between atoms can be obtained from the electronic structure. This is different from *classical* molecular dynamics, in which the force is described by an empirical function. To make the distinction clear, molecular dynamics in which forces are computed by quantum chemical methods like DFT is called *ab initio*

[6] In the context of quantum chemistry, we typically don't use total energy but rather the total potential energy. This potential energy as a function of atomic coordinates is called the *potential energy surface*.

[7] At least for Born-Oppenheimer MD (BOMD). It is possible to treat the nuclei as quantum particles, though this is used only in special cases because nuclei, having a high mass, most often behave like classical particles.

molecular dynamics (AIMD)[8].

Classical MD is much more approximate than AIMD because it uses empirical functions, and there are many things it cannot describe accurately (typically, chemical reactions, when bonds between atoms change completely). It is however much more efficient than AIMD, allowing it to compute longer trajectories and for much larger systems. This is why it is often used in biochemistry, which has to deal with large molecules, while quantum chemical methods are used for smaller reactivity studies.

3 Parallel programming and supercomputers

Despite all the approximations we have made, the amount of computation for any realistic system remains way too large for a human to carry. Computers were invented precisely for that purpose. Quantum chemistry was actually one of the first use cases for early computers (using Hartree-Fock theory though, not DFT). Modern computers are impressively powerful, and it is possible to do DFT calculations on a regular computer for small molecules or small crystals. Larger systems require much more resources than can be provided by a single computer, however, hence the use of *supercomputers*.

3.1 The hardware

Supercomputers used to be large machines made of highly-specialized components, which we extremely fine-tuned to be the fastest in the market. During the 90's, the paradigm shifted to using many smaller and cheaper machines connected through a network, and programming them to collaborate on a single job. Among the reasons for this shift are the availability of low-cost standard components like CPUs and memory, as well as cheap network components.

Modern supercomputers are still based on this idea of aggregating the power of many small computers (typically called *nodes* in this context). They differ in the scale by which they do so: a single computer can easily have thousands of nodes, each with one, two or even four CPUs, in turn each having dozens of computing cores. Having fast processors is of no use if the network can't send data as fast as it is required by the computations, so modern interconnects are also extremely fast.

Those systems can be quite expensive, apparently contradicting the original goal of using cheaper components. But the initial cost of buying a supercomputer is quite small compared to the cost of running it, since they consume hundreds of megawatts of electricity.

[8]Though there is some debate wether DFT is ab initio or not. I say everything trying to solve the Schrödinger equation is ab initio.

3.2 The software

Adapting a program to run on a supercomputer is not trivial: one has to separate the computation into smaller units which can be executed on different processors. This is *parallel programming*, of which there are many kinds, depending on the hardware and the scale at which it is used. Modern processors typically contain multiple processor cores, which can be used in parallel; a programmer knowing how to use *threads* can take advantage of this parallelism. In a supercomputers, each core in each processor in each node can receive a specific task to execute, and communicate with other processors to exchange intermediate results.

This is no easy task, regarding the diversity of parallel programming paradigms available. But more importantly, parallel programming is fundamentally more complex: you execute many programs at once, and those programs need to communicate (what's the point otherwise ?). In scientific computing, two interfaces are commonly used:

- *OpenMP*[9] allows you to take advantage of the multiple cores in modern CPUs (a type of *shared-memory parallelism*). It allows control of multiple *threads of execution*, exchanging data through main memory. It may seem simple, but sharing memory raises many problems which must be managed: how to ensure data consistency if several threads can read and write it at the same time? Uncontrolled access to shared data causes *race conditions*, which must be managed by the programmer using *synchronization primitives*. Essentially, these allow you to prevent other threads from accessing data you are currently modifying. Using synchronization primitives is a delicate programming job, which can easily lead to undeterministic results or *deadlocks*.

- *MPI*[10]: the *Message Passing Interface* allows different processes to send messages to each other (a type of *distributed-memory parallelism*). As opposed to shared-memory parallelism, different tasks cannot access each other's memory directly: the data must be sent from the correct source process to the correct destination process. It is considered a more complex kind of parallel programming, but it is more efficient when the required number of processing cores is high. Processors have a limited number of cores, while nodes can be interconnected indefinitely (that is, as long as someone pays the bill). Each node has its own memory, hence the use of messages to exchange data.

  MPI is a standard specifying an interface for programmers. To actually use the MPI, you need to use an implementation, of which two are commonly

[9] https://www.openmp.org
[10] https://www.mpi-forum.org

used: *OpenMPI*[11] and *MPICH*[12].

Supercomputers have a processor hierarchy: the CPUs contain multiple cores, the nodes contain multiple processors, and the supercomputer contains multiple nodes. Many high-performance programs are written to take advantage of both shared-memory and distributed-memory programming, that is, they use both OpenMP and the MPI. It is obviously more complicated to use any of those individually, but the gain in performance can be largely justified by the cost of running the supercomputer.

4 Interacting with UNIX systems

Virtually all supercomputers run a UNIX-based operating system[13], among which Linux is the most prominent. A good reason for this is that supercomputers tend to be highly specialized machines, from which you want to obtain the best possible performance; an open source system like Linux allows this kind of fine-tuning. It is an added bonus that Linux is also free, which isn't a luxury when you consider the already-high cost of such machines (without even mentioning their energy consumption).

For similar reasons of efficiency, a supercomputer is rarely accessed though a *graphical user interface* (GUI), but rather with a *command-line interface* (CLI, also called a *terminal*). In a GUI, you get a program to execute actions by entering values (such as text or numbers) in graphical elements, and clicking on various buttons to trigger the action. In a terminal, you type *commands* on a single line, and once you are confident you have written the right command, you execute it by pressing `Return`. The terminal often seems puzzling to newcomers, especially if they have no previous programming experience. But in fact, there is a very consistent logic in using the terminal. Learning just a few commands will already give you significant power, and once you *get it*, learning new ones is a easy.

The main point of using the terminal is not machine efficiency though, but human efficiency: you often need to automate tasks in scientific computing, and the terminal makes this straightforward. For instance, you typically have a script which contains commands to launch your simulation with a given set of parameters, and you are very likely to launch it many times with slighly different parameters. Imagine doing this with a GUI: each time you want to execute your program, you would have to click, select and type-in the parameters for your simulation. In the end, the slightly steeper learning curve of the terminal is largely paid back in time and absence-of-frustration later. Hence, this section gives you the minimal

[11] https://www.open-mpi.org, which despite the name has nothing to do with OpenMP

[12] https://www.mpich.org

[13] At https://top500.org/statistics/list/, select the "Operating system family" list and judge for yourself.

set of tools you need to know in order to use the terminal.

4.1 The terminal and some useful commands

The terminal expects you to type in commands by displaying a *prompt*, commonly represented by a dollar sign:

```
$
```

You should not type in the $ symbol when you want to execute a command. It is just an invitation of the terminal for you to execute commands. Be aware that the prompt is customizable, so different computers are likely to have different prompts. Prompts commonly include the machine's name, as well as the current working directory (see later).

There is a large variety of commands available, some being installed by default on some systems, some installed by an administrator, some possibly installed by yourself. There is a minimal set of commands which is almost standard, and so you can expect to have those on most UNIX systems, including Linux. The following sections describe a few of these most important commands.

4.1.1 `pwd`: *program working directory* and filesystem paths

The *current directory* (or *program working directory*, PWD) is your current location in your file system. By default, when logging into a system or opening a terminal, you will usually end up in your *home* directory. Each user has its home directory, which can only be accessed by its owner (and the administrator if this not your machine; you can't do whatever you like on national supercomputers, oddly enough).

To see the path of the program's working directory, you should type the following command:

```
$ pwd
```

followed by a `Return`, in order to execute it. It will return you the path of the current directory, such as:

```
-> /home/nperrea
```

Here again, the arrow symbol -> just indicates this is something output by the terminal. It is just a convention, since in practice most terminals don't print anything at the beginning of an output line.

The path which was returned by `pwd` is actually an *absolute path*, which describes the path of a file or directory from the root of the filesystem. There are also *relative paths*, which reference a file or directory from the current directory. Absolute paths always start with a / symbol, while relative paths never do. There-

fore, if the /home/nperrea folder contains a file named hello_world.c, it can be referenced by the following absolute path:

```
/home/nperrea/hello_world.c
```

Or, if the current directory (output by the pwd command) is /home/nperrea, simply by:

```
hello_world.c
```

4.1.2 cd: *change directory*

The cd command allows you to change the current directory, by providing a path as argument:

```
$ cd new_directory_path
```

where *new_directory_path* can be absolute or relative. The following example changes the current directory from /home/nperrea to the VASP_praticals subdirectory:

```
$ pwd
-> /home/nperrea
$ cd VASP_practicals
$ pwd
-> /home/nperreau/VASP_practicals
```

Notice that cd doesn't return any output. It is a common convention in the UNIX world, that commands do not return anything when all went well.

In addition to absolute and relative paths, there are two special paths which are extremely useful. A single dot ('.') refers to the current directory while two dots ('..') refers to the parent directory. Referring to the current directory is sometimes useful, for example when executing programs located in the current directory:

```
$ ./some_program
```

This is a safety feature; when just typing

```
$ some_program
```

the operating system will look for a program named some_program at various well-known locations, just as it does for the cd and pwd commands we have seen so far. A malicious user could replace this some_program by another program having a different effect, like destroying your hard drive. Notice that Microsoft® Windows® *does not* have this feature.

Referring to the parent directory allows you to revert the effects of a previous `cd` command, which otherwise would only allow you going down the filesystem hierarchy. Thus, applied to the previous example, we would have:

```
$ pwd
-> /home/nperrea/VASP_practicals
$ cd ..
$ pwd
-> /home/nperrea
```

4.1.3 `ls`: listing the contents of a directory

Of course, you do not need to remember the whole file hierarchy of your computer in order to use `cd` and other commands. Without any argument, the `ls` command lists the files and directories contained in the current folder.

It is however more useful when using some options, also called *switches*. By convention, an option to a command can be specified by starting it with one hyphen ('-') or two ('--'). Being a convention, there is some variation among systems, but usually a single hyphen introduces so-called *short-options*, which are specified with a single letter, and two hyphens introduce long options which usually describe the purpose of the option.

For instance, `ls` has a short '`l`' option displaying the output as a list instead of a table:

```
$ ls -l
```

It also has long options. The `help` long is commonly implemented in many commands, and is used to print a documentation of what the command can do, among which all the options it can take:

```
$ ls --help
```

Some useful options for `ls` are:

- '`l`', to print a list instead of a table ;
- 'h', to print files sizes in easily readable units instead of the size in bytes ;
- 'a', to also print the hidden folders and files.

Short options can be combined together in a single switch; that is, the following two commands print the same output:

```
$ ls -l -h -a
$ ls -lha
```

Since the 'lha' switches are so commonly used, many systems have an alias command named `ll` for `ls -lha`.

4.1.4 `cp`: copy files and directories

Another very handy command is `cp`, allowing you to copy files from one location to another. It is used in the following way:

```
$ cp source_path destination_path
```

where you should substitute `source_path` and `destination_path` with the desired locations, either as absolute or relative paths. Notice that is a common convention among terminal commands, that source operands precede destination operands.

You can also copy several files at once, in which case the destination is necessarily a directory:

```
$ cp source_A source_B source_C destination_directory
```

When you want to copy entire directories (that is, the directory named by a path, and its subfiles and subdirectories, and their subfiles and subdirectories, and so on), you have to use the `-r` flag (as in *recursive*):

```
$ cp -r source_directory destination_directory
```

This will copy all the files and subfiles contained in `source_directory` to `destination_directory`.

4.1.5 `mv`: move files and directories

To move files and directories, you can use the `mv` command, which works like `cp`, except the source files will not remain afterwards. It is easy to inadvertently erase files without knowing using `mv`, so it is always a good idea to use it with the `v` (verbose) and `n` (don't overwrite) switches:

```
$ mv -vn source_path destination_path
```

4.1.6 `mkdir`: creating directories

You should of course organise your files in a clever hierarchy. In addition, commands like `cp` will not create directories if they do not exist. To create a directory, you can use the `mkdir` command:

```
$ mkdir new_directory
```

4.1.7 `rmdir`: creating directories

This command is the opposite of `mkdir`: it removes a directory. You should use it with caution, since this will permanently remove the directory instead of moving it into a trash (which is possible, but with system-specific commands). Also, it will not work if the target directory is not empty; it is a safety measure, preventing you from deleting whole file hierarchies without knowing.

To remove a directory, use:

`$ rmdir` *`target_directory`*

4.1.8 `rm`: remove files, directories, and wreak havoc

The `rm` command is probably the most dangerous you will encounter. It can delete files, as well as non-empty directories (as opposed to `rmdir`) and complete file hierarchies. Use it with *extreme* caution.

To remove a file, you can use:

`$ rm` *`some_file`*

To a remove a directory, you can use the 'r' switch, just like for `cp`:

`$ rm -r` *`goodbye_folders`*

Just like `rmdir`, there is no going back if you accidentally delete files. Some useful switches are:

- 'i': always ask confirmation before deleting any file ;
- 'v' (*verbose*): print information on exactly which files are being removed.

4.1.9 `man`: print documentation about a command

Many commands have the `help` option (or similarly, a 'h' short option) to print some documentation when in doubt. More extensive documentation can be found in the venerable *man pages*, short for *manual pages*, through the `man`[14] command. To display documentation about the `ls` command, simply type:

```
$ man ls
```

This will launch a text interface in which you can navigate using the arrow keys, and which you exit by typing 'q'. It is always a good idea to check the man pages when you are introduced to a new command. As a suggestion, it would be wise for this practical work to see what it can teach you about `ssh` and `scp`:

[14] No sexism intended.

```
$ man ssh
$ man scp
```

4.1.10 Interrupting a command

Sometimes, you start executing a command, but finally change your mind, or notice that you made a mistake and absolutely have to stop the command from running to completion. You can interrupt a command's executing with the combined `Ctrl-C` keys for most commands. Some other commands, typically interactive ones (like the Python interpreter), require instead the `Ctrl-D` combination. In general, you should try `Ctrl-C` first, since `Ctrl-D` might also close your terminal.

Also, if you start typing a command but change your mind, you can press `Ctrl-C`; it will interrupt it and present a fresh prompt.

4.1.11 Wildcards

The terminal allows you to match several files or directories by using *wildcards*: special symbols having a special meaning. In general, you can use so-called *regular expressions* to match the names of your files, but to keep things simple, here are just the two most common wildcards:

- '`*`' (a *star*) allows you to match *anything*. For instance, typing the command `ls *.txt` will list all the files ending with `".txt"`.
- '`~`' (a *tilde*) is a shortcut to your *home* directory, which is the name of a user's directory in UNIX systems. It is useful when your are somewhere in the hierarchy and need to come back to a well known place, using for instance `cd ~`. It is also useful when transferring files: instead of putting them in some well-known location, you just put them at `~` and refer to them later using the short path `~/my_file.txt`.

4.1.12 Command history and autocompletion

Two nice features implemented in most terminals is the *command history* and *autocompletion*. The first allows you to recall previously-typed commands using the up and down arrow keys. It is especially useful when typing long commands involving long paths or addresses, such as `ssh` and `scp` commands.

Autocompletion can complete command names and even paths while typing commands, by using the tab key. If you press it once, it can either complete the current part of the command you are typing if it can choose unambiguously, or display nothing. In the second case, you can hit tab twice in a row, and this will list all the possibilities it found. This allows you to complete the command a bit

until it can choose among the possibilities unambiguously, at which point you can press tab again to let the terminal complete it for you.

It is much more intuitive than it sounds.

4.2 Working with remote computers: SSH and X forwarding

Most of the time, you don't access supercomputers by going there and plugging in a keyboard and a screen. That would even be a very bad idea considering the security surrounding some of these computers. Instead, you can access these machines *remotely*, and the most common tool to do so in the UNIX world is the *secure shell* (SSH). It is a terminal which will connect securely to a remote machine, if it allows you to, and execute the commands you type on a local terminal on the remote terminal. For the most part, it behaves *as if* you were typing thoses commands directly on the remote machine.

4.2.1 The *Secure Shell* (SSH)

What you have learned so far about the terminal should work just as well using SSH, at least once you have established a connection. However, a common source of confusion is that you are given a terminal on the remote computer, and hence you work on the remote system. This means that some commands available locally might not be available on the remote computer, and vice versa. In fact, you can even connect to a Linux computer with a Windows computer, though it can be a bit troublesome.

To use SSH, you need to have a user account and associated password on the remote machine. In addition, in order to contact the remote machine, you need to locate it, which can be done with a domain name or an IP address. For instance, if there is a user *hohenberg* on a machine identified by IP address `172.20.134.7`, you would open the SSH session using:

```
$ ssh hohenberg@172.20.134.7
```

You will then be asked to type in your password and validate. If allowed, the session will then open, and you will then be able to type commands on the remote terminal. You can try to execute the command `hostname`, which will give you the name of the machine; if you execute it locally, you should obtain a different name.

4.2.2 `scp`: secure copy

You very often need to send data to the supercomputer for the program to use, and retrieve results once it is done. In quantum chemistry, this data would typically represent the parameters to use, as well as the position and kinds of atoms of your

system of interest. Sometimes these files might be small, but with computations like molecular dynamics those can easily reach the gigabytes. Hence, you will need to transfer files back and forth all the time.

This can be done with the `scp` command, which is a mix between `cp` and `ssh`, and has to be executed on a terminal on you *local* machine:

```
$ scp hohenberg@172.20.134.7:/remote/path /local/path
```

which will prompt you for your password before starting the transfer. In this command, `/remote/path` is the path to a file or directory on the *remote* server, from which we want to download, and `/local/path` is the path on the *local* machine where we want to store the file or directory. As for the `cp` command, the source always precedes the destination, so this example indeed downloads a file from the remote machine to your local machine. To upload a file, you can simply exchange the source and the destination. Be aware though, that just like `cp`, it will overwrite files silently if they are already present (except if using an option to prevent this).

When downloading a single file, this command can be used directly, but you may want to download a full directory with all its subdirectories, or a set of files all at once. In the first case, you can use the `-r` switch (`r` for *recursive*, the same of for `cp`):

```
$ scp -r hohenberg@172.20.134.7:/some/directory /local/directory
```

while in the second case, you can specify a list of files enclosed with *escaped* curly braces. This means you type `\{` instead of just `{`, because otherwise the terminal will interpret the curly braces differently. So, to transfer several files at once, the command will typically look like:

```
$ scp user@172.20.134.7:\{file1,file2,file3\} /local/directory
```

You can also use the `-r` switch in this command, so that you can also download several directory hierarchies at once. A useful command is `readlink`, which allows you to display the absolute path of a remote file on the server:

```
$ readlink -f ./some_file
```

You can copy-paste the resulting path and substitute it for `remote_path`; this way, you won't have to care about relative paths and working directories.