Notes for the Computational Biophysics class

Lorenzo Rovigatti

November 20, 2024

Contents

Chapter 1

Basic notions

1.1 Introduction to the course

As with most computational courses, this course is supposed to have a practical side that should not be overlooked. However, as some of you may have noticed, there will be frontal lessons only. While this may seem contradictory (and in some sense it is), it also means that you are strongly advised to practice on your (or someone else's) computer what you'll hear about (and be shown) during the lectures. Moreover, I will also set up some hands-on (bring-your-own-laptop) lectures to guide you through the most important technical hurdles we will be encountering.

The material that form the bulk of the lectures is based on these books:

- Lehninger et al. [2005]: a bible of biochemistry. Very useful as a reference for the basic biochemistry reactions involved in all biological systems.
- Finkelstein and Ptitsyn [2016]: protein physics in a nutshell. It is based on a series of lectures that the authors have been delivering for years (if not decades). It is very comprehensive, and uses a informal approach that I find very compelling.
- Leach [2001]: principles of molecular modelling, both quantum and classical. Perhaps a bit outdated in some parts, but still a very useful resource for a general introduction to modelling molecular interactions.
- Schlick [2010]: also on molecular modelling, but oriented towards nucleic acids and proteins. Very useful as a crash course to DNA, RNA, and proteins, as well as to the way the are modelled with a computer.
- Frenkel and Smit [2023]: the bible of molecular dynamics and Monte Carlo simulations. I use it to introduce the Monte Carlo algorithm and the basic molecular dynamics techniques.
- Israelachvili [2011]: an incredible (and comprehensive) book on intermolecular forces. For our class, it is especially useful to understand van der Walls and hydrophobic forces
- Giustino [2014] and Böttcher and Herrmann [2021]: I did not read them front-to-back, but only used them as sources for the Molecular quantum mechanics chapter.

1.2 About these notes

I have prepared these notes mainly for myself, but I decided to share them because I think they can be useful to others, if used correctly. Everything that is in here sould be regarded as my take on the topics I present, and therefore should not be trusted. The truth can be found in the books I used as sources, or in the original papers, both of which are always referenced.

Here are some useful things to know:

- Internal links are styled Section 1.4. Hover a link to see a preview.
- Wikipedia links are styled like this. Hover a wikipedia link to see a preview.
- External links are styled like this.
- Hover over a footnote reference to show the associated text[1].

[1] like this.

- Hover over a reference to show the associated citation and a clickable DOI (see *e.g.* Anderson [1972]).
- Most of the acronyms that are scattered throughout the text show tooltips when hovered with the mouse cursor, like HPC.

A box
There are some boxes scattered throughout the text. Their colour weakly correlates with the content: pay particular attention to the yellow and red ones!

1.2.1 Prerequisites

1. This is a computational course, and as such it requires some proficiency with (or at least having the right attitude towards) computers. The most important skill you will need is to use a terminal, since most of the computations (running simulations, analysing results, *etc.*) will be launched from there. Linux and macOS come with pre-installed terminals, while Windows does not[2]. However, it is possible to install a very handy "Windows Subsystem for Linux" that makes it possible to have a good Linux-like experience within Windows. Install instructions can be found here.

2. Knowing to code is important, for several reasons. First of all, implementing algorithms and techniques introduced in class is the best way of understanding how they work and what are their advantages and disadvantages, even though most of research-grade results during one's career will be obtained with production-ready HPC codes written by others. Secondly, reading other people's code can come in very handy (i) to understand what it does and (ii) to extend it to suit your needs. Finally, in computational physics it is common to need to perform custom analyses for which no libraries or codes are available, leaving no other option than writing your own script or program. For the sake of the course, the programming language you are more familiar with is not important, although Python is the *de facto* standard language used to write analysis scripts, while C and C++ (and more rarely FORTRAN) are used to write performance-critical software (or part thereof).

3. By definition, life is an out-of-equilibrium process, since it continuously uses up energy. However, many biophysical processes are in (or close to) equilibrium. Therefore, they can be analysed and understood using the language of thermodynamics and statistical mechanics. You should be familiar with concepts such as free energy, entropy, ensembles, partition function, Boltzmann distribution, as well as with the mathematical tools used to work with these quantities.

4. *Any* knowledge of biology and biochemistry is welcome (and will be useful to its holder). However, I will introduce most of what we need at the beginning, and then some more as the need arises.

1.2.2 Evaluation

Your final grade will be based on:

- the final project (40%)
- the part of the individual oral exam devoted to discussing the final project (30%)
- the part of the individual oral exam where you will be asked questions about the lectures OR the homework assignements carried out during the course (30%)

About the last bullet point, during the course you will be asked to complete three homework assignments. Those are not mandatory, but carrying them out satisfactorily will allow you to skip the questions about the course program during the oral exam.

[2]Or at least it does not come with a Unix-like (POSIX) terminal, which is what we will be using.

Homework assignments

More or less every couple of weeks I will ask you to do a homework assignment in which you will have to write or use a code that does something "useful", connected to what explained in the classroom. These assignments are not mandatory, but if you carry out enough of them in a satisfactorily manner, you will be able to "skip" part of the oral exam where you would be asked questions about the frontal lessons.

The output of each assignment should be:

1. A code, written in any (sensible) language (*e.g.* C, C++, Python, Fortran, Java, but other languages may be also fine, just let me know beforehand).
2. A short (but not too short!) document containing the documentation of your code, and a presentation of the results asked in the assignment.
3. Each assignment has a mandatory part and a few "possible extensions". The "possible extensions" can be implemented to improve the rate of the assignment, or can be used as suggestions for the final project: you are free to extend your code or your analysis in any way you like it, provided it makes sense from the computational and biophysical points of view.

The code documentation should be clear and concise, but also list features, limitations, and known bugs of your code. In addition, it should present a comprehensive guide about how to you use the code (*e.g.* how to call it, how to specify the input, how to read the output, *etc.*). I will provide examples of "good" documentation during the course.

Using AI tools

You are welcome to use ChatGPT, GitHub Copilot or similar tools during the course. However, harsh penalties will be handed out to students who use AI-assisted tools without actually understanding their answers[3]. This applies to both code and text submitted for the homework assignments and for the final project.

1.3 Introduction to biophysics

Biophysics is an interdisciplinary science that applies the principles and methods of physics to understand biological systems. It bridges the gap between biology and physics by using quantitative approaches to study the structure, dynamics, function, and interactions of biological systems, from single molecules, to cells, whole organisms and even ecosystems.

Biophysics involves the investigation of the physical principles governing biological processes. This includes understanding how forces and energy transfer within and between cells, the mechanics of biomolecular interactions, and the dynamics of complex biological networks. At larger length-scales, biophysics also encompasses the study of extended biological systems, such as the mechanics of muscle contraction, neural signal transmission, and the properties of biological membranes. By integrating concepts from (equilibrium and non-equilibrium) thermodynamics, statistical mechanics, and fluid dynamics, biophysics provides a comprehensive framework for understanding life at all levels of organization.

Defined in this way, "biophysics" is an umbrella term whose utility is questionable. It becomes more useful if one acklowedges that phenomena at very different scales share features and properties that make it possible to use tools developed to investigate "classic" physical systems (such as statistical mechanics). In fact, one of the main endeavours of biophysics, which is one of the reasons why proper biologists often scorn biophysicists, is to find unifying principles that can be used to explain or predict different phenomena.

Source
Some parts of this section have been adapted from here.

1.3.1 Some definitions

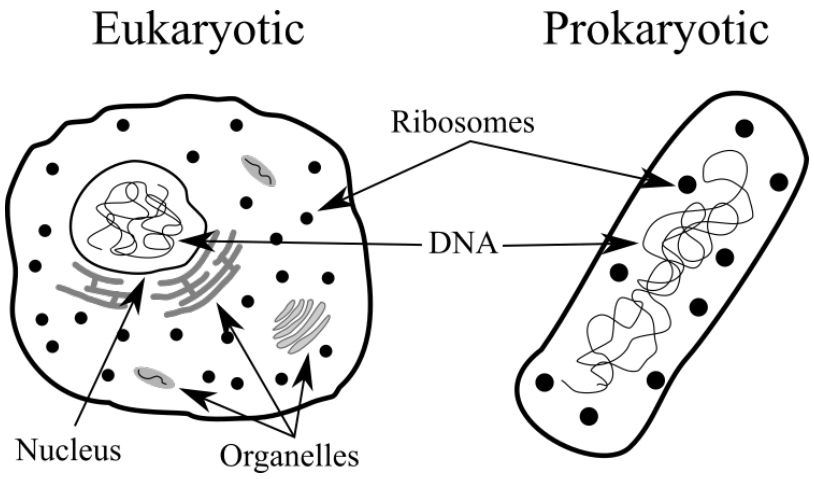

Figure 1.1: The main differences and similarities between eukaryotic and prokaryotic cells. The cells have not been drawn at scale.

Cell A cell is the basic structural and functional unit of all living organisms. It is the smallest unit capable of carrying out the processes necessary for life, such as energy production, metabolism, and reproduction.

Organelles Specialized structures within cells that perform distinct functions necessary for cellular life. Organelles make it possible to carry out tasks in isolation from the rest of the cell, ensure that the cell operates efficiently and responds to its environment.

Procaryotes Unicellular organisms that lack a true nucleus and membrane-bound organelles. Their genetic material is located in a nucleoid, and they typically have a single, circular chromosome along with plasmids. Prokaryotic cells have a rather small size ($0.1 - 5.0\,\mu$m) and reproduce through binary fission[4]. Most have a rigid cell wall made of peptidoglycan (in bacteria). Examples of prokaryotes include bacteria and archaea.

Eukaryotes Organisms whose cells contain a true nucleus enclosed by a nuclear membrane and various membrane-bound organelles such as mitochondria, endoplasmic reticulum, and Golgi apparatus. Their DNA is linear and organized into chromosomes within the nucleus. Eukaryotic cells are generally larger than prokaryotes ($10 - 100\,\mu$m) and reproduce through mitosis for somatic cells and meiosis for gametes. While some eukaryotes have cell walls (*e.g.*, plants and fungi), others do not. Examples of eukaryotes include plants, animals, fungi, and protists.

1.3.2 Macromolecules and polymers

A macromolecule is a molecule composed by a great number of covalently bonded atoms[5]. The most common class of macromolecules is that of polymers, which are molecules composed by smaller subunits, the *monomers*, covalently linked together. If the monomers are all of the same type, the resulting molecule is a *homopolymer*, while if they are different, the molecule is a *heteropolymer*. In

[4]For what concerns us, this is a simpler version of mitosis.

[5]"great number" is a purposedly vague qualifier: there is no strict definition about the number of atoms required for a molecule to be dubbed a macromolecule.

general, monomers can be connected in different ways, giving raise to unidimensional structures such as chains or rings, or to more complicated topologies such as brushes, stars, networks, *etc.*

Focussing on chains, the number of repeating units (also known as *residues*) composing a polymer is called *degree of polymerisation* N, and for common plastic materials is rather large ($N \sim 10^3 - 10^5$). The simplest polymer is the hydrocarbon polyethylene, $(-CH_2-)_n$, which is used to make cheap bags and bottles and accounts for more than 30% of the plastic produced worldwide (see *e.g.* Geyer et al. [2017]). Other very common polymers used to build everyday objects are polypropilene, $-CH_2 - CH(CH_3)-$, which is heat- and fatigue-resistant and therefore used to make hinges, piping systems, containers, and polystyrene, $-CH_2 - CH(C_6H_5)-$, used to make plastic cutlery, containers or insulating foams. The skeletal formulas of these three polymers are shown in Figure 1.2.

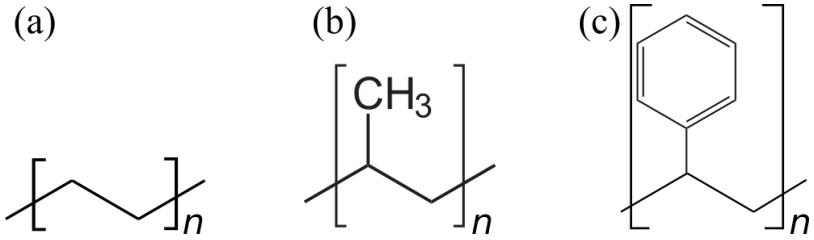

Figure 1.2: The skeletal formulas of (a) polyethylene, (b) polypropilene and (c) polystyrene. Here the repeating unit in (a) is $-CH_2 - CH_2-$ to make it more easy to compare it with the other two.

In the biological context, three of the four main macromolecular components of life are polymers: proteins, nucleic acids and carbohydrates, also known as biopolymers. While there exist (bio)polymeric substances with more complex architectures, the main macromolecules of life have a linear (chain) structure that makes it possible to assign a one dimensional *sequence* to each molecule. This sequence is just the list of monomers composing the chain, spelt from one chain end to the other.

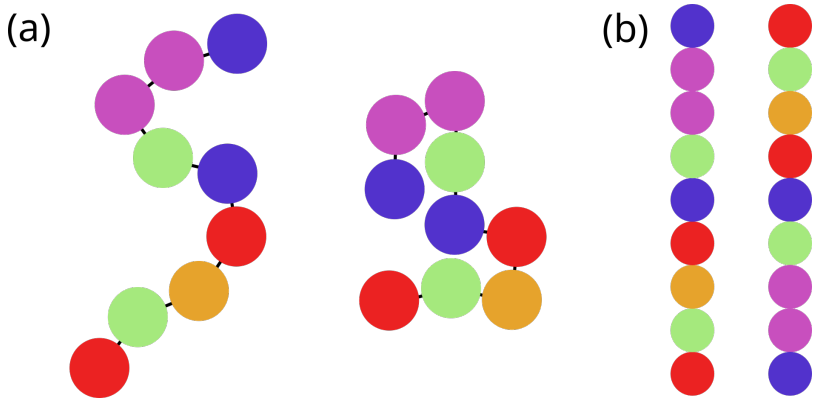

Figure 1.3: A cartoon of a polymer composed by 9 monomers (the coloured spheres) connected by covalent bonds (black lines). Panel (A) shows two 2D conformations, panel (B) shows the two 1D sequences built by listing the monomers that make up the chain, starting from either end.

Figure 1.3 shows an imaginary short polymer composed by 9 monomers of different nature (coloured differently). In general, there are many different spatial arrangements that the same (bio)polymer can take in solution. By contrast, its sequence is fixed, being given by the list of covalently-linked monomers. However, as shown in the figure, in absence of any convention, the sequence can be read from either end, giving raise to an ambiguity. As we will see, there exist conventions for proteins and nucleic acids that get rid of this ambiguity.

Some useful observables

Source
This section has been adapted from the Soft and Biological Matter notes by Prof. Sciortino. See also the bible of polymer physics, Rubinstein and Colby [2003].

First, some definitions.

End-to-end vector, $\vec{R}_{ee}$ The vector distance between the first and last residue of a polymer chain. Its length is the end-to-end distance, R_{ee}.

Contour length, $R_{\mathbf{max}}$ The maximum possible end-to-end distance, which is achieved for a fully-extended polymer chain.

Monomer position, $\vec{R}_i$ The position of the i-th monomer.

Bond vector, $\vec{r}_i$ The vector connecting the $(i-1)-th$ and i-th residues, defined as $\vec{r}_i = \vec{R}_i - \vec{R}_{i-1}$.

Ideal chain A chain in which two residues that are far from each other, *i.e.* residues i and j for which $|i-j| \gg 1$, do not interact.

Chemical distance The number of bonds separating two monomers along the chain, *i.e.* $|i-j|$ for monomers i and j.

Consider a polymer chain composed by $N = n+1$ monomers connected by n bonds. Its instantaneous end-to-end vector is

$$\vec{R}_{ee} = \sum_{i=1}^{n} \vec{r}_i. \tag{1.1}$$

Consider the ensemble average of this quantity, $\langle \vec{R}_{ee} \rangle$, which denotes an average over all possible states of the system (accessed either by considering many chains or many different conformations of the same chain). In this particular case the ensemble average corresponds to averaging over an ensemble of chains having n bonds, with all possible bond orientations. Since there is no preferred direction in this ensemble, the average end-to-end vector is zero[6]. A simple non-zero average that can be built out of the end-to-end vector is

$$\langle \vec{R}_{ee}^2 \rangle = \langle \vec{R}_{ee} \cdot \vec{R}_{ee} \rangle = \left\langle \left(\sum_{i=1}^{n} \vec{r}_i \right) \cdot \left(\sum_{j=1}^{n} \vec{r}_j \right) \right\rangle. \tag{1.2}$$

If the bond vectors are all of the same length l (which is often a good approximation, given the rigidity of the backbone covalent bonds), $\vec{r}_i \cdot \vec{r}_j = l^2 \hat{r}_i \cdot \hat{r}_j = l^2 \cos\theta_{ij}$, where $\cos\theta_{ij}$ is the angle between $\vec{r}_i$ and $\vec{r}_j$, and the mean-squared end-to-end distance can be written as

$$\langle \vec{R}_{ee}^2 \rangle = l^2 \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \cos\theta_{ij} \rangle. \tag{1.3}$$

Note that in this case the contour length has the simple expression $R_{\max} = nl$.

In the simplest polymer model there is no correlation between *any* two monomers i and j. In such a *freely-jointed chain* the average cosine vanishes if $i \neq j$, since

$$\langle \cos\theta_{ij} \rangle = \frac{\int_0^{\pi} \cos\theta \sin\theta d\theta \int_0^{2\pi} d\phi}{\int_0^{\pi} \sin\theta d\theta \int_0^{2\pi} d\phi} = -\frac{1}{4} \cos^2\theta \Big|_0^{\pi} = 0, \tag{1.4}$$

and therefore the double sum in Eq. (1.3) becomes a single sum of n ones, yielding

[6] You can find the same result by realising that an ideal polymer is just a random walk of n steps: if the random walk is not biased, the probability of moving towards a direction is independent of the directions taken before, and therefore, on average, the walker does not move.

$$\langle \vec{R}_{ee}^2 \rangle = nl^2. \tag{1.5}$$

However, in a typical ideal chain, $\langle \cos\theta_{ij} \rangle = 0$ only if i and j are sufficiently far apart from each other. In this case, if we assume that there is a maximum chemical distance m beyond which $\langle \cos\theta_{ij} \rangle = 0$, we can write the inner sum of Eq. (1.3) as

$$\sum_{j=1}^{n} \langle \cos\theta_{ij} \rangle = \sum_{j=1}^{m} \langle \cos\theta_{ij} \rangle \equiv C_\infty, \tag{1.6}$$

where $C_\infty > 1$[7], the so-called *Flory's characteristic ratio*, accounts for the local monomer-monomer correlations due to steric hindrances and hampered rotations around chemical bonds and varies from polymer to polymer. For these ideal chains, Eq. (1.3) can be written as

$$\langle \vec{R}_{ee}^2 \rangle = l^2 \sum_{i=1}^{n} C_\infty = C_\infty nl^2. \tag{1.7}$$

Flexible polymers have many universal properties that are independent of the local chemical structure, and they can all be described in terms of equivalent freely-jointed chains. The equivalent chain has the same mean-squared end-to-end distance and contour length, but a different number of effective beads N of length b, chosen to match the values of $\langle \vec{R}_{ee}^2 \rangle$ and $R_{\max}$:

$$R_{\max} = Nb \tag{1.8}$$

$$\langle \vec{R}_{ee}^2 \rangle = C_\infty nl^2 = Nb^2, \tag{1.9}$$

so that

$$b = \frac{\langle \vec{R}_{ee}^2 \rangle}{R_{\max}} \tag{1.10}$$

$$N = \frac{R_{\max}}{b}. \tag{1.11}$$

The effective bond length b is known as *Kuhn's length*, and it represents the size of a segment that behaves as a freely-jointed monomer in the equivalent chain.

The size of a linear chain is well-described by the square root of its mean-squared end-to-end distance, $\sqrt{\langle \vec{R}_{ee}^2 \rangle}$. However, in some cases this quantity is not well defined (*e.g.* for ring or branched polymers), or it is not easily accessible in experiments. In these cases it is useful to define the radius of gyration, which can be computed for any set of atoms or particles:

$$\vec{R}_g^2 = \frac{1}{N} \sum_{i=1}^{N} (\vec{R}_i - \vec{R}_{\rm cm})^2, \tag{1.12}$$

where $\vec{R}_{\rm cm} = \frac{1}{N} \sum_{j=1}^{N} \vec{R}_j$ is the position of the centre of mass of the polymer. Sometimes (also in simulations), it is not convenient, or possible, to compute the centre of mass. For these cases, Eq. (1.12) can be rewritten in another form by substituting the definition of $\vec{R}_{\rm cm}$, obtaining

$$\vec{R}_g^2 = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{R}_i^2 - \vec{R}_i \cdot \vec{R}_j) = \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{R}_i - \vec{R}_j)^2 \tag{1.13}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{j>i}^{N} (\vec{R}_i - \vec{R}_j)^2, \tag{1.14}$$

[7] It is larger than one since $\cos_{ii} = 1$, and the $i \neq j$ terms are all positive.

where we have first completed the square of the binomial by duplicating the double sum (hence the factor of 2 at the denominator), and then run the inner sum on monomers having index $j > i$, so that each pair of monomers only enters once in the double sum. The associated ensemble average is then

$$\langle \vec{R}_g^2 \rangle = \frac{1}{N} \sum_{i=1}^{N} \sum_{j>i}^{N} \langle (\vec{R}_i - \vec{R}_j)^2 \rangle. \tag{1.15}$$

The radius of gyration and end-to-end distance are closely related. For instance, for a freely-jointed chain, it can be demonstrated that

$$\langle \vec{R}_g^2 \rangle = \frac{\langle \vec{R}_{ee}^2 \rangle}{6}, \tag{1.16}$$

which means that the two quantities scale with n in the same way. This proportionality holds also for other (more complicated models).

In general, for large enough degrees of polymerisation (*i.e.* for $n \gg 1$), polymers are scale-free objects, which means that most of their properties can be expressed as power laws in n. A special role is played by the exponent that connects n to the polymer size (*i.e.* its gyration radius or end-to-end distance), which is called ν:

$$R_{ee} \equiv \sqrt{\langle \vec{R}_{ee}^2 \rangle} \propto n^{\nu} \tag{1.17}$$

$$R_g \equiv \sqrt{\langle \vec{R}_g^2 \rangle} \propto n^{\nu} \tag{1.18}$$

where $\nu = 0.5$ for an ideal chain. Self-avoiding polymers, *i.e.* polymers whose only interaction is repulsive, have $\nu \approx 0.588$, which means that they are *swollen* with respect to ideal polymers: their linear size is larger than it would be if there was no repulsion. Compare it with the scaling of the linear size of a dense object, for which $\nu = 0.33$.

1.3.3 DNA

Deoxyribonucleic acid (DNA) is a macromolecule that stores the genetic information of an organism. DNA contains regions called genes, which encode for proteins to be produced. Other regions of the DNA contain regulatory elements, which partially influence the level of expression of each gene. Within the genetic code of DNA lies both the data about the proteins that need to be encoded, and the control circuitry, in the form of regulatory motifs.

As we will see in depth, in cells DNA is usually found in a "double-stranded" helical form, where each strand is a long chain of repeating units, called nucleotides, of just four types: A(adenine), C(cytosine), T (thymine), and G (guanine). The list of nucleotides is called *sequence* or *primary structure*. In the double strand, A pairs with T and G with C, with the A-T pairing being weaker than the C-G pairing[8].

The two DNA strands in the double helix are complementary, meaning that if there is an A on one strand, it will be bonded to a T on the other, and if there is a C on one strand, it will be bonded to a G on the other. The DNA strands have a chemical directionality, or polarity, with the convention being to list a strand's sequence with the direction along which enzymes[9] called polymerases[10] synthesise DNA and RNA, called the 5' to 3' direction. With this in mind, we can say that that the DNA strands are anti-parallel, as the 5' end of one strand is adjacent to the 3' end of the other. As a result, DNA can be read both in the 3' to 5' direction and the 5' to 3' direction, and genes and other functional elements can be found in each.

Base pairing between nucleotides of DNA constitutes its *secondary structure*. In addition to DNA's secondary structure, there are several extra levels of structure that allow biological DNA to be tightly

[8] For this reason, the genetic composition of bacteria that live in hot springs is 80% G-C.

[9] An enzyme is a protein that speeds up a specific biochemical reaction.

[10] DNA and RNA polymerases are particular proteins (or rather enzymes[11]) that synthesise nucleic acids.

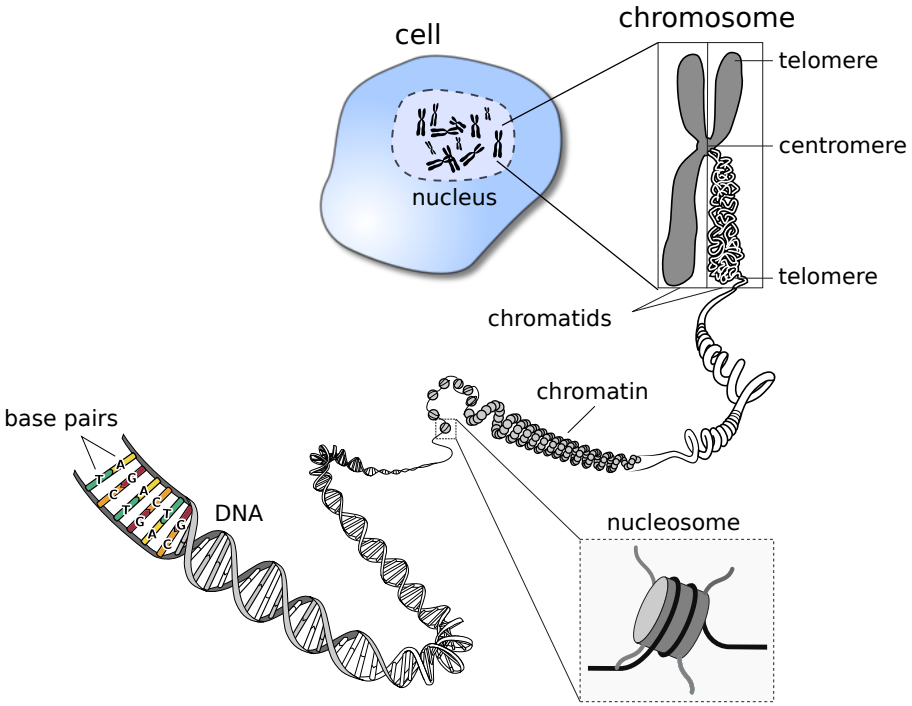

Figure 1.4: The multi-scale organisation of DNA in a eukaryotic cell. Adapted from a figure made by Phrood, via Wikimedia Commons.

compacted and influence gene expression, as schematically shown in Figure 1.4. The tertiary structure describes the twist in the DNA ladder that forms a helical shape. In the quaternary structure, DNA is tightly wound around small proteins called histones. These DNA-histone complexes are further wound into tighter structures seen in chromatin.

Before DNA can be replicated or transcribed into RNA, the chromatin structure must be locally "unpacked". Thus, gene expression may be regulated by modifications to the chromatin structure, which make it easier or harder for the DNA to be unpacked. This regulation of gene expression via chromatin modification is an example of epigenetics.

The structure of DNA allows the strands to be easily separated for the purpose of DNA replication, as well as transcription, translation, recombination, and DNA repair, among others. This was noted by WATSON and CRICK [1953] as "It has not escaped our notice that the specific pairing that we have postulated immediately suggests a possible copying mechanism for the genetic material". In the replication of DNA, the two complementary strands are separated, and each of the strands are used as templates for the construction of a new strand.

DNA polymerases attach to each of the strands at the origin of replication, reading each existing strand from the 3' to 5' direction and placing down complementary bases such that the new strand grows in the 5' to 3' direction. Because polymerases build the chains from 5' to 3', one strand (the leading strand) can be copied continuously, while the other (the lagging strand) grows in pieces which are later glued together by another enzyme, DNA ligase. The end result is 2 double-stranded pieces of DNA, where each is composed of one old strand, and one new strand; for this reason, DNA replication is a semiconservative process.

Many organisms have their DNA split into several chromosomes. Each chromosome contains two strands of DNA, which are complementary to each other but are read in opposite directions. Genes can occur on either strand of DNA. The DNA before a gene (in the 5' region) is considered "upstream", whereas the DNA after a gene (in the 3' region) is considered "downstream".

1.3.4 Transcription and RNA

When a gene product is to be expressed, the associated DNA is partially unwound to form a "bubble", where the two strands are separated. At this point transcription, which is the process by which RNA

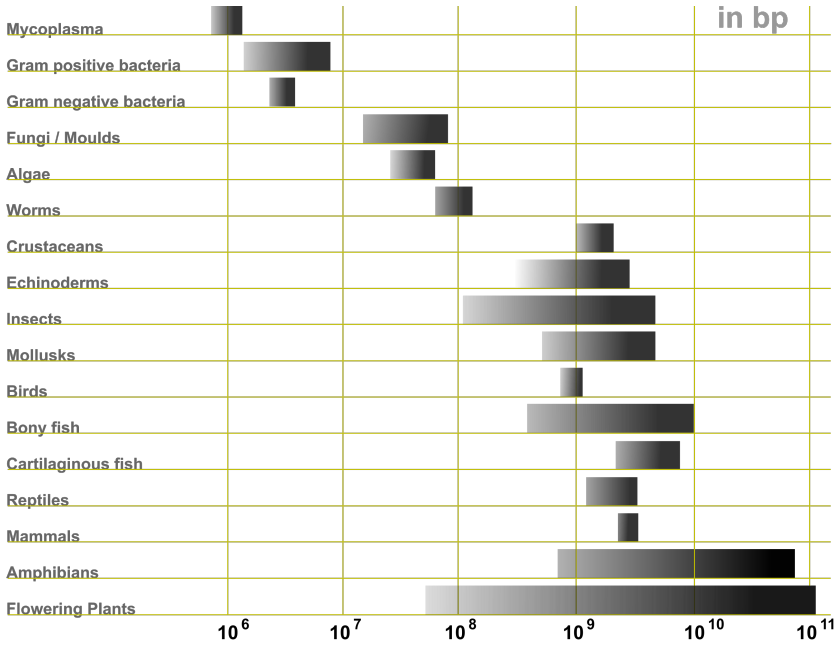

Figure 1.5: Genome size ranges (in base pairs) of various life forms. Credits to Abizar via Wikipedia Commons.

is produced using a DNA template, can commence: RNA polymerase is recruited to the transcription start site by regulatory protein complexes, and then reads the DNA from the 3' to 5' direction, placing down complementary bases to form a messenger RNA (mRNA) strand[12]. RNA uses the same nucleotides as DNA, except Uracil is used instead of Thymine.

mRNA in eukaryotes experience post-translational modifications, or processes that edit the mRNA strand further, starting from the "pre-mRNA" molecule that is complementary to the template DNA strand of a gene. Most notably, a process called splicing removes introns, regions of the gene which don't code for protein, so that only the coding regions, the exons, remain. Different regions of the primary transcript may be spliced out to lead to different protein products (alternative splicing). In this way, an enormous number of different molecules may be generated based on different splicing permutations. In addition to splicing, both ends of the mRNA molecule are processed. The 5' end is capped with a modified guanine nucleotide, while at the 3' end, roughly 250 adenine residues are added to form a poly(A) tail. In most cases, the final molecule will be used by the cell machinery (and, in particular, by ribosomes) to synthesise proteins. As we will see in a moment, the base unit of a protein is an amino acid, and the information about what amino acid to add to the nascent protein is stored in the mRNA as a three-nucleotide sequence, also known as a *codon*.

RNA is not only the intermediary step to code a protein, but RNA molecules also have catalytic and regulatory functions. Though proteins are generally thought to carry out the main cellular functions, it has been shown that RNA molecules can have complex three-dimensional structures and perform diverse functions in the cell. In 1989 the Chemistry Nobel Prize was awarded to Thomas R. Cech and Sidney Altman for their "discovery of catalytic properties of RNA", which greatly contributed to the establishment of the concept of *ribozymes* (**ribo**nucleic acid en**zymes**), which are RNA molecules that have the ability to catalyze specific biochemical reactions.

The most common RNA types are:

1. mRNA (messenger RNA) contains the information to make a protein and is translated into protein sequence.

2. tRNA (transfer RNA) specifies codon-to-amino-acid translation. It contains a 3 base pair anticodon complementary to a codon on the mRNA, and carries the amino acid corresponding to its anticodon attached to its 3' end.

[12]Since the template strand is read 3' → 5', the nascent mRNA is built in the 5' → 3' direction.

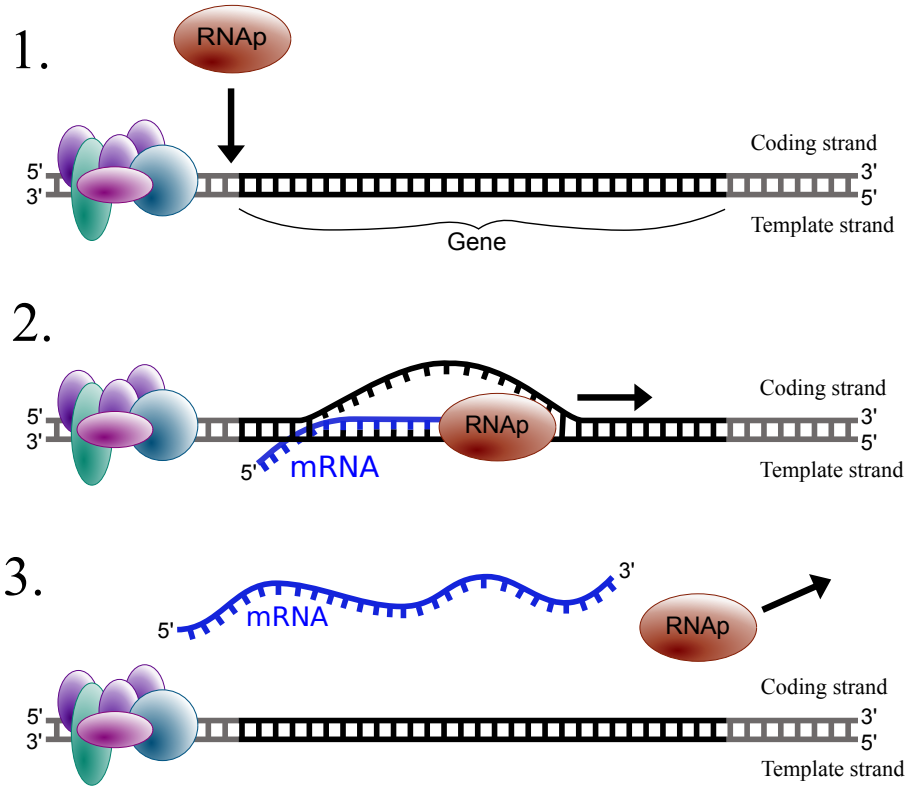

Figure 1.6: A sketch showing three steps that lead to the synthesis of an mRNA: RNA polymerase (RNAp) attaches to the transcription start site (transcription initiation) and start building an RNA chain that is fully complementary to the DNA template strand (transcription elongation). When the gene has been fully transcribed, the RNAp and mRNA both detach and leave (transcription termination). Adapted from here, here, and here.

3. rRNA (ribosomal RBA) forms the core of the ribosome, the organelle responsible for the translation of mRNA to protein.

4. snRNA (small nuclear RNA) is involved in splicing (removing introns from) pre-mRNA, as well as other functions.

The discovery of rybozymes contributed to the "RNA world" hypothesis, for which early life was based entirely on RNA. RNA served as both the information repository (like DNA today) and the functional workhorse (like protein today) in early organisms. Protein is thought to have arisen afterwards via ribosomes, and DNA is thought to have arisen last, via reverse transcription.

1.3.5 Translation and the genetic code

Translation is the process by which the information encoded in mRNA is used to synthesize a protein. Unlike transcription, where the nucleotide sequence in DNA is transcribed into a corresponding nucleotide sequence in RNA, translation involves converting this nucleotide sequence into an amino acid sequence, forming the primary structure of the protein. Since there are 20 different amino acids and only 4 nucleotides, the mRNA is read in sets of three nucleotides, known as codons, each of which specifies a particular amino acid.

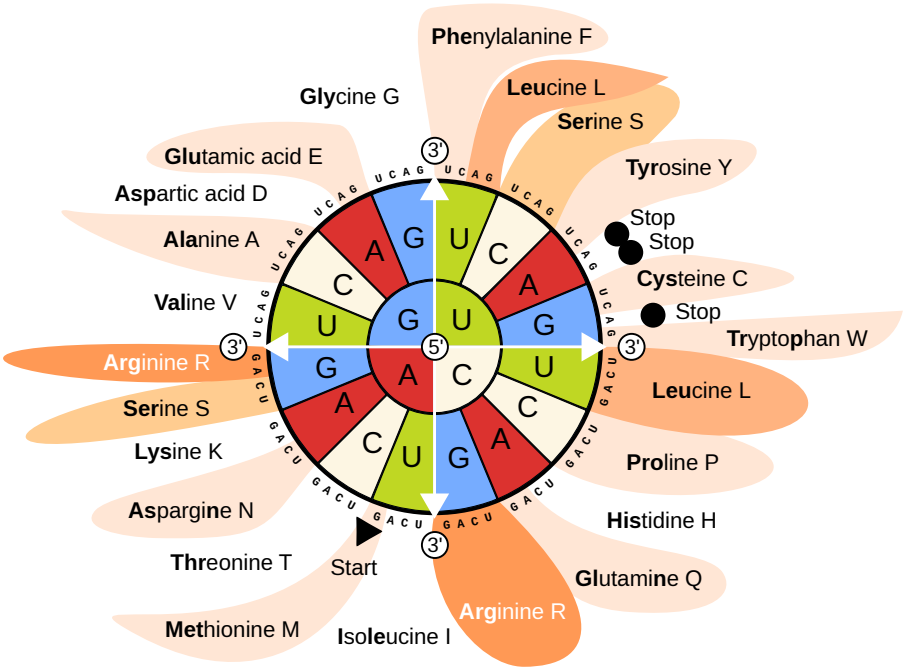

Figure 1.7: The meaning of each sequence made of three nucleotides (also known as a *codon* or *codon triplet*), as read by the ribosome. Each codon should be spelt starting from the inner circle and going outwards. The outer circle shows what corresponds to each codon.

Figure 1.7 shows the meaning of each codon. Since each nucleotide in a codon can have one out of 4 possible values (A, C, G or T), there exist $4^3 = 64$ possible codons. Out of these 64 combinations, three are recognised as "stop signals", while the rest encode amino acids, with the exception of the AUG codon which, in addition to represent methionine, is also typically used to initiate protein syntesis. Since there are 64 possible codon sequences, the code is degenerate, and most amino acids (excluding methionine and tryptophan) are encoded by more than a single codon. Most of the degeneracy occurs in the 3rd codon position.

The outer circle of Figure 1.7 contains the list of standard amino acids, together with their three- and one-letter abbreviations, which are shown in bold and after each AA name, respectively.

1.3.6 Information flow: the central dogma of biology

The single most important fact about biology is the so-called "central dogma of biology", which is a framework that describes the flow of genetic information in biological systems. It was formulated by Francis Crick[13] and presented to the public in a famous lecture given on September 19th in 1957 Cobb [2017]. The first formulation of the central dogma is shown in Figure 1.8, where arrows indicate the *flow of information*: what is used to build what. Note that DNA → protein and RNA → DNA information transfers were highly speculative at that time (and in fact DNA → protein transfers have not been observed *in vivo*).

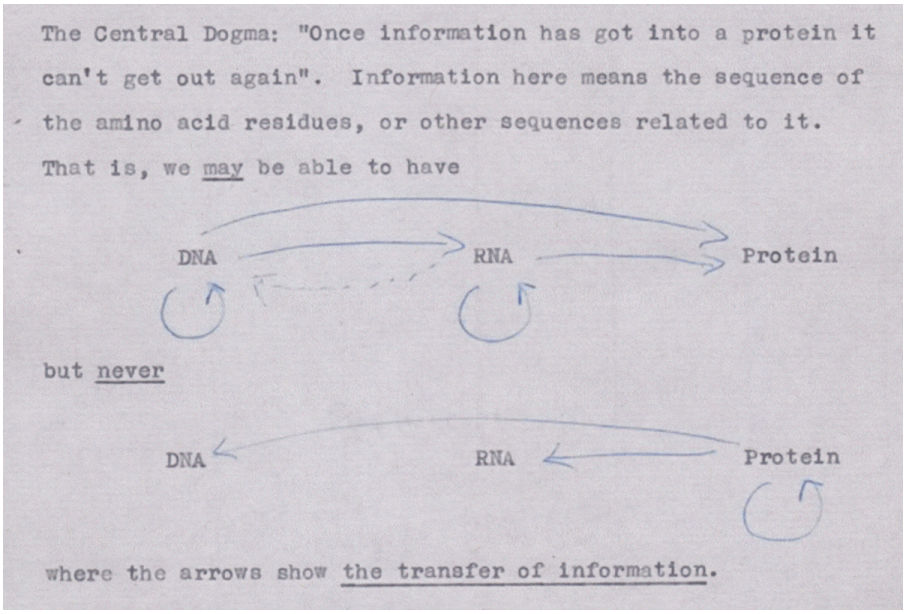

Figure 1.8: The first outline of the central dogma, from un unpublished note made by Crick in 1956. Credits: Wellcome Library, London. Taken from Cobb [2017].

In later formulations Crick expanded and clarified his view on this matter, writing that

Nota Bene: this version of the central dogma is not only the original one, but also the only one that stood the test of time: popularised versions such as "DNA → RNA → protein" or "DNA makes RNA, RNA makes protein" are oversimplified and ambiguous statements[14].

> **Attention**
> Try asking ChatGPT what is the central dogma of biology: Do you see anything wrong with the answer?

[13] Apparently Crick was not familiar with the meaning of "dogma", which is why he used that word instead of other options that would have perhaps avoided later misunderstandings (see *e.g.* Cobb [2017]).

[14] Oversimplified because they leave out important information transfers such as RNA → RNA and RNA → DNA, but also ambiguous since the meaning of arrows and "makes" are not spelt out.

> the central dogma could be stated in the form "once (sequential) information has passed into protein it cannot get out again".
> [...]
> the central dogma was a negative statement, saying that transfers from protein did not exist.
>
> CRICK [1970]

1.3.7 The experimental characterisation of the structure of biomolecules

I will refer to this part when talking about the secondary and tertiary structure of proteins and nucleic acids.

Secondary structure

Several experimental methods can be used to understand the secondary structure of a particular biomolecule, such as proteins or nucleic acids. Here are some common techniques:

1. **Circular Dichroism (CD) Spectroscopy:** CD spectroscopy is widely used to study the secondary structure of proteins. It measures the difference in the absorption of left-handed versus right-handed circularly polarized light, providing information about the overall content of α-helices, β-sheets, nucleic acid helices, and random coils.
2. **Fourier Transform Infrared (FTIR) Spectroscopy:** FTIR spectroscopy measures the absorption of infrared light by the biomolecule, which provides information about the types of chemical bonds and functional groups present. The amide I and II bands are particularly informative for analyzing protein secondary structures.
3. **Ultraviolet (UV) Absorption Spectroscopy:** UV absorption spectroscopy can provide some information about the secondary structure, especially in nucleic acids. It is less informative for proteins compared to CD spectroscopy but can still be useful when combined with other techniques.

TODO
Add figures from Finkelstein book (see pag 81).

Three-dimensional structure

The 3D structure of biomacromolecules such as proteins and nucleic acids can be experimentally determined using several techniques. The most commonly used methods are:

1. **X-ray Crystallography**:
   - **Procedure**: The biomolecule is crystallized, and then X-rays are directed at the crystal. The X-rays diffract as they pass through the crystal, creating a diffraction pattern that can be recorded. The diffraction pattern is analyzed to determine the electron density map, which is used to model the atomic structure of the biomolecule.
   - **Advantages**: High resolution, can be used for large biomolecules.
   - **Limitations**: Requires high-quality crystals, which can be difficult to obtain for some biomolecules (*e.g.* molecules that are not soluble in water, like many membrane proteins).
2. **Nuclear Magnetic Resonance (NMR) Spectroscopy**:
   - **Procedure**: The biomolecule is dissolved in solution, and NMR spectra are obtained by applying a magnetic field and measuring the resulting interactions between nuclear spins. The spectra provide information about the distances and angles between atoms, which are used to calculate the 3D structure.
   - **Advantages**: Provides information about the dynamics and flexibility of biomolecules in solution.
   - **Limitations**: Generally limited to smaller biomolecules (up to about 50 kDa, *i.e.* ≈ 500 amino acids or ≈ 200 nucleotides), although recent advances are pushing these limits.
3. **Cryo-Electron Microscopy (cryo-EM)**:

- **Procedure**: The biomolecule is rapidly frozen in a thin layer of ice, and then imaged using an electron microscope. Thousands of 2D images are collected and computationally combined to reconstruct a 3D model of the biomolecule.
- **Advantages**: Does not require crystallization, suitable for large and complex biomolecules, including those that are difficult to crystallize.
- **Limitations**: Typically lower resolution than X-ray crystallography, although recent advances have significantly improved the resolution.

4. **Small-Angle X-ray Scattering (SAXS)**:

- **Procedure**: The biomolecule is dissolved in solution, and X-rays are scattered at small angles by the sample. The scattering pattern provides low-resolution information about the overall shape and size of the biomolecule.
- **Advantages**: Can study biomolecules in near-physiological conditions, provides information about conformational changes and flexibility.
- **Limitations**: Lower resolution compared to X-ray crystallography and cryo-EM, providing only general shape information.

1.3.8 Some useful numbers and quantities

Quantity	Value
k_BT at 298 K	4.11×10^{-21} J
"	$4.11\,\mathrm{pn \cdot nm}$
RT at 298 K	$0.59\,\mathrm{kcal/mol}$
"	$2.48\,\mathrm{kJ/mol}$
Average nucleotide mass	$\approx 330 \to 300$ Da
Average amino acid mass	$\approx 110 \to 100$ Da

Abuse of notation
When dealing with free energies and Boltzmann factors, it is practical to use the quantity $\beta \equiv 1/k_BT$, which is expressed in units of measure of J^{-1}. However, in chemical physics and biophysics it is more common to use calories. As a result, I will frequently use β to mean $1/RT$.

1.4 Introduction to computational physics

Computational physics is a branch of physics that utilizes computational methods and numerical analysis to solve physical problems that are difficult or impossible to solve analytically. It involves the development and application of algorithms, numerical techniques, and computer software to simulate physical systems, analyze experimental data, and predict the behavior of natural phenomena.

At its core, computational physics merges principles from theoretical physics, applied mathematics, and computer science. It covers a wide range of topics, including classical mechanics, quantum mechanics, statistical mechanics, electrodynamics, condensed matter physics, biophysics, and more. In these days, high-performance computing is often required to perform large-scale simulations and calculations, which are essential for understanding physical processes where many length- and time-scales are involved, which happens often in various domains such as astrophysics, material science, biophysics, and climate modeling.

The methods employed in computational physics vary greatly, and include solving (partial) differential equations, performing molecular simulations, and employing finite element analysis. These techniques allow physicists to model systems at different scales, from subatomic particles to cosmological structures. Interestingly, many of the methods can be applied in multiple fields.

Another important field of application of computational physics is the interpretation of experimental data. By simulating experiments and comparing the results with actual measurements, computational physicists can refine theoretical models and improve the accuracy of predictions. This iterative process enhances our understanding of the underlying physical principles, and can help to uncover new phenomena.

Despite the enormous variety of methods and applications, there are some common concepts that are essential for anyone venturing in the subject, regardless of the specific subfield they choose. Some of these are technical, and are listed in the Section 1.2.1. Here, I will highlight two additional skill sets that every aspiring computational physicist should develop:

- **Bridging Theory and Experiment:** Historically known as "computer experiments", computer simulations sit in the middle between theory and experiment. As such, a good computational physicist is able to act as a bridge between these two worlds. A proficient computational physicist can effectively bridge these two realms, provided he develops a familiarity with the specific jargon, techniques, and methodologies of both fields. This dual expertise is a significant advantage. However, it is sometimes hard to find a fit in a dichotomic world, where "theory or experiment" is all there is. Fortunately, that world is dying (but has not died yet).
- **Algorithmic Problem Solving:** Problem-solving is a critical skill for any physicist, often involving the application of algorithmic reasoning to find solutions. For computational physicists, the ability to think in terms of algorithms is even more crucial. Developing this skill enables them to tackle complex problems systematically and effectively. During the course I will present many algorithms used to tackle some specific (bio)physical problem. Most of the time there will be an accompanying pseudo-code or real code, but in Section 1.2.2 you will be asked to do it yourself.

To get thing started, I will present a programming technique that will be applied later to some biophysical problems. However, before doing that, let's brush up our Python skills with this notebook.

1.4.1 Dynamic programming

> **Tip**
> This part was heavily inspired by István Miklós's Introduction to algorithms in bioinformatics.

At the most simple level, the three groups of biomolecules introduced above (DNA, RNA, and proteins) can be described in terms of a primary structure, which is just a sequence of characters. These characters are taken from a set called alphabet. DNA and RNA have alphabets of four characters (A, C, G, and T or U), while the protein alphabet is composed of 20 elements (the 20 amino acids). While the properties of any of these molecules are determined by their 3D structure, the latter depends, often in a very complicated manner, on their sequence. Therefore, it is most useful to develop tools and methods that can work efficiently with (possibly very long) 1D sequences.

For this reason, I will introduce dynamic programming, which is an algorithmic technique that shines at solving optimisation problems on sequences and trees. The main idea is to write the solution of a given complex problem in terms of simpler subproblems that can be solved. The solution to the original problem can then be "traced back" from the solutions of the simpler problems. A typical dynamic programming algorithm has two phases:

1. **The fill-in phase**, in which a so-called dynamic programming matrix (or table) is filled in. The dynamic programming matrix contains the scores of the subproblems, and once it has been filled, only the score of the solution is known, and not the solution itself.

2. **The trace-back phase**, in which the matrix is traversed back from the optimal score to the beginning to obtain the optimal solution (or solutions).

I will apply dynamical programming to the "coin change" toy problem. We will see later on how to apply it to more biologically relevant contexts.

The coin change problem

There are several variations to this problem. We start with the following: given an amount of money N and a set C of n types of coins, what is the minimum number of coins that can change N? Here is an example: for $N = 8$ and $C = \{1, 2, 5\}$, the minimum number of coins is 3, since $8 = 1 + 2 + 5$.

The simplest way of solving the problem is by enumerating all possible ways N can be changed with the coins in C, and then picking the one containing the fewest amount of coins. However, this algorithm performs worse and worse as N and C increases. Can we estimate by how much?

Algorithm complexity and Big O notation

Algorithm complexity is a crucial concept in computer science that describes how the runtime (or other important assets such as memory consumption) of an algorithm scales with the size of the input. It provides a high-level understanding of the algorithm's efficiency and performance. The standard way to express time complexity is through the so-called big O notation, which focuses on the worst-case scenario, giving an upper bound on the time an algorithm can take as the input size grows.

Big O notation abstracts away constants and less significant terms to highlight the primary factor affecting runtime. For instance, an algorithm with a time complexity of $\mathcal{O}(n)$ will have its execution time increase linearly with the input size n. In contrast, an algorithm with $\mathcal{O}(n^2)$ time complexity will have its execution time grow quadratically. Other common time complexities include $\mathcal{O}(1)$ for constant time, $\mathcal{O}(\log n)$ for logarithmic time, $\mathcal{O}(n \log n)$ for "linearithmic" time, and $\mathcal{O}(2^n)$ for exponential time.

Algorithmic complexity also applies to memory footprint, known as space complexity, which describes the amount of memory an algorithm requires relative to the input size. Often, there is a tradeoff between time and memory, where optimizing for faster execution might increase memory usage and vice versa. Understanding these tradeoffs helps in selecting the most appropriate algorithm for a given problem, ensuring efficient use of computational resources.

Using the notation introduced in the box, we can estimate the cost of the "brute-force" algorithm as follows. Given an amount N, a solution can either not contain c, or contain it up to N/c times. As a result, we have $(N/c)+1$ ways of using each coin c, and we have n such coins. Disregarding multiplying constants and lower-order terms, the overal complexity is then $\mathcal{O}(N^n)$. It is hard to overstate how bad is an exponential scaling. We have to do better!

Let's try with a "greedy" algorithm: we find a solution by taking the largest coin c that is smaller than N and applying the same operation to $N-c$, until the remaining amount becomes zero. Applying this algorithm to the example above would immediately yield the correct $S = \{1, 2, 5\}$ solution. The algorithmic complexity is also much better: the worst-case scenario is the one where we use coins of the same size, which would yield $\mathcal{O}(N)$. However, the algorithm requires that at each iteration we find the largest coin that is smaller than the residual amount. This can be done by either looping over C at each iteration, which would make the complexity $\mathcal{O}(Nn)$, or, which is much better, by sorting C beforehand, so that the total algorithmic complexity would be $\mathcal{O}(N) + \mathcal{O}(n \log n)$. This is **much** better than an exponential complexity. Unfortunately, greedy algorithms are known to be heavily attracted to local minima. For instance, if $N = 8$ and $C = \{1, 4, 5\}$, the greedy algorithm would yield a solution with 4 coins, since $8 = 5 + 1 + 1 + 1$. However it is clear that the best solution in this case is $8 = 4 + 4$.

We need an algorithm that is fast but does reliably find the correct solution. In order to do so, we need to find a way of casting the solution to the problem in terms of solutions of subproblems. To do so we define the accessory function $w(x)$:

$$w(x) = \begin{cases} m & \text{if } x \text{ is changeable,} \\ \infty & \text{if } x \text{ is not changeable,} \\ \infty & \text{if } x < 0, \end{cases} \tag{1.19}$$

where m is the minimum number of coins necessary to change x. We now make the crucial observation that

$$w(x) = \min_{c \in C}\{w(x-c)+1\}. \tag{1.20}$$

Proof. We prove Eq. (1.20) by mathematical induction. For the smallest value of x, such that $x \in C$ (*i.e.*, x is the value of a coin in the set C), the minimum number of coins required is 1, because x itself is a coin. Thus,

$$w(x) = 1 = \min_{c \in C}\{w(x-c)+1\}. \tag{1.21}$$

Assume that the equation holds for all values smaller than x. We need to show that it holds for x.

1. **Case 1:** If x is not changeable (*i.e.* x cannot be represented by any combination of coins from C), then for all $c \in C$, either $x - c$ is not changeable, or $x - c < 0$. In either case both sides of the equation are infinite, and the equality holds trivially.

2. **Case 2:** if x can be represented by some combination of coins, consider the optimal way to make change for x using the minimum number of coins. Suppose $c' \in C$ is one of the coins in the optimal change for x, and we consider the remaining amount to be changed $x - c'$. By the inductive hypothesis, the minimum number of coins required to change $x - c'$ is $w(x - c')$. Therefore, the total number of coins needed to change x is:

$$w(x) = w(x - c') + 1 \tag{1.22}$$

Since c' is chosen to minimize the total number of coins, we have in general

$$w(x) = \min_{c \in C}\{w(x-c)+1\} \tag{1.23}$$

Thus, by induction, the theorem is proved.

□

We can now leverage Eq. (1.20) to calculate the minimum number of coins necessary to change any amount x with a time that is linear in x and in the number of coin types, n, *i.e.* with an algorithmic (time) complexity $\mathcal{O}(xn)$. First, we apply Eq. (1.20) to progressively build a table containing the $w(y)$ values, with $y \leq x$, starting from $y = 0$:

```
DEFINE C as the set of possible coins
DEFINE N as the amount we want to change
DEFINE function w(x)
DEFINE table as an array with N + 1 entries

table[0] = 0
FOR each value y between 1 and N
   SET table[y] to the mininum value of {w(y - c) + 1}, where c is any coin in C
```

Figure 1.9: Pseudocode for the fill-in phase.

Once the table is ready, the answer to our question about the minimum number of coins can be read off its last entry, $w(x)$. However, if we want to know the details of the solution, *e.g.* which coins add up to x, we have to trace back Eq. (1.20):

```
ASSUME the definitions of the fill -in block
SET y = N
DEFINE S as an empty list

WHILE y is larger than 0
   FIND a coin c in C for which w(y) = (y - c) + 1
   ADD c to S
   SET y to y - c
```

Figure 1.10: Pseudocode for the trace-back phase.

Warning

In some cases, multiple ways exist to change an amount using the minimum number of coins. The algorithm described above will identify one such solution but can be extended to find all possible solutions. To achieve this, we must keep track of all branching paths that occur whenever more than one coin satisfies the condition $w(y) = (y - c) + 1$.

A simple example

Let $C = \{1, 4, 5\}$ and $N = 8$. By applying the fill-in algorithm detailed above we obtain the following table

from which we see that if $N = 8$ the optimal solution has two coins. To find out what are the coins in the optimal solutions, we apply the trace back algorithm, which in this case is made of two steps:

1. We start from $y = 8$, and the equation $w(y) = w(y - c) + 1 = 2$ has a single solution, $c = 4$. We set $y = 8 - c = 4$.
2. Now $y = 4$, and the equation $w(y) = w(y - c) + 1 = 1$ has, again a single solution, $c = 4$. We see that $y = 4 - c = 0$, which means that we are done.

Graphically, the steps above can be expressed as follows:

You can find a Python implementation of the brute-force and dynamic-programming algorithms here. Note that two brute-force version returns only the optimal number of coins and not the solution itself. Here is a list of things you can do with this code to brush up your coding and plotting skills:

1. **Easy**: Port the code to another programming language you know (*e.g.* C).
2. **Medium**: Use this code or yours to "experimentally" find out the algorithmic complexity of the two codes, in terms of N and n. In order to do so, solve the coin change problem for a bunch of N, n combinations[15], measure the time t it takes the code to find the optimal solution, and plot the two sets of results ($t(N)$ and $t(n)$).
3. **Hard**: Extend the brute-force solution so that it also returns an optimal solution.
4. **Hard**: Extend the dynamic-programming solution so that it returns all optimal solutions.

[15] in some cases it may be worth averaging over different C at fixed n.

Chapter 2

Proteins

Tip
The main references for this part are Finkelstein and Ptitsyn [2002], Lehninger et al. [2005], and Schlick [2010].

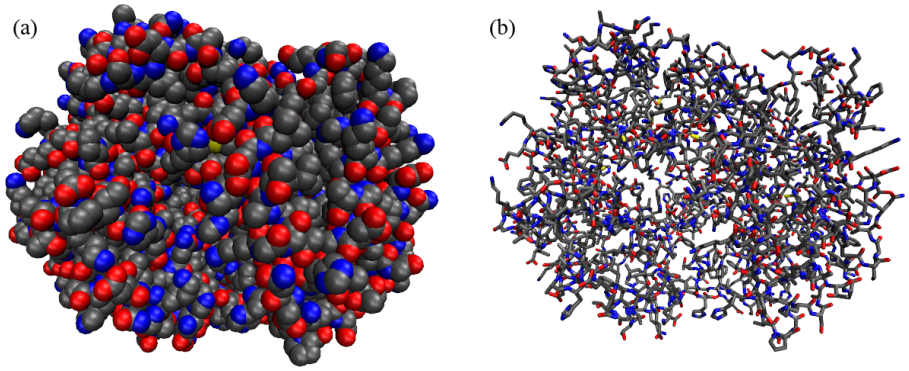

Figure 2.1: A protein (hemoglobin) represented in two different ways: (a) as a collection of atoms, shown as Van der Walls spheres, and (b) by putting emphasis on the covalent bonds that connect the atoms. The colour coding is: carbons are black, oxygen are reds, nitrogens are blue, phospori are yellow.

Proteins (an example of which is shwon in Figure 2.1, are macromolecules composed by amino acids linked by peptide bonds. Let's understand what these two things are.

2.1 Amino acids

An amino acid (AA) is a molecule that consists of

- an amino group: A functional group containing nitrogen, written as $-NH_2$. At neutral pH (pH ≈ 7) this group is protonated, *i.e.* it becomes $-NH_3^+$.
- a carboxyl group: A functional group consisting of a carbon atom double-bonded to an oxygen atom and bonded to a hydroxyl group, written as $-COOH$. At neutral pH (pH ≈ 7) this group is negatively charged by donating a proton, becoming $-COO^-$. Note that the carbon of this group is often named C' to distinguish it from the carbons of the side chains.
- a side chain (R group): A variable group that differs among amino acids and determines the characteristics and properties of each amino acid. The side chain can be as simple as a hydrogen atom (as in glycine) or more complex like a ring structure (as in tryptophan).
- a central carbon atom (C^α) that links together the three foregoing chemical groups plus an additional hydrogen atom.

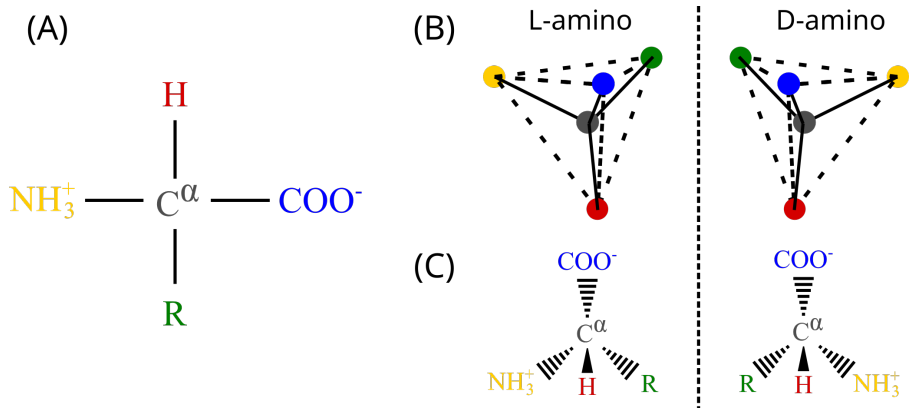

Figure 2.2: (A) The structure of a generic amino acid with side chain R. (B) A cartoon showing the spatial difference between the left-handed (L) and right-handed (D) amino-acid enantiomers. (C) Another representation showing that looking down the $H - C^\alpha$ towards the latter, the first letter(s) of the chemical groups read in clockwise order spell out a proper word ($CORN$) for L-amino acids, but a non-existing word ($CONR$) for D-amino acids. In (B) and (C) the chemical groups are coloured as in panel (A).

The four bonds of the C^α are arranged in a tetrahedral fashion, which means that all amino acids but glycine[1] are chiral molecules: each AA can, in principle, exist into two distinct forms that are one the mirror image of the other (*i.e.* they are enantiomers). Figure 2.2 shows the chemical structure of amino acids in panel (A), and the two enantiomers in panel (B) with two different representations. For reasons that are not yet understood, most proteins found in nature are made of L-amino acids[2]. By contrast, D-amino acids, which are synthesised by special enzymes, are rare but present in living beings, being involved in some specific biological processes (see *e.g.* Cava et al. [2010] and references therein).

2.1.1 List of amino acids

Given the generic nature of the side chain R, there exist countless different amino acid molecules. However, there are 20 standard amino acids that are commonly found in proteins and are encoded by the genetic code. These 20 amino acids are the building blocks of proteins in all known forms of life.

Figure 2.3 shows the chemical structure of the 20 standard amino acids, together with their three- and one-letter abbreviations. In the figure the AAs are grouped in the following classes:

1. Charged amino acids: AAs with side chains that are charged at physiological pH, making them highly hydrophilic. These are:
   - Lysine (Lys, K) and Arginine (Arg, R) - positively charged
   - Aspartic acid (Asp, D) and Glutamic acid (Glu, E) - negatively charged
   - Histidine (His, H) - being amphoteric, can be positively charged, negatively charged or neutral depending on the pH, playing a role in enzyme active sites.

2. Polar unchaged armino acids: AAs with side chains that can form hydrogen bonds, making them hydrophilic but not charged at physiological pH. These are:
   - Serine (Ser, S), Threonine (Thr, T), Asparagine (Asn, N), Glutamine (Gln, Q)
   - Cysteine (Cys, C) - has been traditionally considered a polar (hydrophilic) AA, but this view has been challenged, which is why here it is listed among the hydrophobic AAs. See for instance Nagano et al. [1999].

[1] To convince yourself that glycine is not chiral substitute R with H in the rightmost subpanel of Figure 2.2(C) and look down the $R - C^\alpha$ bond towards C^α: you will see that the resulting view is the same as that of the leftmost panel, which makes the two "enantiomers" identical.

[2] Incorporation of D-amino acids in proteins has been observed to occur only outside of ribosomes. Cava et al. [2010] reports some examples.

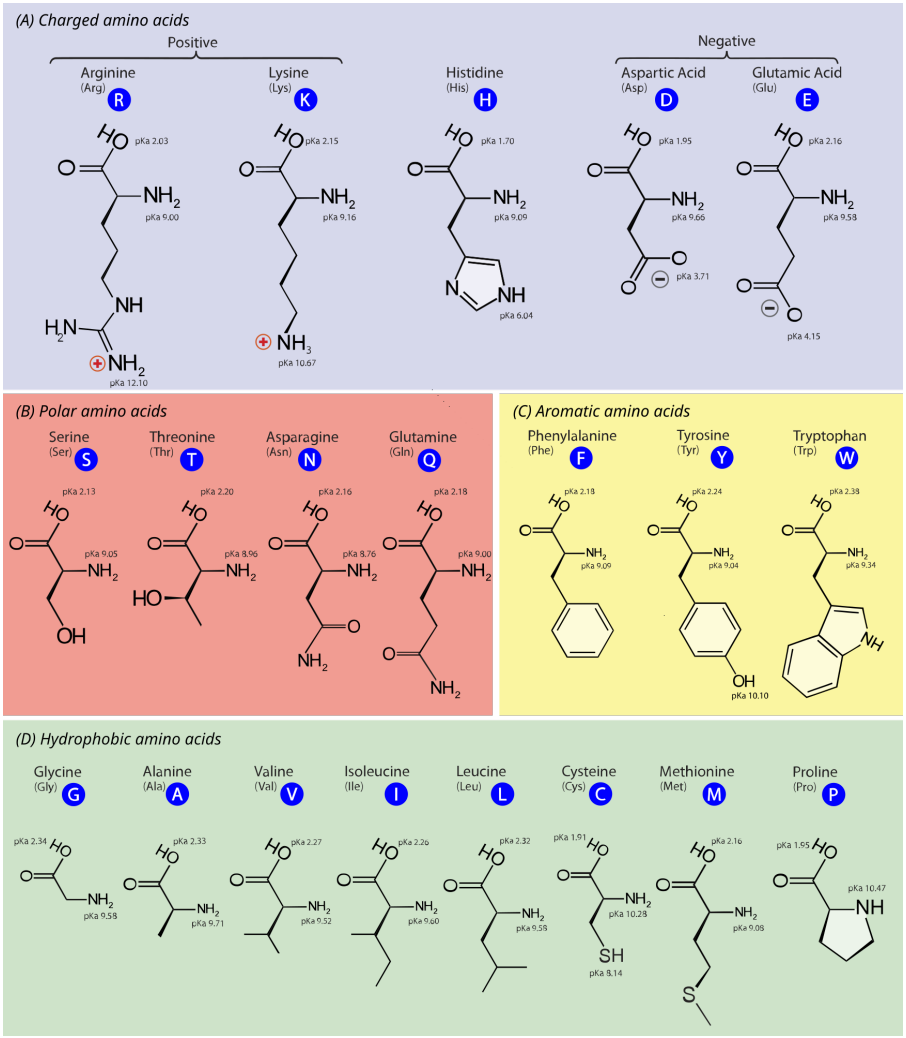

Figure 2.3: The chemical structure of the 20 standard amino acids. As discussed in the text, the AAs are separated into categories according to the properties of their side chains (R groups). Adapted from here.

3. Aromatic amino acids: AAs with side chains with aromatic rings, contributing to protein structure and function through stacking interactions. These are:
    - Phenylalanine (Phe, F), Tryptophan (Trp, W) - the side chains have a strong hydrophobic character
    - Tyrosine (Tyr, Y) - the presence of the hydroxil group $-OH$ endowes this AA with both hydrophobic and hydrophilic features

4. Hydrophobic amino acids: AAs with side chains that are hydrophobic and likely to be found in the interior of proteins. These are:
    - Glycine (Gly, G), Alanine (Ala, A), Valine (Val, V), Isoleucine (Ile, I), Leucine (Leu, L), Methionine (Met, M), Proline (Pro, P)
    - Cysteine (Cys, C) - has been traditionally considered a polar (hydrophilic) AA, but this view has been challenged, which is why here it is listed among the hydrophobic AAs. See for instance Nagano et al. [1999].

Non-standard amino acids

In addition to the 20 standard amino acids, there are a few non-standard amino acids that are found in some proteins or are used in specialized biological processes. Two notable ones are:

- Selenocysteine (Sec, U): Sometimes referred to as the 21st amino acid, it is incorporated into proteins by a unique mechanism that involves a specific tRNA and a specific sequence in the mRNA.
- Pyrrolysine (Pyl, O): Sometimes referred to as the 22nd amino acid, it is found in some archaeal and bacterial proteins and is also incorporated by a specific tRNA and sequence in the mRNA.

There are also many other amino acids that are not incorporated into proteins but have important roles in metabolism, such as ornithine and citrulline. Additionally, post-translational modifications can lead to the formation of amino acid derivatives within proteins, such as phosphorylated serine or hydroxyproline.

Note that in amino acids, the carbon atoms in the side chains are named systematically based on their position relative to the alpha carbon C^{α}. The naming convention is to use successive Greek letters (and possibly numerals for branched side chains) to denote each carbon atom, where the beta carbon, C^{β}, is attached to C^{α}, C^{γ} follows the beta carbon, and so on and so forth.

Here are some examples:

- In alanine the side chain has only one carbon, the beta carbon C^{β}.
- In leucine the side chain has four carbons and branches at the gamma carbon, so that the names are $C^{\beta}, C^{\gamma}, C^{\delta_1}$, and C^{δ_2}.
- In lysine the side chain has four carbons: $C^{\beta}, C^{\gamma}, C^{\delta}$, and C^{ϵ}.

2.1.2 Post-translational modifications

Post-translational modifications (PTMs) are chemical changes that occur to amino acids in proteins after they have been synthesized by ribosomes during the translation process. These modifications are essential for the proper functioning, regulation, and localization of proteins within the cell. PTMs can influence a protein's activity, stability, interactions with other molecules, and overall role in cellular processes. They are critical in the fine-tuning of protein functions and can affect the protein's behavior in various physiological contexts.

One of the most common types of PTMs is phosphorylation, which involves the addition of a phosphate group ($-PO_4$) to specific amino acids within the protein, typically serine, threonine, or tyrosine. A phosphate is charged and hydrophilic, so that its addition to side chains alters the interaction with nearby amino acids, making phosphorylation a key regulatory mechanism that can activate or deactivate enzymes, receptors, and other proteins, thereby modulating signaling pathways

and cellular responses. This modification is reversible and can be dynamically regulated by kinases (which add phosphate groups) and phosphatases (which remove them), allowing for precise control of protein function.

Another important PTM is glycosylation, where carbohydrate groups are attached to specific amino acid, often asparagine (N-linked glycosylation) or serine/threonine (O-linked glycosylation). Glycosylation plays a critical role in protein folding, stability, and cell-cell communication. It can also affect the protein's immunogenicity and its recognition by other molecules, influencing processes such as immune responses and cellular adhesion.

Additional types of PTMs include ubiquitination, where ubiquitin proteins are attached to lysine of a target protein, marking it for degradation by the proteasome[3]. Acetylation, the addition of an acetyl group ($-COCH_3$) to lysine, can impact protein stability and gene expression by modifying histones and other nuclear proteins. Methylation, the addition of methyl groups ($-CH_3$), typically occurs on lysine and arginine and can regulate gene expression and protein-protein interactions.

Finally, a very common PTM is the hydroxylation of proline, whereby a proline's pyrrolidine ring gains a hydroxyl ($-OH$) group. It is one of the main components of Section 2.8.1 which, as we will see in a few lessons, is a structural protein that provides strength, support, and elasticity to connective tissues throughout the body.

2.2 The peptide bond

Two amino acids can be linked together through a *peptide bond*, which is a covalent bond that connects the carboxyl group ($-COOH$) of one AA to the amino group ($-NH_2$) of the other. The reaction through which a peptide bond is formed is called "dehydration synthesis" since the carboxyl group loses a hydroxyl group ($-OH$), and the amino group loses a hydrogen atom ($-H$), which combine to form a water molecule (H_2O) that is released in solution. Once an AA has been incorporated into the growing chain and the water molecule has been removed, what remains of the molecule is called an *amino acid residue*, or just *residue*. Note that under physiological conditions this reaction is not spontaneous, and therefore requires catalysis (usually performed by ribosomes).

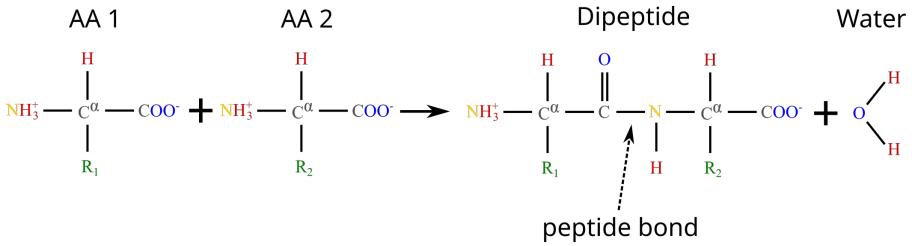

Figure 2.4: The formation of a dipeptide through a dehydration synthesis: two amino acids are joined together, and a water molecule is subsequently released. The peptide bond is indicated by the arrow.

Figure 2.4 shows the chemical reaction that takes place when two amino acids are joined together, highlighting the peptide bond that links them. The generic term for a chain of amino acids connected by peptide bonds is *polypeptide*, which is a class of biopolymers that comprises (but it is not strictly equivalent to) proteins. While there is some ambiguity in the definition, a polypeptide is considered to be a protein when it takes a specific three-dimensional structure[4] and a well-defined biological function. Note that by this definition a protein is not necessarily formed by a single polypeptide chain. Indeed, there are many proteins, like hemoglobin, that are made of multiple polypeptide units linked by non-covalent bonds (see Section 2.9 for a more comprehensive discussion on this matter).

[3]Proteasomes are large, multi-protein complexes responsible for degrading and recycling damaged, misfolded, or unneeded proteins within the cell. It recognizes proteins tagged with the small protein ubiquitin and breaks them down into small peptides.

[4]Contrary to this definition, not all proteins have a specific three dimensional structure, either because they are fully intrinsically-disordered, or because they contain regions that are intrinsically disordered.

Important
The peptide bond has a partial double-bond character arising from resonance: the lone pair of electrons on the nitrogen delocalizes to the carbonyl oxygen, creating a double-bond character between the carbon and nitrogen that restricts rotation around the peptide bond, making it planar and rigid.

The linear sequence of amino acids that are covalently linked together by peptide bonds to form a polypeptide chain is called *the primary sequence* of a protein. The primary sequence is the most basic level of protein structure and dictates the specific order in which amino acids are arranged, and it determines the protein's ultimate shape and function through interactions that lead to higher levels of structural organization which we will discuss later on. As a result, any changes or mutations in the primary sequence can significantly impact the protein's overall structure and function.

2.2.1 Trans and cis conformations

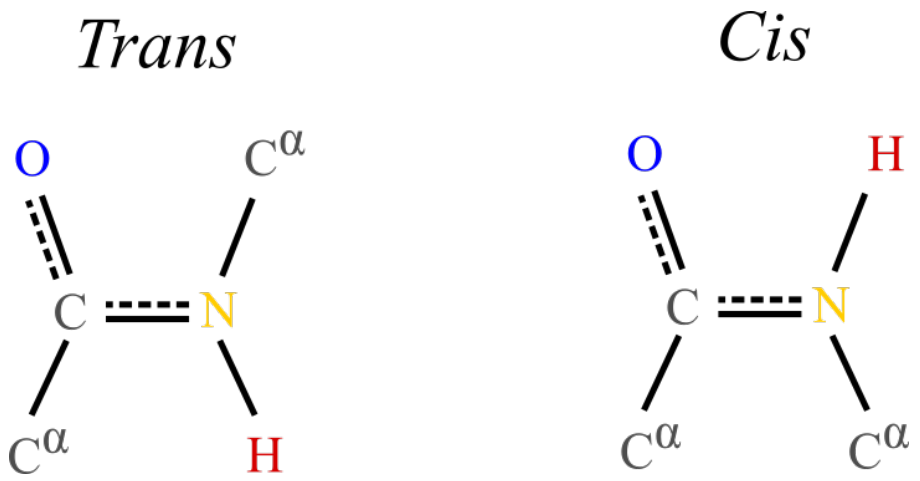

Figure 2.5: Typical trans and cis conformations of the backbone portion of a single amino acid. In the trans conformation the alpha carbons and their attached side chains are opposite to each other, making this configuration much more stable than the cis conformation, which is rarely observed in proteins.

The planarity of the peptide bond means that amino acids can take two distinct conformations around it:

1. *Trans* conformation: the alpha carbon atoms of the adjacent amino acids are positioned on opposite sides of the peptide bond. This arrangement is more stable and commonly observed in proteins because it minimizes steric hindrance between the carbons and side chains of the amino acids.

2. *Cis* conformation: the alpha carbon atoms of the adjacent amino acids are positioned on the same side of the peptide bond. This conformation is less stable due to increased steric clashes between the carbons and side chains, making it less common.

The trans conformation is energetically favorable and typically found in the vast majority of peptide bonds in proteins, while the cis conformation can be found in certain regions of proteins, especially involving the amino acid proline, whose unique cyclic structure partially stabilises the cis conformation.

2.3 Molecular vibrations

The length of chemical (covalent) bonds is of the order of an angstrom, with $C-H$, $O-H$ and $N-H$ being almost exactly 1 Å, $C-C$ being ≈ 1.5 Å, and $C=O$ and the peptide bond $C-N$ being in between (≈ 1.3 Å).

Regarding covalent bond angles, these depend on the nature of the hybridization. We are primarily concerned with sp^2- and sp^3-hybridized atoms, which result in planar (approximately 120°) and tetrahedral (approximately 109.5°) structures, respectively. In polypeptides, sp^2 hybridization is observed in the carbon and nitrogen atoms of the peptide bond, while sp^3 hybridization is seen in the alpha carbon, which forms four bonds, and in oxygen and sulfur atoms, which typically form two bonds.

As we will discuss later, vibrations of bond lengths and angles have characteristic frequencies associated to the infrared (IR):

- The bond lengths of interest have typical frequencies that go from $3\cdot10^{13}$ Hz for $C-C$ to $9\cdot10^{13}$ Hz for $C-H$ (corresponding to 1000 cm^{-1} and 3000 cm^{-1}, respectively)
- Bond angles have typical frequencies of $1.8\cdot10^{13}$ to $2.7\cdot10^{13}$ Hz for C^α, $1.5\cdot10^{13}$ to $2.4\cdot10^{13}$ Hz for C, and $2.1\cdot10^{13}$ to $3.6\cdot10^{13}$ Hz for N (500 to 1200 cm^{-1} for the total range).

The values of these typical frequencies can be compared to the thermal energy, k_BT, by using Planck's relation, $U=h\nu$, which yields

$$\nu_{\text{th}} = \frac{k_BT}{h} \approx 6.2\cdot10^{12}\,\text{Hz}. \tag{2.1}$$

Noting that this value is roughly half of the lowest vibrational or bending frequency reported above, we can conclude that thermal energy does not significantly impact bond stretching and angle bending vibrations[5]. As a result, covalent bond and angle vibrations contribute very little to the conformational flexibility of polypeptides, at least under normal conditions.

By contrast, the typical frequencies of rotations around single bonds are generally much lower than those of bond stretching or bond angle bending vibrations. In fact, some of these frequencies are in the range that is accessible by thermal energy at room temperature, endowing polypeptides with a rotational flexibility that is essential for their conformational dynamics, allowing them to adopt various functional states.

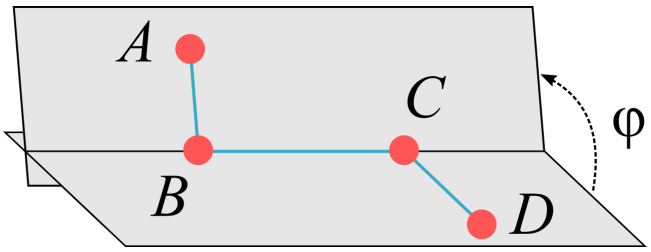

Figure 2.6: The dihedral angle φ is defined as the angle formed by the planes determined by the ABC and BCD atoms, which are connected by covalent bonds represented by blue lines. Note that in this picture only one of the two possible choices for the angle, *i.e.* the dihedral angle φ, is shown explicitly (see text for details).

The rotation angle around a bond is called torsional or *dihedral* angle, which, as shown in Figure 2.6, is the angle formed by two intersecting planes determined by the positions of four atoms. Given the four atoms in figure, $ABCD$, let $\overrightarrow{XY}$ be the vector connecting two covalently linked atoms X and Y. Given the ambiguity of defining the normal to a plane, the definition of the torsional angle requires a convention. If we choose to define the normal vector to the ABC plane as $\vec{n}_1 = \overrightarrow{AB}\times\overrightarrow{BC}$, the two normal vectors to the BCD plane are

[5]Temperature-induced fluctuations of bond angles are somewhat larger than those of bond lengths, and therefore cannot be disregarded completely. However the effect is rather minor, as typical amplitudes ar of the order of 5ř (Finkelstein and Ptitsyn [2002]).

$$\begin{aligned}\vec{n}_2 &= \overrightarrow{DC} \times \overrightarrow{CB} \\ \vec{n}_2' &= \overrightarrow{BC} \times \overrightarrow{CD} = -\vec{n}_2.\end{aligned} \tag{2.2}$$

We can now define two torsional angles, φ, $\hat{\varphi}$, as the angles between the normal vectors, *viz.*

$$\begin{aligned}\cos\varphi &= \frac{\vec{n}_1 \cdot \vec{n}_2}{\|\vec{n}_1\|\|\vec{n}_2\|} \\ \cos\hat{\varphi} &= \frac{\vec{n}_1 \cdot \vec{n}_2'}{\|\vec{n}_1\|\|\vec{n}_2'\|} = -\cos\varphi.\end{aligned} \tag{2.3}$$

The two angles are connected by the relation $\varphi + \hat{\varphi} = \pi$. Following Schlick [2010], I will call φ the dihedral angle, and $\hat{\varphi}$ the torsional angle, although the two terms are often used interchangeably.

As noted Section 2.2, the peptide bond, whose associated dihedral angle is called ω, has a partial double-bond character that restricts rotations around it. The peptide bond is considered to be "planar", *i.e.* that the dihedral angle takes values $\omega = \pi$ (trans) or $\omega = 0$ (cis), with the latter, as we said, being somewhat less common. Deviations from these values are considered to be rare, but this view has been challenged (see *e.g.* Berkholz et al. [2011]).

By contrast, rotations around bonds that connect sp^2- and sp^3-hybridized atoms are associated to energy barriers that are of the order of k_BT and therefore are the main contributors to the flexibility of the macromolecules. In the main chains of peptides the dihedral angles involved in these rotations are those associated to the $N - C^\alpha$ and $C^\alpha - C$ bonds, which are called ϕ and ψ.

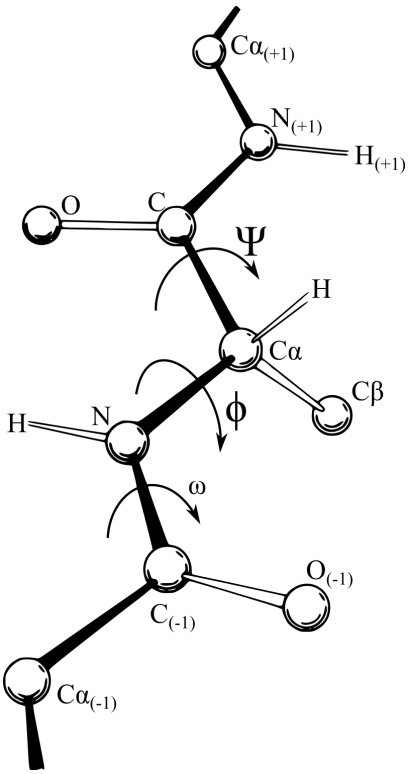

Figure 2.7: A polypeptide backbone showing the definition of the dihedral angles ϕ, ψ and ω. Credits to Dcrjsr and Adam Rdzikowski via Wikimedia Commons.

A graphical representation of a part of a polypeptide backbone with its associated dihedral angles is shown in Figure 2.7.

2.3.1 Ramachandran Plots

The small energy barriers associated to rotations around ϕ and ψ only take into account the atom themselves, not the rest of the molecule to which they are attached. Steric hindrances, resulting from

the repulsive interactions between atoms and groups attached to the C, C^α and N atoms, disfavor or even prohibit certain combinations of (ϕ, ψ) values.

A graphical representation of the allowed and disallowed dihedral angles of amino acid residues in protein structures is the Ramachandran plot, introduced for the first time in Ramachandran et al. [1963]. By plotting ϕ and ψ on the x and y axes, respectively, the plot reveals regions where the angles are sterically favorable, corresponding to common secondary structures and motifs like α-helices and β-sheets which will be introduced below. This visualization is crucial for validating protein structures, as it highlights conformational possibilities and identifies potential structural anomalies (*i.e.* AA conformations that lie in disallowed regions of the plot).

Warning
In many papers and books on the subject of protein structure, there is an important fact that is often not stressed or even omitted (perhaps because it should be obvious to people who, unlike us, know some biochemistry): in the original explanation of the Ramachandran plot, the steric clashes that contribute to excluding some (ϕ, ψ) combinations are those involving at least third neighbours (often called 1-4 interactions).

Given a protein structure, it is possible to extract and plot the ϕ and ψ values obtained for each residue on the same figure. However, different amino acids display different flexibilities depending on their associated side chains. Therefore, it is common to produce multiple Ramachandran plots, each serving specific purposes and providing insights into various aspects of protein structure and conformation. The most common ones are:

- **Glycine Ramachandran Plot:** glycine residues have more conformational freedom due to the absence of a side chain, and therefore a wider range of allowed ϕ and ψ angles compared to the general case, reflecting the famous enhanced flexibility of this AA.
- **Pre-Proline Ramachandran Plot:** residues that precede proline in the primary sequence often exhibit distinct conformational preferences given by the steric influence of proline.
- **Proline Ramachandran Plot:** proline residues have restricted ϕ and ψ angles due to the cyclic nature of its side chain, resulting in a limited range of conformations, highlighting the unique structural constraints of this AA.
- **Ile-Val Ramachandran Plot:** the branched carbons of isoleucine (Ile) and valine (Val) give them a distinct shape of disallowed ϕ-ψ regions.
- **General Ramachandran Plot:** ϕ and ψ angles for all residues that are not part of one of the foregoing categories.

Figure 2.14: Ramachandran plots of amino acids that compose the human hemoglobin protein 1A3N (points), plotted on background 85th and 75th centiles contour lines, white and blue lines, respectively, generated by analysing the Top8000 PDB data set. I have used a modified version of this software to make the plots.

Figure **??** shows examples of the plot types described above, with the empty (non-coloured) "space" being associated to angle combinations that are sterically forbidden. The points in the panels refer to the (ϕ, ψ) combinations of the residues of the human hemoglobin protein, while the background contour plots have been calculated by extracting the values of ϕ and ψ from 8000 reference protein structures listed in a database[6].

A comparison between Figure **??** and the other plots shows that glycine has a much greater conformational flexibility than any other AA. By contrast, Figure **??** and Figure **??** (and, to a smaller extent, Figure **??**) are much more constrained than an Figure **??**.

[6]While it is not explicitly stated in the repository of the software I have used to make the figure, I believe that the reference Top8000 database used is this one. The updated version of the same database can instead be downloaded here.

2.3.2 Non-covalent interactions

A closer look at the plots also reveals that $\phi \approx 0$ configurations are forbidden, while $\psi \approx 0$ are infrequent but not entirely disallowed. What is the difference between these two situations? And, more generally, what causes the shape of the Ramachandran plots?

The "forbidden" regions on this plot correspond to combinations of these angles that result in significant steric clashes and unfavourable interactions, making these conformations highly improbable or energetically unfavorable. These interactions are termed *noncovalent*, since they arise between pairs of atoms that are not involved in a bond. I will briefly sketch the most important noncovalent interactions in this context. Of course, as shown in Figure **??**, the absolute and relative importances of these terms depend not only on the nature of the amino acid, but also on its local environment.

Van der Waals interactions

The most prominent reason for the forbidden regions is steric hindrance, where atoms are positioned too closely together, leading to repulsive forces. This happens when the dihedral angles place the backbone or side chain atoms in close proximity, causing overlaps or severe crowding, making some conformations energetically unfavorable.

The atom-atom repulsion is a general phenomenon occurring when electron clouds belonging to atoms with fully-occupied orbitals overlap. The repulsion is caused by the Pauli exclusion principle, which prevents two electrons from occupying the same quantum state, and it is a steeply increasing function of the decreasing interaction distance. It is common to model the functional dependence of the repulsive interaction energy on r, the interatomic distance, with the (heuristic) form $\sim r^{-12}$, which is not based on first-principles calculations but is computationally cheap and good enough for most purposes (Leach [2001], Schlick [2010]).

When the atom-atom interdistance is sufficiently larger, quantum mechanical effects lead to attractive forces (the most important being the famous dispersion or London [1930] force) that arise from temporary dipoles induced in atoms or molecules as electrons move around. In general, the total attractive contribution is the sum of three contributions having the same functional dependence on distance, r^{-6} (see *e.g.* Israelachvili [2011]).

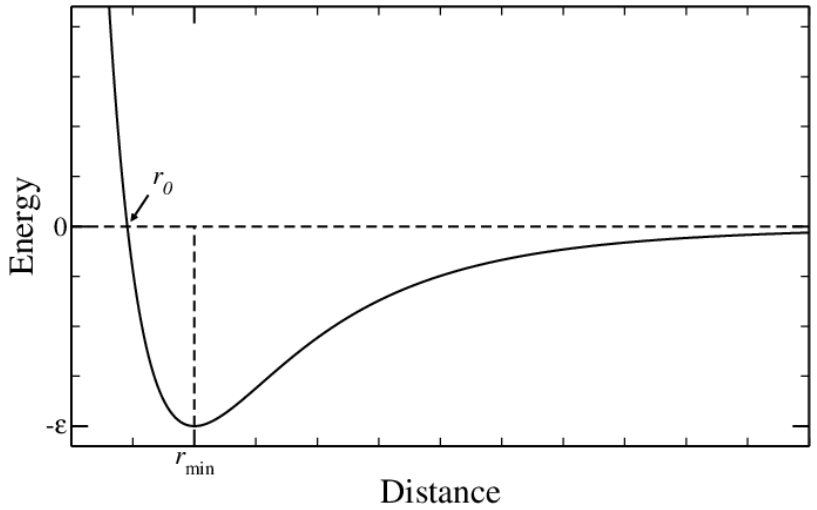

Figure 2.15: The Lennard-Jones potential, often used to model van der Waals interactions.

The total interaction, which is the sum of the two terms, is termed van der Walls interaction and is often modelled with the famous Lennard-Jones (LJ) potential, *viz.*

$$V(r) = \epsilon \left(\left(\frac{r_{\min}}{r} \right)^{12} - 2 \left(\frac{r_{\min}}{r} \right)^{6} \right), \tag{2.4}$$

where ϵ is the depth of the minimum arising from the competition between attraction and repulsion, and $r_{\min}$ is its position. A plot of the LJ potential is presented in Figure 2.15. The figure highlights

Table 2.1: Typical values for the parameters of the van der Walls interaction between same-type atoms (listed in the first column), as reported in Finkelstein and Ptitsyn [2002].

Atom	ϵ (kcal / mol)	$\beta\epsilon = \epsilon/k_BT$ at 298 K	$r_{\min}$ (Å)	r_0 (Å)
H	0.12	0.20	2.4	2.0
C	0.12	0.20	3.4	3.0
O	0.23	0.39	3.0	2.7
N	0.20	0.34	3.1	2.7

an additional length, r_0, which is the distance at which the energy becomes positive, and therefore can be seen as an estimate for the minimum distance below which two atoms "clash"[7]. Therefore, the value that r_0 takes for each pair of noncovalent interaction can be useful to estimate the stability of a given conformation.

An alternative form of the Lennard-Jones potential

In molecular simulations, it is common to express the LJ potential in the following slightly different manner:

$$V(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right), \tag{2.5}$$

where $\sigma = r_0$ is also known as the "diameter" of the atom.

Typical values for ϵ, r_0 and $r_{\min}$ for the main self interactions, *i.e.* interactions between atoms of the same type, are listed in Table 2.1. Note that the interaction strengths are always smaller than k_BT at ambient temperature. In the case of mixed interactions between atoms of type i and j, there exist several combining rules to estimate the values of the van der Waals parameters (see Leach [2001] for a discussion). The most common are the Lorentz-Berthelot rules, which amount to taking the arithmetic mean for the radii and the geometric mean for the energy strengths:

$$r_{\min}^{ij} = \frac{r_{\min}^{ii} + r_{\min}^{jj}}{2} \tag{2.6}$$

$$\epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}}. \tag{2.7}$$

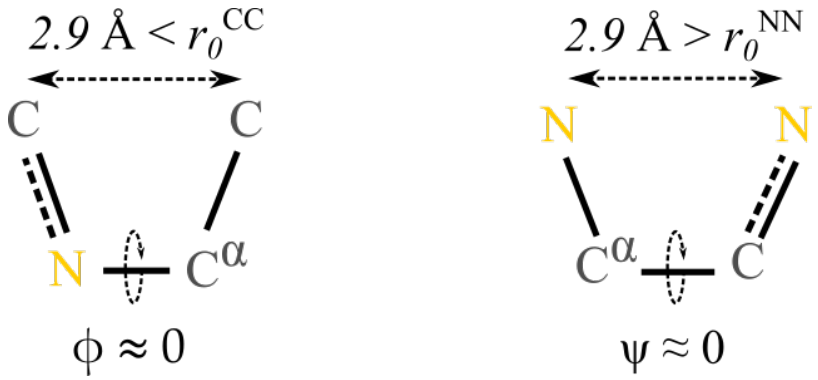

Figure 2.16: Cis conformation over ϕ (left) and over ψ (right). The numbers above the dashed lines are the average distances between C (left) and N (right) atoms.

[7] In writing this part I was heavily inspired by Finkelstein and Ptitsyn [2002], but I'm swapping the names of the two characteristic distances, r_0 and $r_{\min}$, which I found misleading in the original discussion.

We are now equipped to rationalise our observation that values of $\phi \approx 0$ are forbidden, while conformations with $\psi \approx 0$ are infrequent but not impossible. The difference between the typical conformations having these dihedral values can be appreciated by looking at Figure 2.16: in both cases the two 1-4 atoms connected to the atoms involved in the bond that defines the dihedral lie in the same half-plane and are roughly at the same distance ($\approx 2.9\,\text{Å}$). However, in the ϕ case the two atoms are both carbons, which have a "minimum allowed distance" of $r_0^{\text{CC}} \approx 3.0\,\text{Å}$, making this conformation essentially forbidden. By contrast, in the ψ case both atoms are nitrogens, so that the minimum allowed distance is $r_0^{\text{NN}} \approx 2.7\,\text{Å}$, which is smaller than the actual $N-N$ distance: according to the effect of this 1-4 interaction, this conformation is allowed.

Of course, even if we look at the contributions due to van der Waals forces only, there are other atom-atom interactions that forbid specific conformations. These interactions involve mostly the oxygen, the alpha carbon and the N hydrogen of the backbone, but also the side chains. However, in the latter case the main contributor is the beta carbon, which is why the only AA lacking it, glycine, has a distinctively different Ramachandran plot.

Hydrogen bonds

Certain combinations of ϕ and ψ angles can prevent the formation of hydrogen bonds, which are particular bonds that can link amino acids between them and with water molecules (or other solutes): if the backbone dihedral angles do not permit optimal hydrogen bonding, the conformation is unlikely to be stable or favorable. But what is a hydrogen bond?

A hydrogen bond (HB) is a type of weak chemical bond that occurs when a hydrogen atom, which is covalently bonded to an electronegative atom (like oxygen, nitrogen, or fluorine), experiences an attraction to another electronegative atom in a different molecule or a different part of the same molecule, and can be understood in terms of electrostatic interactions. Indeed, atoms like nitrogen, oxygen, and fluorine are highly electronegative, meaning they have a strong tendency to attract electrons. In a molecule, when hydrogen is covalently bonded to an electronegative atom, the shared electron is drawn closer to it, creating a partial negative charge on it and a partial positive charge on the hydrogen atom which generates an uneven distribution of electron density. If the hydrogen atom comes close enough to another electronegative atom having a partial negative charge, the electrostatic interaction between the opposite partial charges will give raise to an effective attraction. This effect can create reversible bonds between different molecules, or between different parts of the same molecule.

Important
Molecules or chemical groups capable of participating in hydrogen bonds are called *polar*.

Notable examples in this context are mainly those in which there are O and N atoms involved: $O-H :: O$, $N-H :: N$, $O-H :: N$, *etc.*, where :: is shorthand for a hydrogen bond. In all these cases, the atom covalently bound to the hydrogen (or sometimes the whole group) is called the *donor*, while the other one (or the molecule it is part of) is the *acceptor*. As a rule of thumb, the number of HBs that an electronegative atom can accept is equal to its number of lone pairs[8], which is one for nitrogen and two for oxygen.

We now look at the geometry of a hydrogen bond. In a HB, the distance between the acceptor and the hydrogen atom varies slightly depending on the specific molecules and environmental conditions, but typically ranges from 1.8 to 2.2 Å. As a result, the distance between the acceptor and the atom covalently bonded to the hydrogen is close to r_{min}, the optimal van der Waals distance for interactions involving O and/or N. A fundamental property of hydrogen bonds is that, unlike in van der Waals interactions, HBs are highly directional: their strength depends on the relative orientations of the chemical groups involved. Specifically, the optimal hydrogen bond occurs when the donor atom, hydrogen atom, and acceptor atom are aligned linearly, with an angle close to 180ř. Deviation from this linear alignment results in weaker hydrogen bonds, or even bond breakage if it exceeds $\approx 20-30°$.

[8]Pairs of valence electrons that are not shared with another atom in a covalent bond and are not involved in bonding.

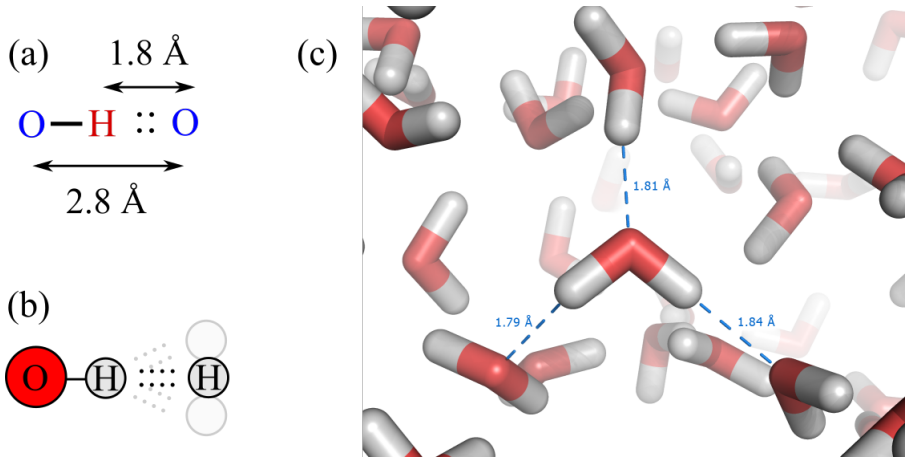

Figure 2.17: (a) Typical distances in a hydrogen bond. (b) Optimal hydrogen bonds are linear, but small deviations are possible. (c) Hydrogen bonds in a liquid water molecular dynamics simulation, with some acceptor-hydrogen distances highlighted (credits to Splette and Wikimedia).

Figure 2.17 shows a schematic view of the main properties of hydrogen bonds, together with a simulation snapshot of liquid water where some hydrogen bonds are explicitly highlighted.

The importance of HBs on the association of molecules can be appreciated by looking at the following table, which shows the melting and boiling temperatures at ambient pressure, T_m and T_b, of 3 molecules of similar weight:

Molecule	# of HBs	Molecular weight	T_m	T_b
Methane, CH_4	0	16 g/mol	$-182.6°$ C	$-161.6°$ C
Ammonia, NH_3	1	17 g/mol	$-77.7°$ C	$-33.3°$ C
Water, H_2O	2	18 g/mol	0° C	100° C

carbon is very weakly electronegative, it does not form any hydrogen bond. As a result, association in methane, as embodied by the values of T_m and T_b, happens at very low temperature, where the attraction is provided by Section 2.3.2 only. However, as electronegativity (and the number of lone pairs) increases from C to N and then O, HBs become more and more relevant, substantially increasing the melting and boiling temperatures.

Following Finkelstein and Ptitsyn [2002], the strength of hydrogen bonds can be estimated by comparing the (latent) heat of vaporization[9] of molecules that are composed by the same atoms, *e.g.* dimethyl ether $H_3C - O - CH_3$, 5 kcal / mol, and ethanol, $CH_3 - CH_2 - OH$, 10 kcal/mol. Since the latter has an OH group that the former lacks, there is an additional contribution of the latent heat due to one HB per molecule which amounts to $\approx 10 - 5 = 5$ kcal/mol. A similar estimate can be made by using the heat of vaporization of ice (we will see why in a moment), which is ≈ 12 kcal/mol. Of this value, ≈ 2 kcal/mol are due to van der Waals interactions, as estimated by looking at the evaporation of analogous, but non-HB-forming, molecules (*e.g.* CH_4 or O_2). The rest is provided by the two hydrogen bonds that each water molecule can form, yielding again an energy gain of ≈ 5 kcal/mol per HB. Recalling that, at room temperature, $k_BT \approx 0.6$ kcal / mol, we can estimate that one HB provides $\approx 8\, k_BT$, suggesting that the formation of hydrogen bonds is highly favoured under ambient conditions. The order of magnitude of this value is intermediate between the van der Waals interaction strength, which is a fraction of k_BT (see the table reported in the Section 2.3.2), and the strength of covalent bonds, which amounts to hundreds of kJ / mol ($> 100\, k_BT$).

In water, hydrogen bonds are the main driving force for condensation and ice formation. Since each oxygen can accept two bonds, and each hydrogen can mediate one donor-acceptor interaction, each water molecule can be involved in four bonds that are arranged on a (nearly perfect) tetrahedron. This geometry can be rationalised as follows. In general, electron pairs (both bonding and lone pairs) around a central atom will arrange themselves to minimize repulsion, but the repulsion between lone

[9]The amount of energy (usually expressed per mole) that must be supplied to a liquid to transform it into a gas.

pairs is greater than the repulsion between bonding pairs. Consequently, the two lone pairs will occupy positions that are as far apart as possible, pushing the bonding pairs closer together. As a result, the four pairs of electrons (two lone pairs and two bonding pairs) around the oxygen atom in water adopt a tetrahedral arrangement to minimize repulsion, with the bond angle between the hydrogen atoms in a water molecule being $\approx 104.5°$, slightly less than the ideal tetrahedral angle of 109.5ř due to the greater repulsion exerted by the lone pairs.

Since the natural tetrahedral geometry of water molecules leads to the maximum number of HBs, it should not come as a surprise that regular ice has a diamond cubic cell, where each molecule is connected to four others through (almost) linear hydrogen bonds. When ice melts into liquid water, it releases 80 cal/g of heat. In turn, when water turns into vapour at 100°, 600 cal/g of boiling heat is released. If we assume that no hydrogen bonds are present in water vapour, we could use these values to estimate that a fraction of $80/(600 + 80) \approx 0.12$ breaks when ice melts. However, this is not the case. In fact, it is perhaps surprising that liquid water at ambient conditions has roughly the same number of HBs of ice, *i.e.* ≈ 4. What happens then when ice melts into liquid water? The hydrogen bonds become looser, and their length and, more importantly, angular fluctuations increase. Of course, these fluctuations have an energetic cost, which is counterbalanced by the entropy provided by the many possible ways of realising disordered networks made of molecules connected by "flexible" hydrogen bonds.

Why does ice float on water?

The famous property of liquid water being denser than ice, which is one of the very many anomalies of water, can be qualitatively understood by considering the geometrical constraints set by hydrogen bonds.

In hexagonal ice, which is the most stable form of ice at ambient pressure, each water molecule forms four linear hydrogen bonds with neighboring water molecules, creating an almost perfect tetrahedral arrangement that leads to an open, hexagonal lattice structure with a lot of empty space within. By contrast, in liquid water each molecule still participates in nearly four hydrogen bonds. However, these bonds are not fixed and can fluctuate, allowing the molecules to pack closer together, which leads to a higher density, 1.00 g/cm^3 at 4° C, compared to that of ice, 0.917 g/cm^3.

We now go back to proteins, with the working hypothesis that, under ambient conditions, all hydrogen bonds are formed[10]. If we look at the peptide backbone, we see that it contains nitrogen and oxygen, which are polar groups and can be involved in hydrogen bonds. In addition, as Section 2.1.1, many side chains are polar, or even charged under physiological conditions. As a result, a polypeptide has many chemical groups that can be involved in hydrogen bonds. What partners do these chemical groups bond to: other polar groups or water molecules?

To answer this question let us consider two chemical groups on the same polypeptide, an acceptor A and a donor D, that can form a hydrogen bond. Assuming that all hydrogen bonds are formed, there are two possible states: A is bonded to D, or they are both bonded to water molecules. The equilibrium between these two states can be described by

$$D :: HOH + A :: OH_2 \rightleftharpoons D :: A + HOH :: OH_2. \tag{2.8}$$

When D and A form a HB, the two water molecules are freed to form hydrogen bonds with other water molecules. Since the number of hydrogen bonds on the left-hand side matches that on the right-hand side, the energy remains essentially unchanged. However, entropic contributions must also be considered:

1. When a hydrogen bond forms between A and D, the polypeptide's flexible backbone is typically forced to adopt a more constrained and specific conformation to facilitate this interaction, leading to a loss of conformational freedom and therefore a loss of entropy.

[10]This is not a particularly strong approximation.

2. When the intramolecular hydrogen bond forms, the water molecules previously bonded to the polar groups are released into the bulk solvent. These water molecules, previously constrained in position and orientation (low-entropy state), now return to a more disordered state, increasing the system's entropy

Interestingly, it turns out that, on average, the entropic contributions of these two effects not only have opposite signs but are also comparable in magnitude (see Finkelstein and Ptitsyn [2002] for a more thorough discussion), so that the conformation the backbone adopts is "marginally" stable and depends on the local environment rather than on intrinsic factors.

Electrostatic interactions

According to classical electrostatic, the interaction energy between two charges q_1 and q_2 at distance r in a homogeneous medium is

$$U(r) = \frac{q_1 q_2}{4\pi\epsilon r}, \tag{2.9}$$

where $\epsilon = \epsilon_r \epsilon_0$, ϵ_r is the relative dielectric permittivity of the medium and ϵ_0 is the absolute permittivity of vacuum. Common values of ϵ_r are ≈ 80 for water, ≈ 3 for a folded protein and 1 for vacuum (or air). However, we are not dealing with homogenous media, as the distances we consider are very much comparable with inter-molecular separations. In all-atom simulations, solvent and solute atoms and molecules are handled explicitly and therefore there is no "medium" and Eq. (2.9) is applied with $\epsilon_r = 1$. By contrast, when one is interested in coarse-grained models, where the solvent's effect is described implicitly, or in theoretical considerations, when the description should be as simple as possible, one has to address the question of how to describe the effect of macromolecular charged groups and their interactions.

Let's start by considering what happens inside a protein. If we consider two positive charges separated by a distance of 3 Å, their interaction energy is ≈ 1.5 kcal/mol ($\approx 2.5 k_B T$) in water, while a whopping ≈ 40 kcal/mol ($\approx 70 k_B T$) in a non-polarisable medium such as plastics or dry proteins ($\epsilon_r \approx 3$). Such a large repulsion would be enough to destroy a protein, which is usually stabilised by a "reserve" free energy of ≈ 10 kcal/mol (Finkelstein and Ptitsyn [2002]). This tells us that charged side chains will not be found close to each other in a protein core. However, we can go further and consider the electrostatic self-energy of a single atom of charge q and radius R[11]:

$$U_{\text{self}} = \frac{q^2}{8\pi\epsilon R}. \tag{2.10}$$

Setting $q = e$ and $R = 1.5$ Å yield the same values we obtained for interacting pairs (*i.e.* ≈ 1.5 kcal/mol in water, ≈ 40 kcal/mol in the protein), meaning that the cost of having a single charged group inside a protein core exceeds the stability of the protein itself. As a consequence, ionisable side chains are nearly always uncharged when embedded in a core, since the free-energy cost of donating or accepting a H^+ (to or from water) is much smaller than the energetic contribution due to the electrostatic self-energy of a charge.

Now we can consider what happens outside of the protein. Following Finkelstein and Ptitsyn [2002] we can leverage basic[12] electrostatics in the presence of dielectric materials to find that even close to the protein surface the dielectric constant is much larger than that inside the core ($\approx 40 \gg 3$). Thus, the charge-charge interaction strength is always severely reduced compared to the vacuum.

Everything said above rests on the assumption that the medium can be regarded as continous and homogeneous. However, when dealing with atomic distances ($\approx$ Å), the granularity of the solvent cannot, in principle, be neglected. For instance, if two charges are $3-4$ Å apart, no atoms or molecules can get in between them and affect the resulting electrostatic interaction. However, it turns out that the water molecules that are attracted by the pair of charges and come from the sides are

[11] This quantity is just the work required to charge up a spherical body of radius R, $\int_0^q \frac{q' dq'}{4\pi\epsilon R}$.

[12] basic but by no means obvious or straightforward.

enough to screen the electrostatic interactions almost as if the solvent were homogeneous and had the "macroscopic" dielectric permittivity ϵ_r.

This should not come as a surprise if we think about kitchen salt, $NaCl$: the binding energy between the Na^+ and Cl^- ions, if we again take $R = 3\,\text{Å}$, is -1.5 kcal/mol and -120 kcal/mol in bulk water ($\epsilon_r = 80$) and vacuum ($\epsilon_r = 1$), respectively. Even if we assume that the breakdown of the assumption of the continouous and homogeneous nature of the solvent brings the relative dielectric constant down to "only" $\epsilon_r = 20$, we would still have an interaction energy of -6.0 kcal/mol, which is about the same strength of a single water-water Section 2.3.2. In this case we would expect the concentration of a saturated salt solution to be close to that of saturated water vapour, which is $\approx 10^{-4}$ mol / l (or 10^{-4} M). However, this is many times smaller than the $NaCl$ concentration in salty sea water (~ 1 M), which confirms that ϵ_r should be much larger than 20, and therefore rather close to the bulk value of 80, even when $R \approx 3\,\text{Å}$.

2.3.3 Rotamers

Almost all amino acids can adopt different orientations of side chains around single bonds. Exceptions are

- glycine, which has no rotational freedom in terms of side chain orientation,
- alanine, whose side chain is a methyl group, $-CH3$, which is small and symmetrical, offering no significant variation in rotameric states,
- proline, whose side chain forms a ring by bonding back to the backbone nitrogen, severely restricting the rotation around its side chain.

The distinct conformations of an amino acid at fixed values of (ϕ, ψ) are called *rotamers* and arise due to the rotation around the bonds connecting the side chains to the main backbone of the polypeptide chain. The dihedral angles connected to the rotamers are referred to as χ angles, which describe the rotations around the bonds within the side chains of amino acids. Each rotatable bond in a side chain is associated with a specific χ angle, labeled sequentially from the backbone outward. For instance, χ_1 is the dihedral angle around the bond between the alpha carbon and the beta carbon, χ_2 is the dihedral angle around the bond between the beta carbon and the gamma carbon, and so on for longer side chains.

Each amino acid side chain can adopt multiple rotameric states, influenced by steric interactions, hydrogen bonding, and other intramolecular forces. The different rotamers contribute to the overall flexibility and diversity of protein structures, allowing proteins to achieve their functional conformations and interact effectively with other molecules.

2.4 Primary structure

The order in which amino acids are linked together by peptide bonds is known as the *primary structure* of a protein. The convention to list the sequence of a protein is to use the one-letter or three-letter codes for amino acids in the sequence, which has a well-defined directionality. The sequence starts at the end of the amino acid chain that has a free amine group ($-NH_2$), which is called N-terminus and, by convention, marks the beginning of a protein. The other end has a free carboxyl group ($-COOH$) and is known as the C-terminus, which is considered the end of the protein.

An example

The sequence of bovine (cow) rhodopsin expressed with the one-letter code is

```
MNGTEGPNFYVPFSNKTGVVRSPFEAPQYYLAEPWQFSMLAAYMFLLIMLGFPINFLTLYVTVQHKK
```

2.5 Visualising molecules

When dealing with molecular models, it is very important to be able to visualise the 3D structure of the system. Indeed, visualization tools make it possible to explore the three-dimensional conformation of molecules, helping to identify key structural features (*e.g.* active sites, binding pockets, regions of flexibility, *etc.*). When performing simulations, the are many uses for visualisations. Here I will list the main ones:

1. It is very important to familiarise with the system under investigation: always look at what you simulate!

2. A big part of setting up complicated simulations is to prepare the initial configuration. This is often a multi-step process, and being able to visually follow this process can speed-up the troubleshooting of possible issues.

3. Sometimes (often...) simulations provide "wrong" results, either because something went awry during the simulation itself, or because some parameters were incorrectly set. You should always make sure that the final (or equilibrium) configurations of your simulations make sense and match the results you obtain, and visualising them help in this direction.

4. A big part of doing science is to be able to efficiently and clearly explain your results to an audience. Good figures are fundamental to convey the important messages of your work, and sometimes a visualisation of a 3D structure (or part of a 3D structure) can very efficiently make a point across.

The next two sections will briefly introduce how information about a molecular system can be stored on a computer, and how to visualise a molecular structure.

2.5.1 File formats

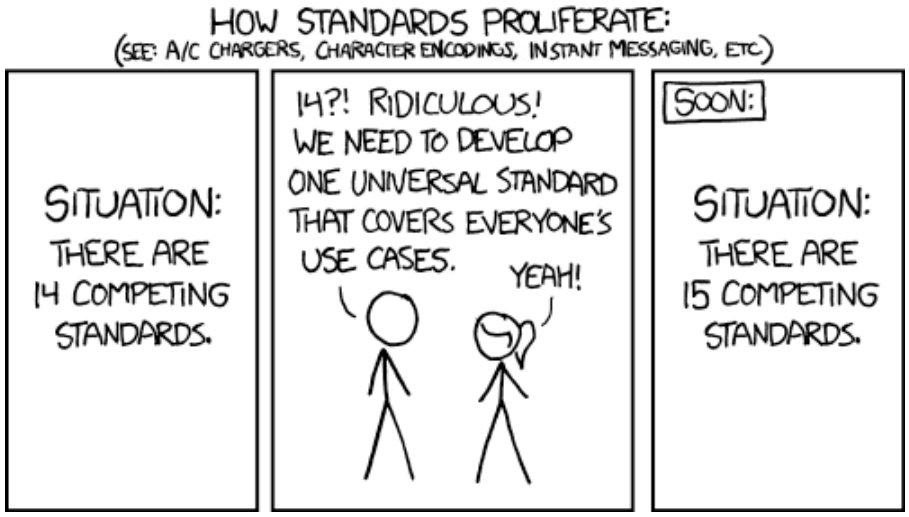

Figure 2.18: The proliferation of standards. Credits to xkcd.

For historical (but not only, as shown in Figure 2.18) reasons, there exist multiple file formats for representing molecular structures, simulation data, and related information. While each main simulation software has its own, there are also some (usually software-agnostic) standard formats that are widely used to share information about molecular structures. In structural and computational biology, the standard is the Protein Data Bank (PDB) file format, originally developed in the 1970s to archive experimental data from X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy. Each PDB file contains a detailed description of the atomic coordinates, connectivity, and

sometimes additional information like secondary structure annotations, heteroatoms, and crystallographic data.

The full format specification can be found at this link. Here I will provide a short description based on this one, which should be enough for most use cases. A PDB file is a human-readable text file where each line of information in the file is called a record, and the records are arranged in a specific order to describe a structure. The first part of a PDB file usually contains some metadata, with the most common record types being:

- **HEADER**: a general description of the contents of the file, including the name of the macromolecule, classification, deposition date, and PDB ID.
- **TITLE**: a more detailed description of the structure, often including the protein's function, any ligands or cofactors, and relevant experimental conditions.
- **COMPND**: the macromolecular components of the structure (*e.g.* the names of individual protein chains, nucleic acids, ligands, *etc.*).
- **SOURCE**: the source of each macromolecule (*e.g.* the organism, tissue, or cell line from which the structure was derived).
- **KEYWDS**: keywords that describe the structure, such as enzyme classification, protein family, or biological process.
- **EXPDTA**: nformation about the experimental method used to determine the structure (*e.g.* X-ray crystallography, NMR spectroscopy, cryo-electron microscopy, *etc.*).
- **AUTHOR**: the names of the researchers who contributed to the structure determination.
- **REVDAT**: the revision history of the PDB file, including the dates of modifications and a brief description of the changes made.
- **JRNL**: reference(s) of the publication(s) associated with the structure determination.
- **REMARK**: more general remarks or comments related to the structure (*e.g.* experimental details, data processing, *etc.*).
- **SEQRES**: the primary sequences of the macromolecules stored in the file, with the amino acid or nucleotide residues given by their one-letter codes.

The actual structure and connectivity is stored in records that are usually of the following types:

- **ATOM**: atomic coordinate record containing the x, y, z coordinates, in Å, for atoms in standard residues (amino acids and nucleic acids).
- **HETATM**: same as ATOM, but for atoms in nonstandard residues. Nonstandard residues include ions, solvent (such as water molecules), and other molecules such as co-factors. The only functional difference from ATOM records is that HETATM residues are by default not connected to other residues.
- **TER**: indicates the end of a chain of residues.
- **HELIX**: indicates the location and type (right-handed α, *etc.*) of helices, which are secondary structures that will be introduced soon. One record per helix.
- **SHEET**: indicates the location, sense (anti-parallel, *etc.*) and registration with respect to the previous strand in the sheet (if any) of each β-strand, which is another type of secondary structure (see below). One record per strand.
- **SSBOND**: defines disulfide bonds, which are particular bonds linking cysteine residues that will be introduced later.

Each record has a specific format that was designed when punched cards where still common. Therefore, records are made of fields that have a fixed width, and never go beyond 80 columns. Here I report the format of the ATOM, HETATM, and TER records:

Record Type	Columns	Data	Justification	Data Type
ATOM	1-4	"ATOM"		character
	7-11	Atom serial number	right	integer
	13-16	Atom name	left	character
	17	Alternate location indicator		character
	18-20	Residue name	right	character
	22	Chain identifier		character
	23-26	Residue sequence number	right	integer
	27	Code for insertions of residues		character
	31-38	x coordinate	right	real (8.3)
	39-46	y coordinate	right	real (8.3)
	47-54	z coordinate	right	real (8.3)
	55-60	Occupancy	right	real (6.2)
	61-66	Temperature factor	right	real (6.2)
	73-76	Segment identifier	left	character
	77-78	Element symbol	right	character
	79-80	Charge		character
HETATM	1-6	"HETATM"		character
	7-80	same as ATOM records		
TER	1-3	"TER"		character
	7-11	Serial number	right	integer
	18-20	Residue name	right	character
	22	Chain identifier		character
	23-26	Residue sequence number	right	integer
	27	Code for insertions of residues		character

following excerpt, taken from the PDB file of human hemoglobin, shows how a PDB file looks like:

```
HEADER    OXYGEN TRANSPORT                        22 -JAN -98   1A3N
TITLE     DEOXY HUMAN HEMOGLOBIN
...
AUTHOR    J.TAME,B.VALLONE
...
REMARK   2
REMARK   2 RESOLUTION.    1.80 ANGSTROMS.
REMARK   3
REMARK   3 REFINEMENT.
REMARK   3   PROGRAM     : REFMAC
REMARK   3   AUTHORS     : MURSHUDOV,SKUBAK,LEBEDEV,PANNU,STEINER,
REMARK   3               : NICHOLLS,WINN,LONG,VAGIN
...
HELIX    1   1 PRO A    4  SER A   35  1                                  32
HELIX    2   2 PRO A   37  TYR A   42  5                                   6
HELIX    3   3 ALA A   53  ALA A   71  1                                  19
HELIX    4   4 MET A   76  ALA A   79  1                                   4
HELIX    5   5 SER A   81  HIS A   89  1                                   9
```

```
...
ATOM    219  CB  ARG A  31      -2.135   3.057  22.133  1.00  6.25           C
ATOM    220  CG  ARG A  31      -3.684   3.130  22.122  1.00  7.07           C
ATOM    221  CD  ARG A  31      -4.155   4.091  21.001  1.00  7.50           C
ATOM    222  NE  ARG A  31      -5.608   4.102  20.966  1.00  8.30           N
...
TER    4374      HIS D 146
HETATM 4375  CHA HEM A 142       8.456  12.968  30.943  1.00 12.71           C
HETATM 4376  CHB HEM A 142       6.923  14.956  26.818  1.00 15.63           C
HETATM 4377  CHC HEM A 142       8.220  10.950  24.353  1.00  9.88           C
HETATM 4378  CHD HEM A 142       9.359   8.784  28.599  1.00  7.91           C
...
END
```

The PDB is a legacy format
While PDB files are still very common, they are being superseded by more flexible[13] file formats. For instance, the PDB format has been declared obsolete by the Protein Data Bank database, whose default has become the Adams et al. [2019]. Expect the PDB format to slowly disappear!

Python implementation
Head over here for a Python notebook that shows how to parse and analyse PDB files.

2.5.2 Software tools

There are many software tools that make it possible to visualise and interact with molecular models. Here is a (very non-comprehensive) list:

- VMD is open source. It supports many formats, handles large systems efficiently, and has numerous plugins available and powerful scripting capabilities. However, its interface is rather ugly (especially compared to the other visualisation tools), and many functionalities can be complex for beginners.
- PyMol is open source. It can be used to (more or less) easily producing high-quality, publication-ready images and animations, it has an extensive user community and good documentation and powerful scripting capabilities with Python. However, it can be get slow with very large molecular systems, and it can be complex for beginners to master all features and functionalities.
- Chimera[14] is not open source, although its source code is available. It is the most intuitive and easy to use software presented here, especially for beginners, and it has excellent visual quality for creating publication-quality images, supports a wide range of file formats and integrates well with other bioinformatics tools. It has somewhat less powerful scripting capabilities compared to VMD and PyMOL, and ChimeraX has some features behind a paywall.

I urge you to install and try at least one of these programs. Use this file, which I used to generate Figure 2.19, or this real protein file to test the software. In Chimera and PyMol you can directly open the file, while with VMD you'll have to create a new molecule (File → New molecule... → Browse → Load) and choose a sensible representation (Graphics → Representations... and then pick *e.g.* "NewCartoon" from the "Drawing method" menu).

2.6 Secondary structure

The secondary structure of proteins refers to the local, repetitive folding patterns of the polypeptide chain. These structures are stabilized primarily by hydrogen bonds between the backbone atoms in

[14]There is also a new software, called ChimeraX, that is advertised for having "higher performance and many new features".

the polypeptide chain, specifically between the carbonyl oxygen ($C = O$) of one amino acid and the amide hydrogen ($N - H$) of another.

The most common secondary structures found in proteins are helices (and α-helices in particular) and β-sheets.

2.6.1 Helices

A helix of length n_0 is a repetitive structure whereby the $C = O$ group of the i-th AA in thee chain is hydrogen-bonded to the $H - N$ group of a $(i + k)$-th residue, for n_0 consecutive values of i. In protein systems, a helix is identified by a name k_N^H, where:

- k is the number of residues per helical turn;
- N is the number of atoms in the ring closed by a hydrogen bond. For instance, the rings formed in a regular $k = 2$ helix are 7: $O :: H - N - C - C^\alpha - N - C$ (where the latter C belongs to the i-th AA).
- H is the handedness of the helix: if we look down a helix (*i.e.* if AAs with higher index come towards us) and the $(i + 1)$-th AA is staggered in the counterclockwise direction with respect to the i-th the the helix is *right-handed* (R), otherwise is left-handed (L).

The helices that appear in proteins are 3_{10}^R, 3.6_{13}^R (also known as α-helix) and 4.4_{16}^R (also known as π helix). Note that they are all right-handed[15]. The π-helix is uncommon since its open structure disfavours some stabilising van der Waals interactions, and it is rarely (if ever) present in canonical (*i.e.* regular) form[16]. Therefore, we will not consider it any further.

Given the periodicity of a helix, there are optimal values of ϕ and ψ that maximize efficient atomic packing and precisely align the peptide bonds, thereby stabilizing the helical structure. These values, together with other properties of the two most common helices, are reported in the table below:

Helix	HB pattern	Residues per turn	Pitch (Å)	ϕ	ψ
3_{10}^R	$i \to i + 3$	3	6.0	$-50°$	$-25°$
α_R-helix (3.6_{13}^R)	$i \to i + 4$	3.6	5.4	$-60°$	$-45°$

both these helices are usually right handed, I will drop the R sub- and superscript.

In helices, the side chains (R groups) of the amino acids are oriented outward from the helical axis: If you imagine looking down the helical axis, the side chains would appear as spokes radiating out from the central helical core. Due to the helical twist, the side chains are staggered along the length of the helix, so that they are not directly above or below each other, and they are tilted back towards the N-terminus, giving them a slightly downward orientation relative to the helical axis.

Figure Figure 2.19 shows an optimal α-helix made with a Python library that makes it possible to build polypeptides specifying the sequence and the geometry (in terms of ϕ and ψ). The polypeptide is composed of consecutive Alanine residues with $\phi = -60°$ and $\psi = -45°$, and it is shown with different representations:

- The licorice representation, also known as "ball-and-stick", where atoms are small spheres and bonds are cylinders, and everything has the same diameter, is very useful to look at molecular connections.
- Ribbon-like and/or wireframe representations are very useful to understand the organisation in terms of secondary structures.
- Drawing atoms as van der Waals spheres, also known as space-filling representation, makes it possible to appreciate the efficient packing of folded structures, but it is otherwise rather limited in its utility since it leaves only the protein surface visibile.

[15] Short segments of left-handed α-helices can occur in glycine-rich segments, since Gly is the only achiral amino acid.

[16] Short π-helices are often found flanked by α-helices and near functional sites of proteins, but their identification has sparked some debate (see *e.g.* Weaver [2000], Fodje and Al-Karadaghi [2002], Zubcevic and Lee [2019]).

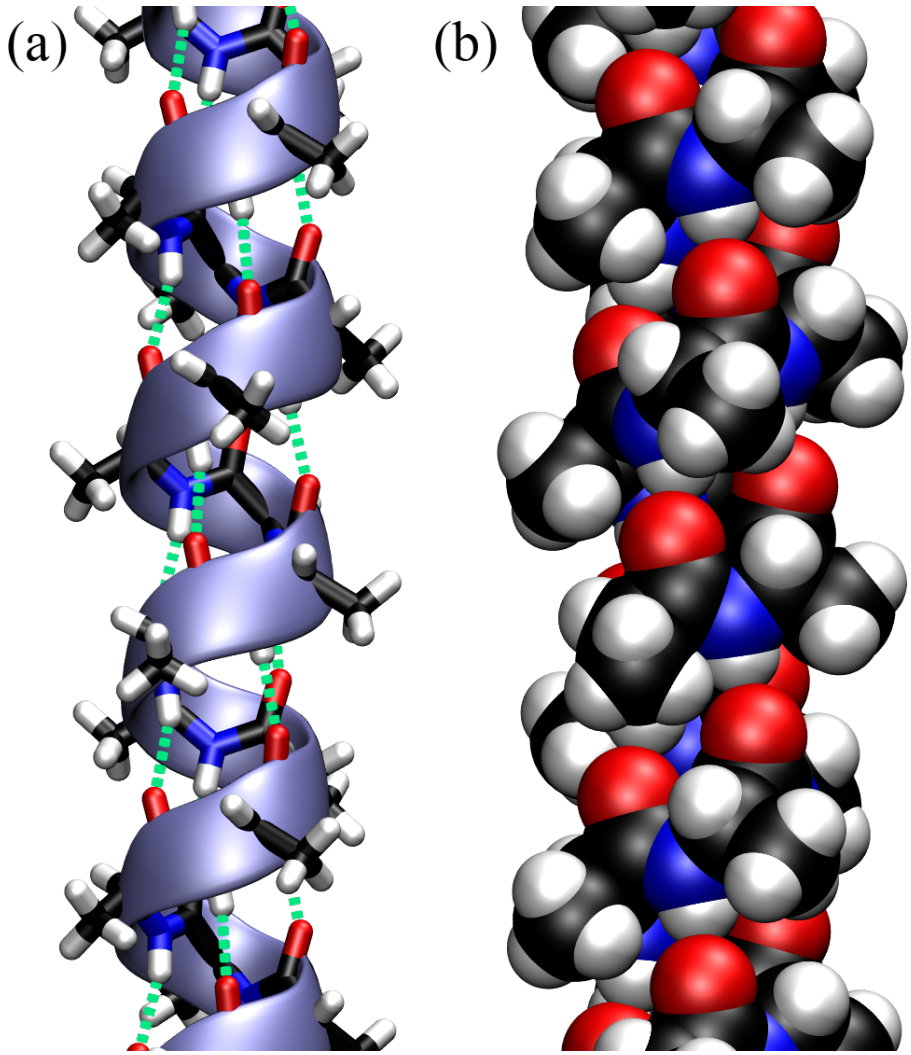

Figure 2.19: An α-helix made by a polypeptide made of consecutive Ala. (a) Licorice representation, where the atoms and the bonds have the same diameter, superimposed with a ribbon that follows the backbone and shows the helical turns. The hydrogen bonds that stabilise the structure are shown with green dashed lines. (b) The same helix with the atoms represented as van der Waals spheres.

2.6.2 β-structures

β-structures (or β-sheets) comprise the other class of common secondary structures. β-structures are formed by β-strands, which are stretchs of polypeptide chain, typically 5-10 amino acids long, in an extended conformation. Unlike the helical structure presented above, β-strands are not stable *per se*, but they are stabilised by hydrogen bonds formed with other strands. In fact, two or more β-strands can connect laterally by backbone hydrogen bonds to form a sheet-like array termed β-sheet. Since the backbone has a directionality ($N \rightarrow C$), the lateral connection joins strands that either run in the same direction or not:

- In an antiparallel β-sheet, which is the most common β motif in proteins, adjacent strands have alternating N-terminus and C-terminus orientations.
- In a parallel β-sheet, the N-termini of adjacent strands are oriented in the same direction.

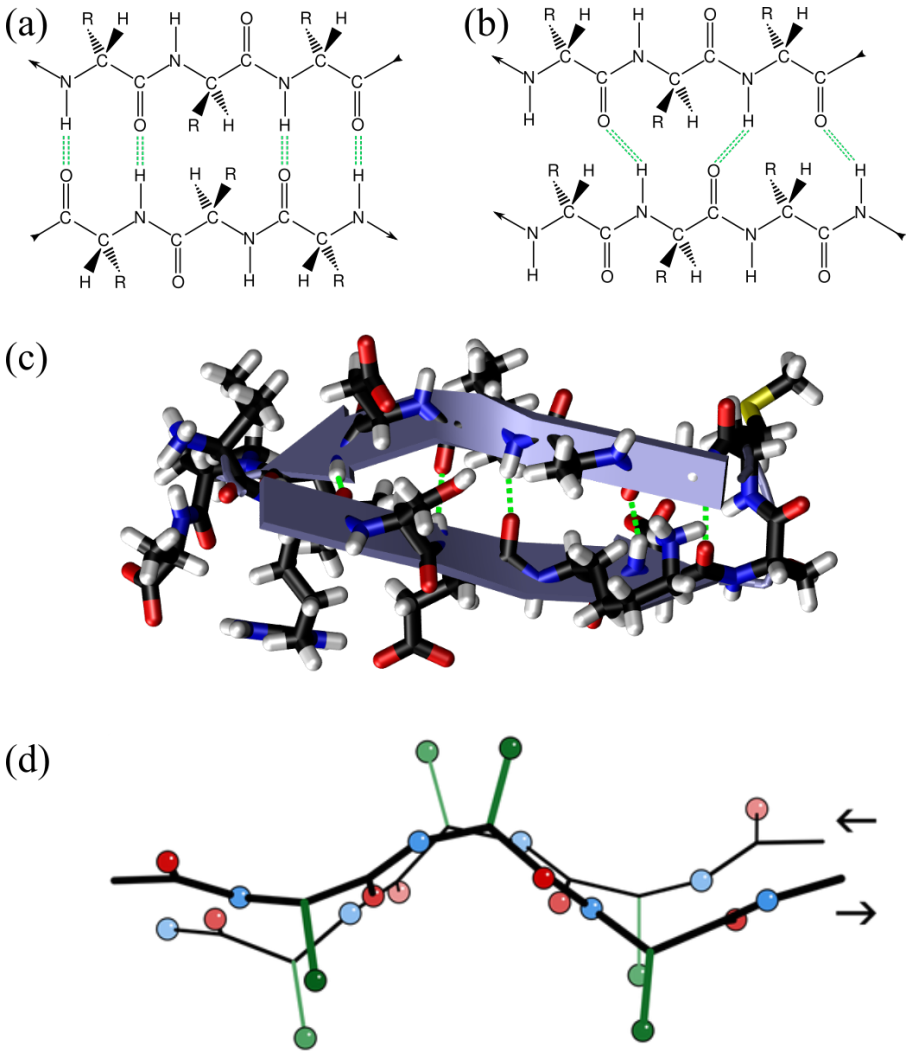

Figure 2.20: Two segments of a chain form a β-sheet by connecting through hydrogen bonds (green dashed lines). The arrows at the beginning and at the end of the segments show the chain directionality. The two segments run anti-parallel and parallel to each other in (a) and (b), respectively (adapted from here). In (c) the X-ray structure of the first 18 residues of Fumarase C where the ribbon representation ("Cartoon" in VMD) is superimposed to the atomic structure and clearly shows the presence of a β-hairpin: two anti-parallel β-strands connected by a short turn. Panel (d) shows two anti-parallel β-strands fragments taken from the crystal structure of the Micrococcus Lysodeikticus catalase, where the hydrogens are omitted for clarity, and only the first carbons of the side-chain carbon are shown (in green). Credits to Dcrjsr via Wikimedia commons.

The hydrogen bonds form between the carbonyl oxygen of one amino acid in a strand and the amide hydrogen of an amino acid in the adjacent strand. As shown in Figure 2.20, in antiparallel

sheets, these hydrogen bonds are more linear and stronger, contributing to greater stability compared to the less linear hydrogen bonds in parallel sheets. Parallel β-sheets containing 4 or fewer strands are not common, which may be due to the lower stability provided by the sub-optimal geometry of their hydrogen bonds. Figure 2.20(c) shows a 3D conformation of a short protein fragment that adopts a rather common β-sheet conformation called "β-hairpin", where two anti-parallel β-strands are connected by a short turn.

The last panel of Figure 2.20 shows two other important properties of β-sheet:

1. In a fully extended β-strand, the side chains of amino acids point straight up and straight down in an alternating fashion. When β-strands align to form a sheet, their C^α atoms are positioned next to each other, with the side chains from each strand pointing in the same general direction. As a result, if a side chain points straight up, the bond to the carbonyl carbon must angle slightly downward due to the tetrahedral bond angle, endowing the strands with a "pleated" appearance[17].

2. β-strands usually have a right-handed twist arising from a combination of the chiral nature of L-amino acids, geometric constraints of peptide bond angles, and the need to optimize hydrogen bonding and minimize steric hindrance (see *e.g.* Chothia [1973]). The hydrogen bonds that stabilize a sheet do not eliminate the twist of individual strands but rather accommodate it, leading to a twisted sheet. Parallel strands tend to be more twisted.

The preferred dihedral angles of parallel and anti-parallel β-sheets are reported in the table below:

Type	ϕ	ψ
Parallel, $\uparrow\uparrow$	$-120°$	$115°$
Anti-parallel, $\uparrow\downarrow$	$-140°$	$135°$

2.6.3 Irregular secondary structures

Irregular secondary structures in proteins are regions that do not conform to the regular patterns of α-helices or β-sheets. These structures include loops, turns, and coils, and they play important roles in the overall shape and function of proteins. First of all, these structures can serve as linkers or hinge regions that facilitate the movement of domains within a protein, and make it possible to adopt multiple conformations necessary for function.

Moreover, they are often exposed to the solvent, where they can form hydrogen bonds that compensate for the lack of regular secondary structure, mitigating the high energy cost associated with broken hydrogen bonds. This, in turn, makes them accessible for interactions with other molecules, such as substrates, inhibitors, and other proteins.

Here is a breakdown of the most common irregular secondary structures:

- Loops are flexible regions that connect α-helices and β-sheets. They do not have a regular, repeating structure, and often occur on the surface of proteins and are involved in interactions with other molecules or proteins. They can also contribute to the active sites of enzymes.
- Turns are short sequences of amino acids that cause a sharp change in direction of the polypeptide chain. They typically involve 4-5 amino acids and are critical for the compact folding of proteins. They often occur at the ends of β-strands, facilitating the formation of antiparallel β-sheets through the formation of β-hairpins (see Figure 2.20(c) for an example). Some very short turns (4 AAs) are known as β-turns and are stabilised by a hydrogen bond.
- Coils are non-repetitive, flexible regions without a defined secondary structure. They are sometimes referred to as random coils, though they are not truly random. Coils provide flexibility and enable conformational changes in proteins. They often link regular secondary structures and can be involved in binding and recognition processes.

[17] The pleating is so marked that sometimes β-structures are called β-pleated sheets.

2.7 Hydrophobic interactions

As we will see in later lectures, the main driver for protein compaction (*folding*) is hydrophobicity. In general, a molecule's solubility in water is a crucial property that can be determined experimentally. By comparing the concentrations of various substances dissolved in water, we can ascertain which molecule is more soluble. This comparison reveals water's affinity for different solutes, essentially indicating which substances are more favorably integrated into the aqueous environment. Comparing the outcome of such experiments on, say, sugar and oil would make it clear that the former is much more "hydrophilic" (water-lover) than the other, which has a strong hydrophobic ("afraid of water") character. Systematic experiments show that uncharged molecules or chemical groups that cannot form hydrogen bonds are hydrophobic and, when put in water, tend to shy away from the water molecules, clustering together.

In order to explore this effect, we consider hydrocarbons as model systems, since many non-polar amino acids feature hydrocarbon side chains (see Section 2.1.1), and refer to the free-energy difference of transferring a substance from a nonpolar or slightly polar solvent to water, ΔG. The nonpolar solvent containing the solute molecules, which in the simplest case can be the bulk liquid of the substance of interest, is put in contact with water through a semipermeable membrane that allows the transfer of the solute only. When the equilibrium is reached, the chemical potential of the solute is the same in the bulk and in water, *viz.*

$$\log c_w/c_0 + \beta\mu_{\mathrm{ex},w} = \log c_b c_0 + \beta\mu_{\mathrm{ex},b}, \tag{2.11}$$

where $\beta = 1/k_B T$, the w and b subscripts refer to the water and bulk systems, respectively, c_i is the concentration of the substance of interest in the system i, c_0 is a constant to make the argument of the logarithms dimensionless, and $\mu_{\mathrm{ex},i}$ is its excess (with respect to the ideal gas) chemical potential. The difference between the excess chemical potentials of the two systems can be evaluated from the (experimentally measurable) values of c_w and c_b through

$$\mu_{\mathrm{ex},w} - \mu_{\mathrm{ex},b} \equiv \Delta\mu = k_B T \log\left(\frac{c_b}{c_w}\right). \tag{2.12}$$

This quantity is the free-energy cost of moving one molecule from the bulk to the water, also known as the solvation free energy, ΔG[18]. In turns out that this cost is approximately proportional to the molecule's accessible nonpolar surface area A, as described by the relationship $\gamma = \Delta G/A \sim 0.02$ kcal/molÅ^2, where γ is the surface (or interfacial) free energy (see Figure 2.21). This relationship implies that larger nonpolar surface areas result in greater free-energy costs for solvation. Interestingly, the free-energy cost increases with temperature, highlighting that the entropic contribution, $-T\Delta S$, plays a major role in the process.

But what causes entropy to decrease when a hydrophobic molecule is inserted into an aqueous solvent? The effect arises from the high cost of breaking hydrogen bonds in liquid water. Indeed, when a nonpolar solute is introduced, water molecules reorient around the solute to maintain as many hydrogen bonds as possible in order to avoid breaking existing hydrogen bonds, which would be energetically very expensive. This reorientation, while preserving hydrogen bonds, restricts the movement of water molecules and effectively "freezes" their orientations, leading to a decrease in entropy. It is these ice-like water molecules that causes the anomalously high heat capacity of hydrocarbons such as cyclohexane, C_6H_{12}, which can increase by up to an order of magnitude in an aqueous solvent ("iceberg effect").

Ice-like does not mean ordered

Water molecules take "frozen" orientations close to a hydrophobic surface to avoid breaking hydrogen bonds, but they do so without becoming crystalline: there is no long-range translational order, but few layers of molecules that adopt orientations that are compatible with HBs close to the underlying surface.

[18]Note that in SI units $\Delta G = N\Delta\mu$, while the two coincide if expressed in kcal / mol (or kJ / mol).

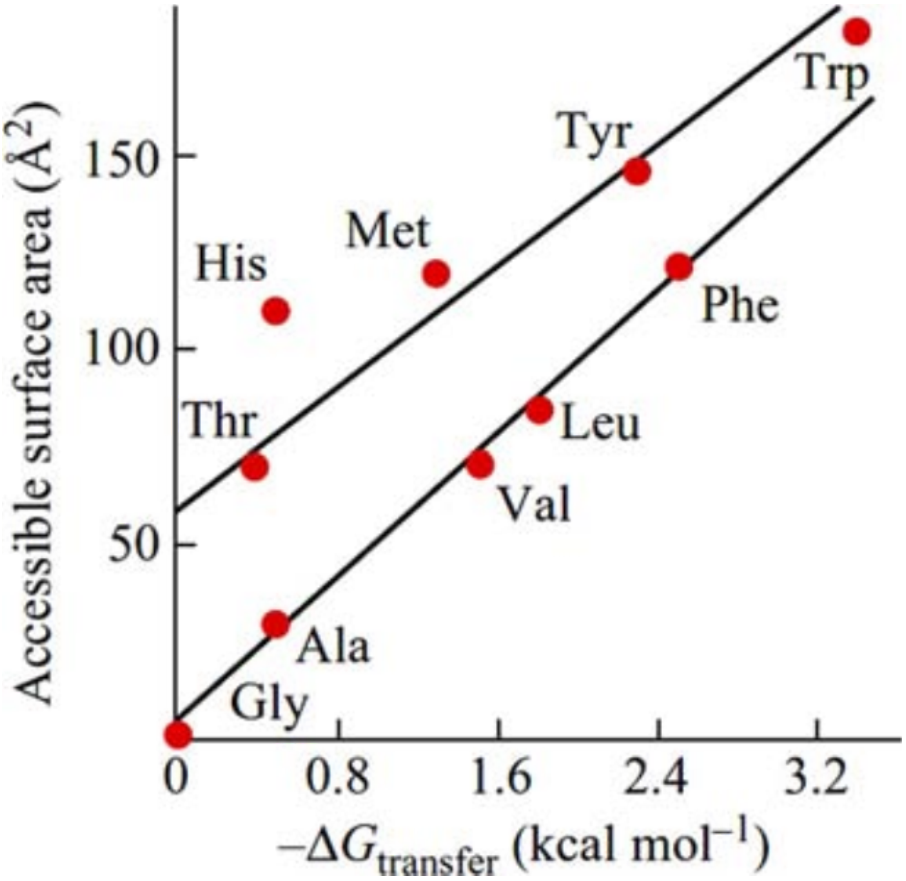

Figure 2.21: The solvation free energy as a function of the accessible surface area of amino acid side chains. Note that for both quantities the value associated to glycine (which has no side chain) has been substracted prior to plotting. The side chains of Ala, Val, Leu, Phe consist of hydrocarbons only; those of Thr and Tyr additionally have one OH-group each, Met an SH-group, Trp an NH-group, and His an N-atom and an NH-group. Therefore, their accessible non-polar area is smaller than their total accessible surface. Taken from Finkelstein and Ptitsyn [2016].

The hydrophobic effect, which is the observation that non-polar substances are immiscible with water because of entropic effects, is very important in determining the stability of proteins. In fact, proteins that live in aqueous environments[19] have a core composed by non-polar AAs whose formation is, by and large, driven by hydrophobic forces. The origin of these forces can be understood by considering two hydrophobic objects with accessible surface area A_1 and A_2, respectively. The cost of solvating the two objects when they are far from each other is $\Delta G_{\text{sep}} \approx \gamma(A_1 + A_2)$, where $(A_1 + A_2)$ is the extent of the hydrophobic surface. However, if the two objects approach each other, the nonpolar surface area exposed to water decreases, leading to fewer water molecules with 'frozen' orientations, which in turn reduces the entropic cost associated with solvating the objects. As a result, the solvation free energy decreases, $\Delta G_{\text{bonded}} = \gamma A_{12} < \Delta G_{\text{sep}}$. Therefore, a state where hydrophobic surfaces are close enough that no water molecule can slip in is thermodynamically favoured with respect to a state where the same surfaces are far from each other.

2.7.1 Structured water

Around hydrophilic regions of the protein, water molecules form hydration shells through hydrogen bonds and electrostatic interactions while, as we have just seen, near hydrophobic regions, water molecules cannot form hydrogen bonds with the protein surface, but they reorient to maximize hydrogen bonding among themselves while minimizing disruption. In both cases there is the appearance of *structured water*, which is an organized arrangement of water molecules in the immediate vicinity of the protein surface.

The hydration layers around the protein have a restricted mobility and orientation that reduce their ability to respond to an applied electric field, in turn affecting the dielectric constant, which takes values that, depending on the distance from and type and geometry of the protein's surface, can be much lower than in the bulk (where $\epsilon_r \approx 80$). The reduced dielectric environment enhances electrostatic interactions between charged groups, possibly influencing protein folding, stability, and interactions with other molecules, such as substrates, inhibitors, or other proteins.

Moreover, structured water around proteins exhibits slower dynamics compared to bulk water, which significantly impacts protein behavior and function. For instance, it can also influence the rates of enzymatic reactions by modulating the flexibility and movement of active site residues.

2.8 Tertiary structure

The tertiary structure of proteins refers to the overall three-dimensional shape of a single polypeptide chain, resulting from the folding and interactions of its secondary structural elements. Let's start by considering what is the geometry of the single residues that are part of well-folded proteins[20], which, as discussed Section 2.3.1, at first order we can assume to be fully described by the ϕ and ψ dihedrals.

Figure 2.22 shows the general[22] Ramachandran plot of 8000 well-characterised proteins, as well as the (ϕ, ψ) positions of the optimal secondary structure elements (perfect helices and β-sheets). It is evident that the majority of residues have dihedrals that are compatible with regular secondary structures. However, the probability distribution is not perfectly peaked at the optimal values (particularly for β-sheets), and it also exhibits considerable width. One obvious reason for this is that while there is a single "optimal" conformation for a given secondary structure, it is entropically favorable to adopt conformations that are close to this optimal state without incurring significant energetic penalties.

Additionally, and more interestingly, secondary structures interact with each other and pack together to fold into the native state of the protein. To achieve this, residues must accommodate various

[19] As we will briefly discuss below, not all proteins are in direct contact with the cytoplasm or with another aqueous environments: Section 2.8.1 and Section 2.8.2 proteins often perform their biological function in hydrophobic (or not-so-hydrophilic) environments.

[20] For the moment we leave out proteins that are intrinsically disordered or have intrinsically disordered domanis, see [21].

[22] I remind you that a "general" Ramachandran plot shows the ϕ and ψ angles of residues that are not Gly, Leu, Val, Pro, or come before a Pro.

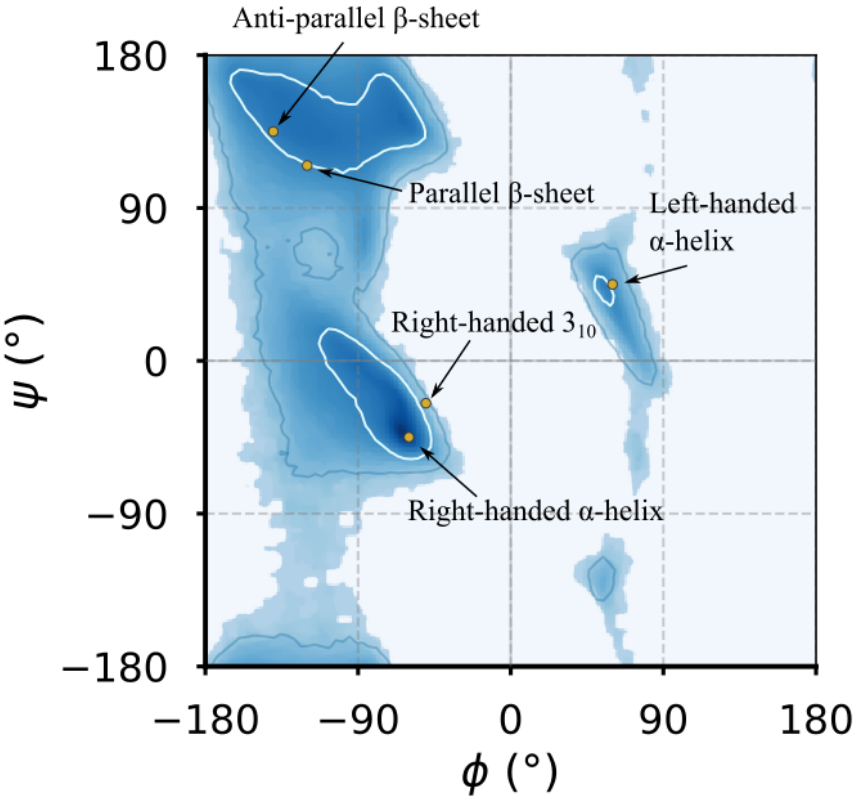

Figure 2.22: The Ramachandran general plot of the Top8000 PDB data set, with contour lines highlighting the 80th and 95th centile in blue and white, respectively. The yellow circles are the optimal dihedral angles of the regular secondary structures introduced in the previous section, where those of the left-handed α-helix are the same as the α-helix, taken with the opposite signs.

constraints and interactions imposed by neighboring structures. These interactions include steric hindrances, electrostatic forces, and hydrogen bonding, which can cause deviations from the optimal dihedral angles. As a result, the conformations observed in Ramachandran plots reflect a balance between maintaining the overall stability of the secondary structures and allowing the flexibility required for proper folding and function, leading to broader, less sharply peaked (ϕ, ψ) distributions.

The regions in the Ramachandran plot that are far from those associated to regular secondary structures are usually populated by residues that are in an irregular (or even disordered) structure such as a coil or a loop.

Most proteins are generally either densely packed structures or composed of densely packed domains. The primary driving force behind the formation of these packed domains is the Section 2.7, which tends to compact the hydrophobic side chains into a central core that excludes water molecules. In terms of purely hydrophobic interactions, the resulting "molten globule" would be liquid, resembling an oil droplet. However the presence of additional interactions - such as van der Waals interactions, hydrogen bonds, ionic bonds, and disulfide bridges (which we will discuss shortly) - further solidifies the protein, endowing it with the stability and shape-retaining properties essential for its biological function (Finkelstein and Ptitsyn [2002]).

Now I will (very!) briefly discuss how secondary structures pack and layer to form the native structures of proteins. Following Finkelstein and Ptitsyn [2002], we classify folded proteins into three macrocategories: fibrous, membrane and globular proteins[23].

2.8.1 Fibrous proteins

Fibrous proteins are a class of proteins characterized by their elongated, fiber-like shapes. Their primary role is to provide mechanical support, strength, and elasticity to cells and tissues. Fibrous proteins are typically very large and form huge aggregates, making them insoluble in water. Their primary structure often contains repetitive sequences that facilitate the formation of regular secondary structures and the interactions between adjacent chains which lead to the formation of higher-order aggregates (Section 2.9 in protein lingo). These proteins are essential for maintaining structural integrity, protecting tissues, allowing flexibility, and playing a crucial role in tissue repair and wound healing.

β-structural proteins

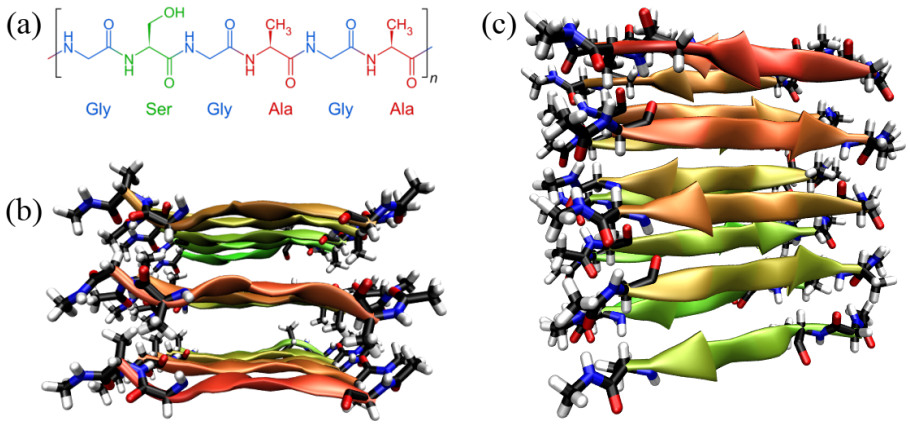

Figure 2.23: Silk fibroin: its primary structure (a) (credits to Sponk via Wikipedia Commons), and a fragment of FOSSEY, S.A.; NEMETHY, G.; GIBSON, K.D.; SCHERAGA, H.A. [1991][25] as seen from the top (b) and from the side (c).

Figure 2.23 shows the primary and tertiary structure[26] of a representative of β-structural proteins:

[23] For the moment we leave out proteins that are intrinsically disordered or have intrinsically disordered domanis, see [24].

[26] I did not find any proper PDB structure for silk fibroin, but only a file containing the "theoretical model coordinates" of a fragment of silk fibroin.

silk fibroin. Silk fibroin is a fibrous protein produced by spiders and silkworms. Silk fibroin consists of repetitive sequences rich in glycine, alanine, and serine: a six-residue block like the one shown in the figure, where Gly alternates with larger residues, is repeated 8 times. These octads are repeated tens of times, interspersed by less regular sequences. The mostly small amino acids of the sequence facilitate tight packing and formation of β-sheets, forming extensive hydrogen-bonded networks that contribute to the material's strength and stability. In turn, as shown in the figure, the β-sheets are stacked together, leading to highly ordered, quasi-crystalline regions that are interspersed with less ordered amorphous regions. Those are formed by irregular parts of the fibroin itself, as well as by the disordered matrix protein sericin.

α-structural proteins

Figure 2.26: Coiled coils. Figure **??** has been generated by using the structure solved in (Lee et al. [2012]), while Figure 2.26 has been adapted from Lapenta et al. [2018].

A typical example of α-structural fibrous proteins is keratin, a protein crucial for the structural integrity and protection of vertebrate tissues, including hair, nails, feathers, horns, and skin. Composed primarily of amino acids with a high cysteine content, keratin forms α-helices that assemble into left-handed supercoiled helices called "coiled coils".

Figure **??** shows a part of the coiled coil generated by interacting regions of two proteins that compose keratin. As illustrated in Figure 2.26, the primary structure of proteins forming coiled coils is characterized by a specific repeating pattern of amino acids, known as a heptad repeat. This repeating sequence consists of seven amino acids, denoted as (a-b-c-d-e-f-g), where positions a and d are typically occupied by hydrophobic residues (such as leucine, isoleucine, valine, or methionine) that interact with each other in the core of the coiled coil, stabilizing the structure through hydrophobic interactions. Indeed, when one of the helix is rotated by 20°, the side chains of amino acids at the a and d positions in one α-helix (the "knobs") project outward from the helix and fit into the spaces ("holes") formed by the side chains of the neighboring helices.

Note

In coiled coils, the α-helix has 3.5 residues per turn instead of 3.6 as in normal helices. This distortion, proposed for the first time by Crick [1953], is what enables the "knobs into holes" packing.

In some cases coiled coils further assemble into durable fibrils reinforced by disulfide bonds provided by cysteine residues (identified by the sulfur atoms shown as yellow van der Waals spheres in Figure **??**). This structure gives keratin its strength, rigidity, and insolubility, making it resistant to environmental stress.

Disulfide bonds

Disulfide bonds are covalent linkages formed between the sulfur atoms of two cysteine residues ($-C^{\beta}H_2 - SH$ side chain) within a polypeptide chain or between different polypeptide chains. In the formation of disulfide bonds, the thiol groups $-SH$ of two cysteine residues undergo an oxidation reaction, resulting in a disulfide linkage ($-S - S-$). At room temperature this process does not occur on reasonable timescales, which is why, in the cellular environment, the formation of disulfide bonds is catalyzed by enzymes, particularly in the endoplasmic reticulum (ER) of eukaryotic cells. For instance, the enzyme protein disulfide isomerase (PDI) facilitates the formation and rearrangement of disulfide bonds, catalyzing the oxidation of thiol groups and rearranging incorrect disulfide bonds to ensure that only the correct Cys residues are connected.

Collagen

Collagen is a major component of the extracellular matrix in various connective tissues of animals. It is one of the most abundant proteins in vertebrates, making up about a quarter of their total protein mass. Collagen provides structural support, strength, and elasticity to tissues such as skin, tendons, ligaments, cartilage, bones, and blood vessels.

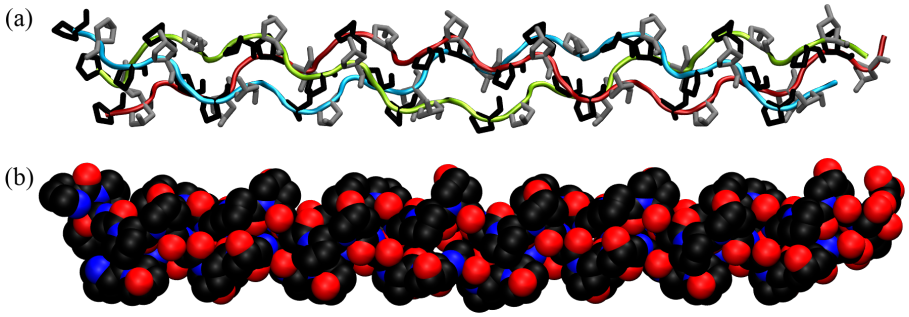

Figure 2.27: The collagen triple helix or tropocollagen, taken from here. In (a) the backbones of the three chains are shown with ribbons of different colours, while prolines and hydroxyprolines are shown in black and silver, respectively. (b) shows the same configuration with atoms as van der Waals spheres.

The primary structure of collagen consists of a repeating tripeptide sequence Gly-X-Y, where X is often proline, and Y is often hydroxyproline, which is a post-translational modification of proline.

Each polypeptide chain containing the repeating sequence forms a left-handed helix with 3 residues per turn, which is stabilized by the kinks induced by the pyrrolidine rings of proline and hydroxyproline residues. However, the single helix does not form any hydrogen bonds and therefore it is not stable in isolation. In turn, as shown in Figure 2.27, three helices can wind around each other in a right-handed fashion to form a superhelix (the *tropocollagen* molecule), stabilised by HBs formed by the Gly residues[27]. Moreover, the three chains are closely packed together, and the interior of the triple helix is very small and hydrophobic, requiring every third residue of the helix to interact with this central region. As a result, only the small hydrogen atom of glycine's side chain can fit and make contact with the center: any substitution of Gly with a larger amino acid can lead to distortions or disruptions in the triple helix, potentially leading to diseases such as osteogenesis imperfecta.

Eat your vegetables!
The hydroxylation of proline and lysine residues in collagen synthesis, a process catalyzed by the enzymes prolyl hydroxylase and lysyl hydroxylase, requires the presence of ascorbic acid, commonly known as vitamin C. In the absence of adequate ascorbic acid, these enzymes cannot function properly, leading to the production of collagen with reduced stability. This deficiency results in weakened connective tissues, clinically manifesting as scurvy, a disease historically common among sailors who lacked access to vitamin C-rich foods, characterized by symptoms such as weakness, bleeding gums, joint pain, delayed wound healing, and petechiae.

Triple helices, in turn, associate into higher order structures (collagen fibrils) that are held together by covalent cross-links catalysed by an enzyme that join together modified residues. Note that the process of collagen formation is not spontaneous but comprises many steps that require the cell machinery (see Finkelstein and Ptitsyn [2002] and references therein).

[27] Although hydroxyproline has a $-OH$ group that can form a hydrogen bond, there is strong evidence that its contribution to the stability of the triple helix comes from stereoelectronic effects rather than hydration (Kotch et al. [2008]).

2.8.2 Membrane proteins

Biological membranes are thin, flexible structures that form the boundaries of cells and organelles (which are cell compartments), regulating the movement of substances in and out. They play crucial roles in cellular communication, energy transduction, and maintaining homeostasis. Membranes are composed of

1. lipid bilayers, which consist of two layers of phospholipids with hydrophilic heads facing outward and hydrophobic tails facing inward, creating an insulating barrier;
2. membrane proteins, which are embedded within or associated with the lipid bilayer, acting as conductors, facilitating the selective transport of molecules across the membrane and transmitting signals between the cell's internal and external environments.

Membrane proteins are further classified in integral and peripheral proteins: integral membrane proteins are embedded within the lipid bilayer, often spanning it multiple times, while peripheral membrane proteins are attached to the membrane surface.

The transmembrane parts of the membrane proteins have structures that are moderately more complex than those of fibrous proteins. Since the interior of a membrane is a fatty (hydrophobic) environment, proteins need to adopt a structure where all hydrogen bonds are formed, and no polar groups are exposed. Therefore, these proteins are poorly water-soluble, and basically all their residues are packed into regular secondary structures (α-helices or β-sheets) that are, in turn, organised into higher order structures, with the most common being helix bundles and β-barrels.

Helix bundles

Helix bundles are structural motifs composed of several α-helices that are arranged in a specific, often tightly packed, formation. These bundles are a common feature in the architecture of membrane proteins, particularly those that span the lipid bilayer multiple times. The helices in these bundles are typically amphipathic, meaning they have both hydrophobic and hydrophilic regions. This property allows them to interact favorably with both the hydrophobic core of the lipid bilayer and the aqueous environments on either side of the membrane.

The formation of helix bundles in membrane proteins is driven by several factors. Firstly, the hydrophobic effect plays a critical role; the hydrophobic side chains of the α-helices tend to aggregate together to minimize their exposure to water. This aggregation helps to stabilize the protein structure within the lipid bilayer. Additionally, other interactions such as hydrogen bonds, salt bridges, and van der Waals forces between the helices further stabilize the bundle.

The presence of protein with helix bundles influence the properties of the membrane itself, since they can affect the fluidity and curvature of the membrane, which in turn impacts the behavior of other membrane-associated molecules and the overall dynamics of the lipid bilayer. The integration of helix bundles into the membrane can create microdomains or rafts that serve as organizing centers for cellular signaling pathways.

Structurally, the tight packing of helices provides stability to the protein, ensuring that it maintains its correct shape and function within the dynamic environment of the cell membrane. Functionally, the arrangement of helices can create specific sites for binding ligands, ions, or other molecules, which is essential for the protein's role in processes such as signal transduction, transport, and enzymatic activity. For example, in G-protein-coupled receptors (GPCRs), which are integral membran proteins, the helical bundle forms a pocket that can bind other molecules, triggering conformational changes that transmit signals across the membrane, initiating a multitude of cellular responses and signaling cascades within the cytoplasm. Thus, it should come as no surprise that this class of proteins is the target of many drugs[28].

[28] "Many druggable targets for treatment of common diseases involve G-protein-coupled receptors (GPCRs) that mediate therapeutic effects of $\approx 34\%$ of the marketed drugs. The sales of GPCR targeting drugs represent a global market share of over 27% [more than 180 billion US dollars]" Hauser et al. [2018]).

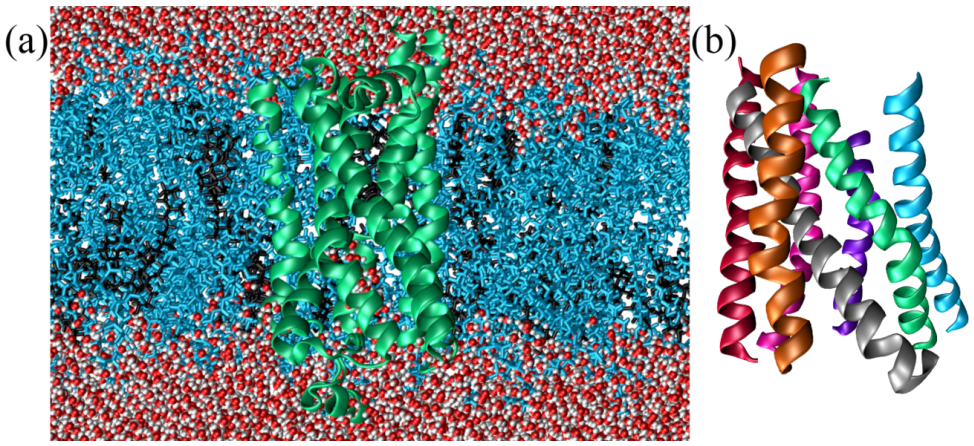

Figure 2.28: A GPCR protein (taken from here) embedded in a membrane composed of a double layer of cholesterol (in black) and phosphatidylcholine (POPC, in cyan) in a 7:3 ratio. Here both surfaces of the membrane are put in contact with water. In (a) the lipids and the water (in red and white) are shown explicitly, while in (b) only the helical bundle is shown, to highlight the angles between the helices (or part thereof).

Figure 2.28 shows the $\beta(2)$ adrenergic receptor, a GPCR protein, embedded in a lipid membrane sandwiched between two water layers. A GPCR is formed by seven transmembrane α-helices connected by alternating intracellular and extracellular loops. The helices are tilted with respect to each other, so that their side chains form the "knobs into holes" packing introduced Section 2.8.1 (Pratap et al. [2013]).

β-barrels

β-barrels are cylindrical structures that are a common feature in the outer membranes of gram-negative bacteria, mitochondria, and chloroplasts. Each β-barrel is formed by β-strands that are hydrogen-bonded to each other in a sheet and then wrapped around to form a closed barrel. The edges of the β-sheet are connected by short loops that create a continuous surface.

The formation of β-barrels is driven by the hydrophobic effect, which causes the non-polar side chains to interact with the lipid bilayer, stabilizing the structure within the membrane. Additionally, the hydrogen bonds between β-strands provide structural integrity, making β-barrels very stable. Structurally, they provide a rigid and stable scaffold that can maintain their shape even under varying conditions. This rigidity is essential for their role as transporters, ensuring that the passage of molecules is controlled and efficient. Functionally, β- barrels can form pores or channels that are highly selective, allowing only specific substances to pass through. This selectivity is achieved through the size and charge of the residues lining the inside of the barrel, which can create a filter for specific molecules.

Figure 2.29 shows a E. coli porin, a class of membrane proteins that form large, water-filled channels through the outer membrane of several microorganisms and facilitate the passive diffusion of small molecules, such as ions, nutrients, and metabolic waste products, across the membrane. The figure shows only the protein, with the top view highlighting the large pore in the middle, and the side view demonstrating the regular pattern formed by the anti-parallel β-strands. In the figure residues are coloured according to their hydrophobic character[29], which determines an alternating pattern that is typical for β-barrels, where the hydrophobic residues face outward to interact with the lipid bilayer, and the hydrophilic residues face inward to create a water-filled channel.

[29]The binary classification into hydrophobic/non-hydrophobic is somewhat ambiguous, but it will serve its purpose here.

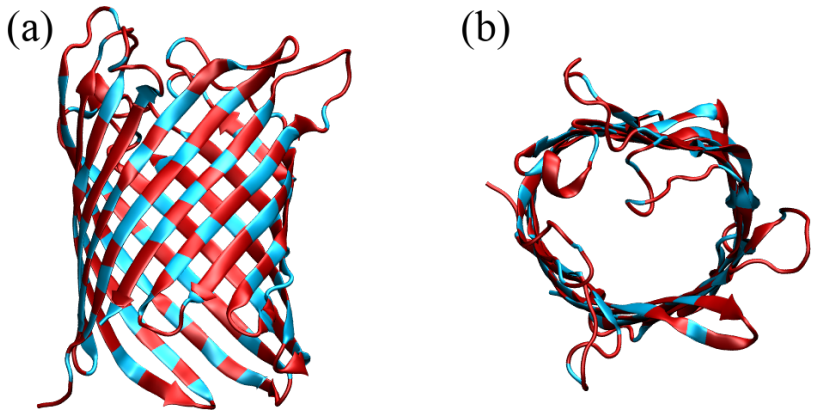

Figure 2.29: The crystal structure of an E. coli porin seen from the top (a) and from the side (b) (Korkmaz-Özkan et al. [2010]. Hydrophobic residues are coloured in red, the rest in cyan. In (a), the water is at the top and the cytoplasm is at the bottom.

2.8.3 Globular proteins

Globular proteins are a diverse group of proteins characterized by their compact, roughly spherical shapes and high solubility in water. Unlike in fibrous proteins, in globular proteins the three-dimensional arrangements of the secondary structure motifs are more complex, with hydrophobic amino acid residues typically buried in the protein's interior and hydrophilic residues exposed to the solvent, making them soluble and functional in the cell's aqueous environment. Globular proteins perform a wide range of functions, including enzymatic catalysis (*e.g.*, lysozyme and DNA polymerase), transport and storage of molecules (*e.g.*, hemoglobin and myoglobin), immune responses (*e.g.*, antibodies), and regulatory roles (*e.g.*, hormones like insulin).

Water-soluble globular proteins are the most studied because they are the most amenable to experimental research, and they are by far the most known in terms of structure. Depending on their size, they either pack in a single, compact globule with a radius of a few nanometers, or into multiple globules (called *domains*) connected together by loops and coils. Taken all together, the majority ($\approx 90\%$) of the residues are organised in secondary structural motifs: $\approx 30\%$ in helices, $\approx 30\%$ in β-strands, and $\approx 30\%$in loops and turns. Therefore, a hierarchical classification of proteins (which goes further than the fibrous/membrane/globular split) can be made in terms of what types of secondary structures they contain, and how those are packed together. There are several databases where proteins are classified into classes and subclasses according to their structure, packing and homology (similarity) with respect to other proteins, with the classification criteria differing from database to database. Here I will follow Schlick [2010] and describe the top levels of the SCOP database[30]. For reference, other well-known databases are CATH and ECOD.

The main levels of classification in SCOP are as follows:

- **Class**: The highest level of classification, based on the overall secondary structure content of the proteins. Classes include all-α proteins, all-β proteins, α/β proteins (where α-helices and β-strands are interspersed), and $\alpha + \beta$ proteins (where α-helices and β-strands are segregated).
- **Fold**: Groups proteins with the same major secondary structures in the same arrangement and with the same topological connections. Proteins within a fold share major structural features even if they do not have sequence similarity.
- **Superfamily**: Includes proteins that share a common fold and are believed to have a common evolutionary origin. Members of a superfamily may have low sequence similarity but have functional and structural features indicating a common ancestry.
- **Family**: Proteins within a superfamily that have clear sequence similarity, indicating a closer evolutionary relationship. Families are groups of proteins that share high sequence identity and often similar functions.

[30]Structural Classification of Proteins database.

While I will not delve[31] too much on the classification, which would be outside of the scope of these lectures, I want to provide some examples of folds (also known as "supersecondary structures"). I will provide one example for each class of proteins, starting with two examples we have already encountered: even though they are not globular (nor water-soluble), they contain some common motifs that are also rather easy to identify:

- Figure 2.28 are transmembrane α-proteins characterised by an up-and-down bundle of seven transmembrane helices. Helical bundles are very common folds in proteins, and this particular one is a fold on its own according to SCOP.
- The Figure 2.29 is a transmembrane β-protein of the Porins superfamily. The main structural motif is a Section 2.8.2, which on SCOP is not a proper fold, since β-barrels can be very different with each other, and are classified according to two numbers:
  - the number of β-strands in the barrel;
  - the "shear number", which is a measure of the staggering of nearby strands.

Figure 2.32: Examples of α/β and $\alpha+\beta$ proteins. The left panels show the protein with a ribbon-like representation, while the right panels show the accessible surface of the same protein from the same proint of view. The two structures have been taken from here and here.

The other two classes, α/β and $\alpha+\beta$, comprise proteins that contain both α-helices and β-sheets. However, they differ in the order with which the secondary structures are arranged:

- in α/β proteins α-helices and β-strands alternate to form layered or barrel structures. Figure **??** shows a classic α/β barrel, which has its own SCOP fold. Note that the β-sheet is formed by parallel β-strands.
- $\alpha+\beta$ proteins contain α-helical and β regions that are well-separated (*i.e.* non-alternating). The SCOP lysozyme-like fold is a common example, and a protein of this type is shown in Figure 2.32.

Folds can be very complex from the topological point of view (especially β-structures), but a more thorough classification is goes beyond the scope of these lessons.

2.9 Quaternary structure

The quaternary structure of a protein refers to the higher-level organization and assembly of multiple polypeptide chains, or subunits, into a single functional complex: unlike the primary, secondary, and tertiary structures, which involve the folding of a single polypeptide chain, the quaternary structure pertains to how these chains interact and fit together. This level of protein structure is crucial for the functionality of many proteins, particularly those that operate as multi-subunit complexes. We have already seen some examples when we discussed Section 2.8.1.

Subunits in a protein's quaternary structure can be identical, known as homomeric, or different, referred to as heteromeric. The interactions between these subunits are stabilized by the same interactions that drive the formation of Section 2.6 and Section 2.8 structures, including hydrogen bonds, van der Waals forces, electrostatic and hydrophobic interactions, and disulfide bridges.

The spatial arrangement of subunits within the quaternary structure is vital for the protein's functionality, allowing the protein to interact correctly with other molecules and perform its biological activities efficiently. For example, the quaternary structure of hemoglobin, which is shown in Figure 2.33 and consists of four subunits in a nearly-tetrahedral arrangement, enables it to bind and release oxygen molecules effectively. Similarly, enzymes that function as multi-subunit complexes often rely on their quaternary structure to bring catalytic sites into proximity, enhancing their catalytic efficiency.

[31] Apparently using "delve" is a big indicator that a piece of text has been written by ChatGPT. Although I have used ChatGPT to polish my writing here and there, I ensure you that this "delve" is my own!

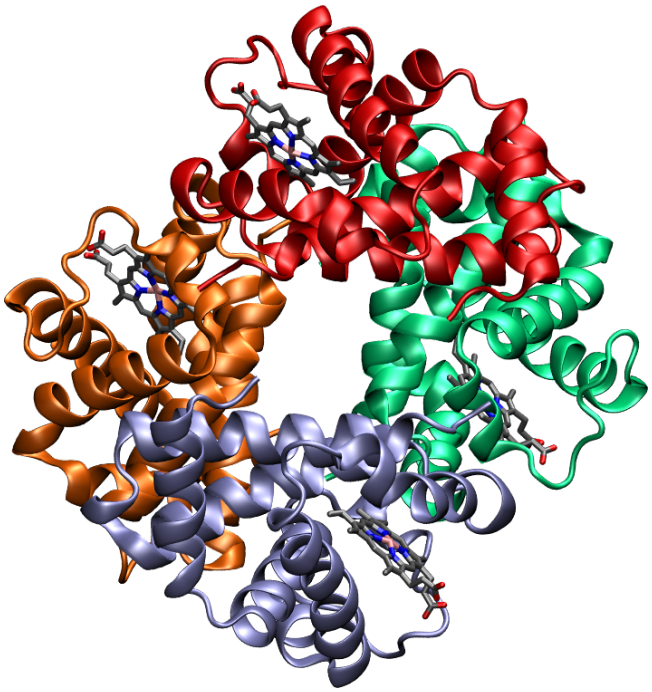

Figure 2.33: The structure of human hemoglobin. The four subunits are shown with different colours, while their bound heme groups are depicted with the licorice representation.

Chapter 3

Nucleic acids

Tip
The main references for this part are Lehninger et al. [2005], and Schlick [2010].

Here I take for granted that you are familiar with the general concepts introduced in the previous chapter regarding the interactions between atoms and molecules.

3.1 The basic structure of DNA and RNA

Deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) are the fundamental molecules that carry and execute the genetic instructions essential for all forms of life. Like proteins, they are macromolecules. However, the repeating monomer is not an amino acid, but a nucleotide, which is a molecule consisting of a a cyclic sugar (a pentose), a phosphate group, and a nitrogenous base. A nucleotide without the phosphate group is called a nucleoside.

In the classic DNA double helix (or *duplex*) described by WATSON and CRICK [1953], a flexible ladder-like structure is formed by two chains (strands) that wrap around a virtual central axis. The two chains' backbones consist of alternating sugar and phosphate units, while the rungs that connect them consist of nitrogenous bases held together by hydrogen bonds and stabilised by stacking interactions. A schematic of a single nucleotide and of the Watson-Crick mechanism are shown in Figure 3.1.

The discovery of the double helix

The discovery of the DNA double helix in 1953 marked a revolutionary advancement in molecular biology, primarily credited to James Watson and Francis Crick. Their success was built on the pivotal contributions of Rosalind Franklin and Maurice Wilkins, whose X-ray diffraction images of DNA provided critical evidence for the helical structure. In particular, these images, among which there was the famous "Photo 51", shown in Figure 3.2, revealed the density and helical form of DNA. Wilkins shared these images, without her direct permission, with Watson and Crick who, thanks to their understanding of chemical bonding, enabled them to propose the accurate double helix model, fundamentally transforming our comprehension of genetic information storage and transmission.

Use this page, prepared by prof. Roberto di Leonardo, to understand how the geometry of the double helix controls its X-ray diffraction pattern.

Let's briefly discuss the main building blocks of a nucleotide.

3.1.1 The pentose

A pentose is a simple sugar (*i.e.* a monosaccharide) containing five carbon atoms. In nucleic acids, pentose is present in its cyclic form, where the ring contains four carbons and one oxygen. In DNA,

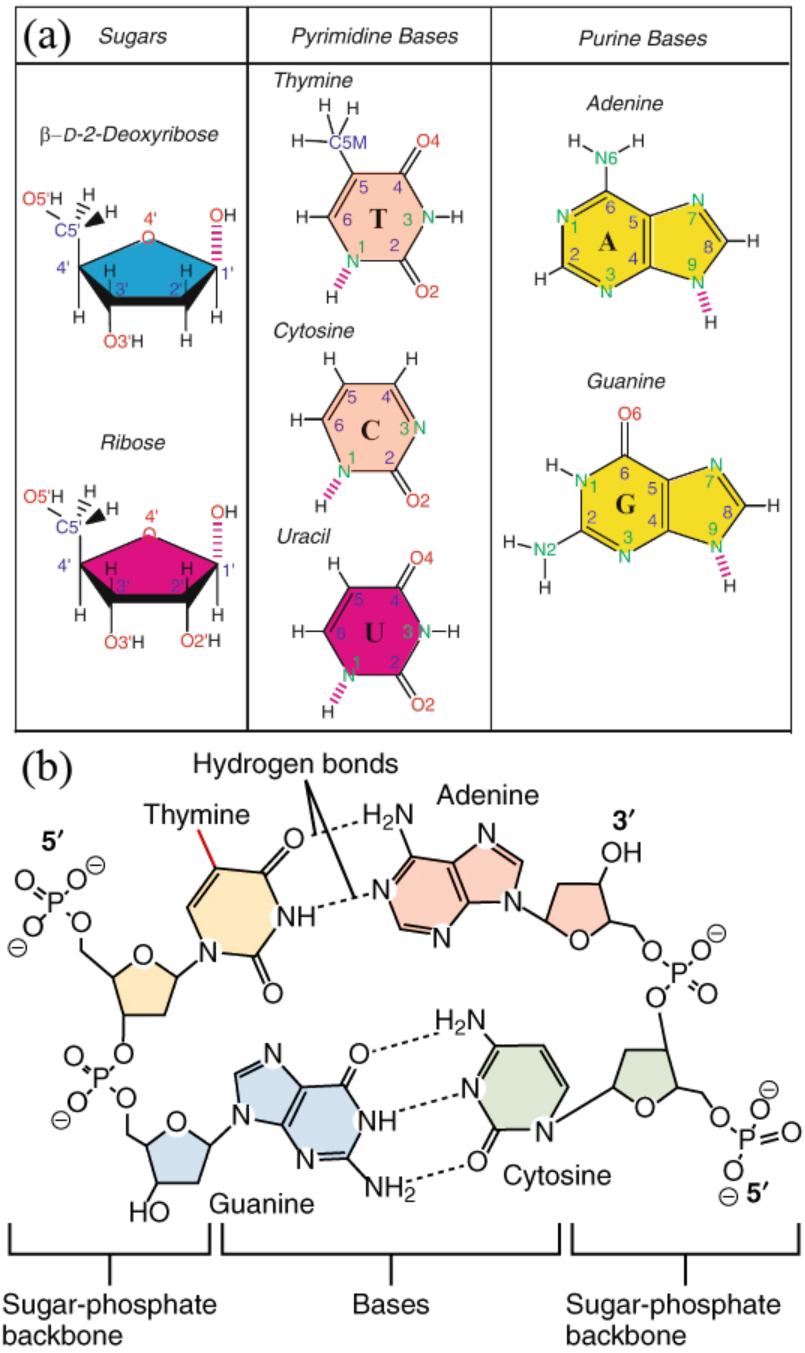

Figure 3.1: (a) The sugar and nitrogenous bases that make up DNA and RNA. Non-hydrogen atoms are labelled, and the broken lines indicate links that bind the compound to the other building blocks. Adapted from Schlick [2010]. (b) A sketch showing two bonded DNA dinucleotides. The presence of a $-CH_3$ group connected to the pyrimidine ring highlighted in red makes the yellow-shaded base a thymine instead of a uracil. Adapted from here.

one of the hydroxyl ($-OH$) groups is replaced by a hydrogen atom. This is not a small change since, as we already know, a hydroxyl group can be involved in a hydrogen bond.

The three-dimensional conformation of the five-carbon sugar ring in nucleotides is non-planar, and the ring can adopt various forms, with the the C2'-endo and C3'-endo puckers being the most common forms for B-DNA and A-RNA (and A-DNA as well), which are helical structure that will be introduced in a moment. These puckering conformations, where the C2' or C3' atoms, respectively, are above the plane formed by the other four atoms, affect the overall structure and flexibility of DNA and RNA molecules and are one of the main source of conformational flexibility in nucleic acids (Schlick [2010]).

3.1.2 The phosphate group

A phosphate group is, by definition, the same for DNA and RNA and is composed by a phosphate atom attached to four oxygens. When in its anion form, a phosphate can be written as $[PO_4]^{-3}$: three of the oxygens are charged, and the fourth one is connected to the phosphate through a double bond. In nucleic acids, the phosphate group form a phosphodiester bond with the 3' and 5' carbon atoms of the pentoses of consecutive nucleosides, joining them together to form the strand's backbone. I note here that you should not imagine the backbone as a rigid, fixed object, but as composed by segments around which the molecule can, to some extent, rotate. Indeed, there are six dihedral angles in the backbone (named, rather unoriginally, $\alpha, \beta, \gamma, \delta, \epsilon$, and ζ) that take different values depending on the local conformation (see Schlick [2010] for more details).

Finally, as shown in Figure 3.1(b), a phosphate that connects two nucleotides has a single residual negative charge. By contrast, phosphates at the beginning of a strand (also known as the 5' end) are doubly charged.

3.1.3 The nitrogenous base

The nitrogenous bases endow the monomer with the exquisite selectivity that underlie the Watson and Crick double helix. Nitrogenous bases found in nucleic acids are

- **Purines**, molecules with five- and a six-membered rings fused together: guanine (G) and adenine (A), present in both DNA and RNA.
- **Pyrimidines**, molecules with a single six-membered rings: cytosine (C), present in both DNA and RNA, thymine (T, DNA-only), uracil (U, RNA-only)[1].

The chemical structure of the bases are shown in Figure 3.1(b). Note that uracil has the same chemical structure as thymine, minus the $-CH_3$ group connected highlighted in red in the figure. Each base is connected to the C1' atom of the corresponding pentose through a nitrogen, which results in a *glycosyl* bond around which the base can rotate, contributing to the flexibility of the polymer. The dihedral angle associated to this rotation is called χ.

In the classic pairing scheme unveiled by Watson and Crick, the geometry of the bases is such that G binds only to C, and A binds only to T (or U in RNA) by means of three and two hydrogen bonds, respectively. Additional non-canonical base pairings that are not only possible, but also biologically relevant exist. Here I will mention wobble base pairs, where G can bind with U[2] with a thermodynamic stability that is similar to that of canonical base pairs, and Hoogsteen base pairs, where nucleotides can bind with an alternative geometry that can drive the formation of uncommon secondary structures such as triple-stranded helices or G-quadruplexes.

3.1.4 Chain polarity

As mentioned earlier, the phosphodiester bond links the C5' atom of a nucleotide with the C3' atom of the one the follows it. As a result, DNA and RNA strands have a polarity, *i.e.* they are inherently directional. By convention, a strand starts at the $C5' - OH$ group (termed 5' end or terminal) and ends at the $C3' - OH$ group (the 3' end or terminal), and its sequence is also specified in this way.

[1] Here I simplify, since chemical modifications are possible (*e.g.* methylation), and sometimes U and T can end up in DNA and RNA, respectively.

[2] and also U, A and C can bind to the non-standard nucleotide hypoxanthine.

This is important since, as we will discuss in a moment, only anti-parallel strands (or anti-parallel sections of the same strand) can pair and form secondary structures.

3.1.5 Hydrophobicity and hydrophilicity

The nitrogenous bases have both hydrophobic and hydrophilic regions: the aromatic ring structures have hydrophobic properties, but they also contain functional groups that can form hydrogen bonds, contributing to some degree of hydrophilicity. However, the partial hydrophobic character of the bases is more than counterbalanced by the backbone, which is highly hydrophilic due to the negatively charged phosphate groups and to the hydroxyl groups ($-OH$) of the sugar that can form hydrogen bonds with water. As a result, nucleotides readily dissolve in water.

The strong hydrophilic character of DNA and RNA strands makes their tertiary structure somewhat simpler compared to proteins, since once the Section 3.2 are formed, there is no strong interaction driving the formation of tightly packed super-secondary or tertiary structures[3].

3.1.6 Hybridisation and denaturation

Two nucleotides coming together and binding to each other form a base pair (BP). Since hydrogen bonds are Section 2.3.2, HB formation requires that the two nucleotides approach each other with the right mutual orientation. However, note that bases on their own (*i.e.* not part of a strand) can pair with any other base, as well as with themselves, often with more than one arrangement (Voet and Rich [1970]). It is only in an extended paired region that the correct orientation is obtained only if the two nucleotides are compatible (*i.e.* if they can form a canonical or a non-canonical base pair, as mentioned above), and the strands run anti-parallel to each other. In this geometry, which is adopted by the most common DNA and RNA helical conformations,

- the bases on one strand can align perfectly with their complementary bases on the opposite strand, allowing stable hydrogen bonding;
- the sugar-phosphate backbones run in opposite directions, allowing the bases to stack neatly on top of each other, contributing to the overall stability of the double helix through van der Waals forces and hydrophobic interactions;
- negatively charged phosphate groups on the sugar-phosphate backbone are positioned in a way that minimizes repulsion.

Interestingly the spontaneous process through which single strands pair and form a duplex, which is called *hybridisation*, is now taken for granted, but was initially met with skepticism[4], since researchers at the time believed that duplex formation required enzymes to overcome electrostatic and entropic penalties (Rich [2009]).

In addition to hydrogen bonding, duplexes are stabilised by interactions acting between the aromatic rings of consecutive nitrogenous bases (see Figure 3.3). These so-called base stacking interactions have a hydrophobic nature that is complemented by van der Waals and dipole-dipole termsthat contribute significantly to the stability and structural integrity of the DNA molecule. Rather counter-intuitively, there is now ample evidence that base stacking is at least as important for duplex stability as base pairing, if not more (see *e.g.* Yakovchuk [2006], Vologodskii and Frank-Kamenetskii [2018], and Banerjee et al. [2023]).

Measuring the stacking strength

The classic study of Protozanova et al. [2004] used a clever experimental setup based on the observation that a duplex where one of the composing strands is broken in two (*i.e.* it is *nicked*) has a mobility that is lower compared to an intact one. The reason is that a nicked strand can take two conformations: one straight, where all the stacking interactions that resembles that of

[3] Of course, this argument applies to solutions of nucleic acids only. In the cell, RNA, DNA, and proteins can and do interact together to form higher order structures.

[4] "You mean [that a double helix can form] without an enzyme?" Rich [2009]

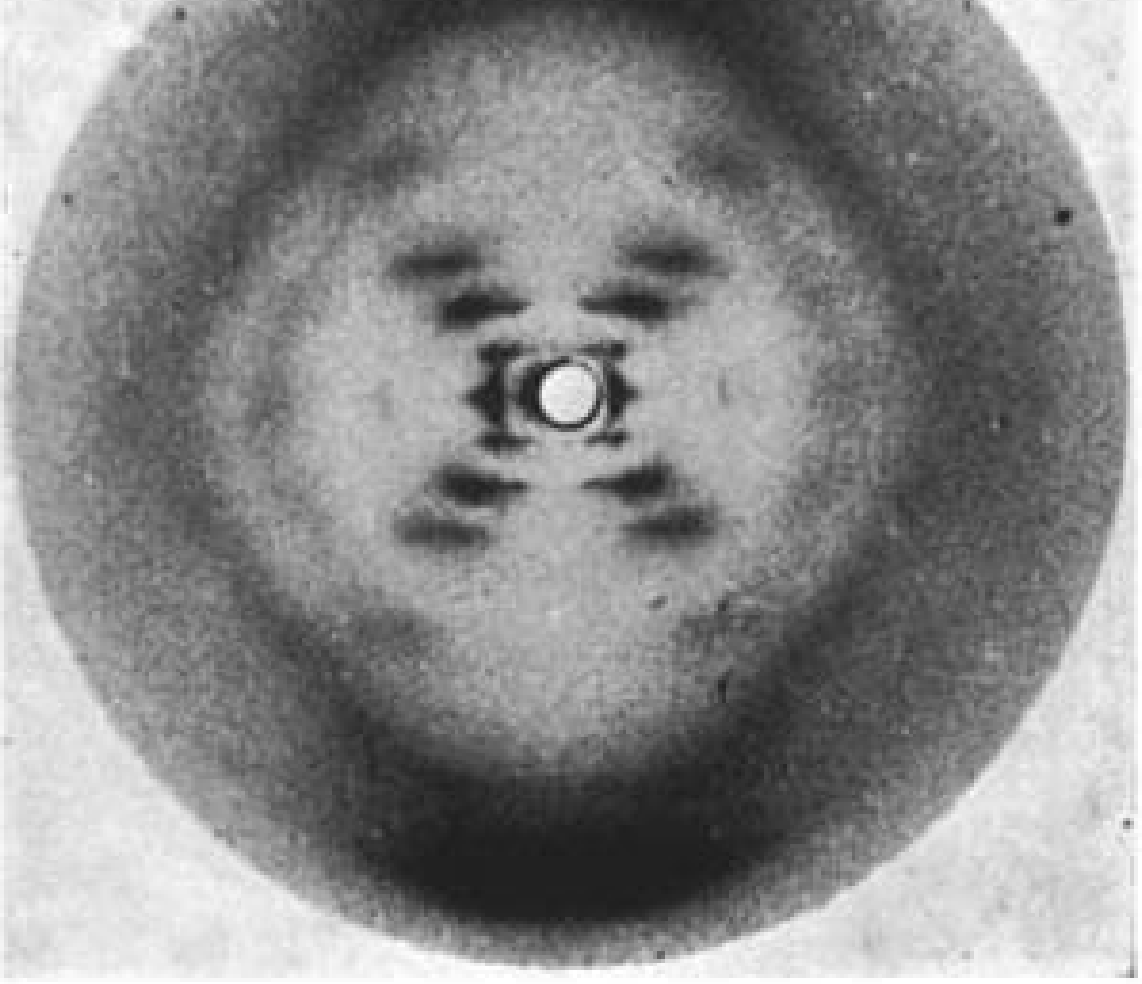

Figure 3.2: The famous "Photo 51", taken 1952 by Raymond Gosling during his work with Rosalind Franklin. Credits to Raymond Gosling/King's College London.

an intact duplex are present, and one kinked, where all base pairs are formed but the stacking interaction at the nicked site is broken.

Assuming that

1. the stacking strength between two consecutive bases depends only weakly on whether the two bases are part of the same strand or not;
2. the nicked duplex is in the straight conformation with probability p_s and it is kinked with probability $p_k = 1 - p_s$;
3. the mobility of the straight and kinked conformations are μ_s and μ_k, respectively;

it is possible to write an expression for the mobility of a nicked duplex μ as a weighted average over the two conformations:

$$\mu = p_s \mu_s + p_k \mu_k, \tag{3.1}$$

which can be recast as

$$\frac{p_k}{p_s} = \frac{\mu - \mu_s}{\mu_k - \mu}. \tag{3.2}$$

Therefore, if μ_s and μ_k are known, a measure of μ can be linked to an estimate of the ratio between the equilibrium populations of the two states. However, in equilibrium the latter is connected to the free-energy difference between the two states, ΔG^{ST} through the relation

$$\frac{p_k}{p_s} = \exp\left(-\frac{\Delta G^{\mathrm{ST}}}{k_B T}\right). \tag{3.3}$$

By changing the sequence of the nucleotides that flank the nicking site it is possible to obtain the values of ΔG^{ST} for each dinucleotide step.

The same technique has been used to understand how the stacking parameters depend on temperature and ionic strength in Yakovchuk [2006].

The hybridisation of complementary strands is an enthalpy-driven process, and therefore it is promoted by a temperature decrease: at high temperature, where entropy dominates, the two strands are separated. As T decreases, the energetic gain of forming BPs progressively leads to the stabilisation of the duplex. Strand hybridisation is a cooperative transition characterised by melting temperature and width that are controlled by the sequence, as well as by the buffer conditions (pH, ionic strength, *etc.*).

The transition is fully reversible: a duplex can be separated into its two composing strands (*i.e.* melted or denatured) by raising the temperature sufficiently.

3.2 Secondary structure

Hydrogen bonding and base stacking drives are the main drivers for the formation of secondary structure in nucleic acids which, akin to the secondary structure of proteins, refers to the local conformation taken by polynucleotide chains. In the biological setting there is a marked difference between the possible secondary structures of DNA and RNA. Indeed, the secondary structure of biological DNA is rather boring, as it tends to form long and stable double-stranded helices storing the genetic code. By contrast, the secondary structure of biological RNA is more diverse, owing to the additional hydroxyl group present in the sugar, and to the fact that RNA is nearly always found as a single strand. Its complex secondary structure allows RNA to fulfill its many roles in processes such as catalysis, regulation, and protein synthesis.

Note

The last two decades has seen the steady rise of DNA nanotechnology, which uses synthetic DNA to mimic, at least to some degree, the complexity of RNA secondary structure in order to exploit it for applications.

Assuming that each base can either be unbound or involved in a single base pair[5], the secondary structure of nucleic acids is defined by the pairing information of the nucleotides. By assigning a unique index to each nucleotide, this information can be represented as a list of index pairs, where each entry represents a single base pair. While, as I mentioned already, there are key differences between DNA and RNA, the surge of DNA nanotechnology has blurred the boundaries between the two, and many of the secondary structures that were unique to RNA can also be found in synthetic DNA systems. As a result, in this part I will refer to generic "nucleic acids", if not explicitly stated otherwise.

Figure 3.4 shows the secondary structure of an RNA molecule drawn as a graph, which is a common representation. The backbone runs from the 5' to the 3' end and it is drawn with grey sticks representing the graph's edges, while the single nucleotides are the vertices, drawn as circles. Base pairs are depicted with red connections. The RNA molecule has been designed to feature the main secondary structure motifs, which are highlighted by coloured dashed rectangles. These are:

- **Double-stranded part**: Sections made of consecutive base pairs.
- **Hairpin** (also known as a *stem-loop* structure): A single strand that folds back on itself to form a stem-loop structure. The stem is a double-stranded region where bases pair with complementary bases, and the loop is a single-stranded region at the tip.

[5]This is already an approximation, since some non-canonical (*e.g.* Hoogsteen) base pairings can connect a nucleotide to another which is already involved in a base pair.

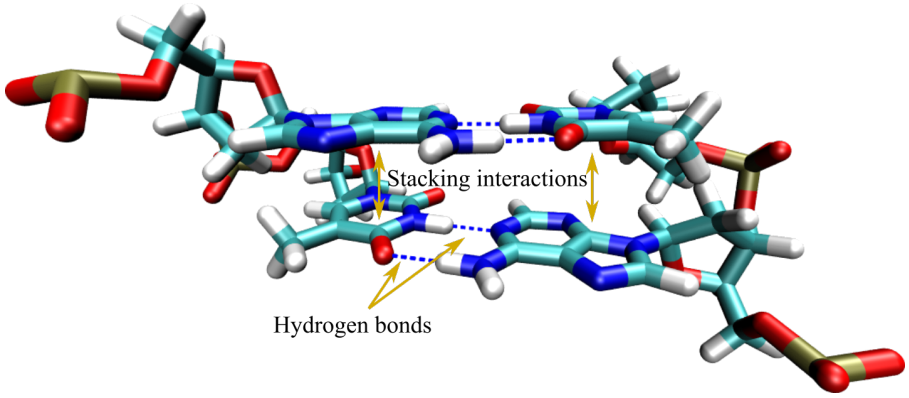

Figure 3.3: A base pair step where hydrogen bonds and stacking interactions, the two main mechanisms responsible for hybridisation, are highlighted by arrows.

- **Bulge**: Occurs when one or more unpaired nucleotides bulge out from one side of a double-stranded stem.
- **Internal loop**: Formed when there are unpaired nucleotides on both sides of the double-stranded stem, creating a loop in the middle of the stem. An internal loop made of one nucleotide on each strand is often called a **mismatch**.
- **Multibranched loop**: A loop where three or more double-stranded stems converge.
- **Pseudoknot**: Formed when bases in a loop pair with complementary bases that are separated by at least one double-stranded section. This results in a crossover of strands, forming a knot-like structure. A common way a pseudoknot forms is when a single-stranded loop of a hairpin pairs with a complementary sequence outside the hairpin.

The structure of a biological RNA molecule

Transfer RNA (tRNA) is a type of short (76 to 90 nucleotides) RNA molecule that plays a crucial role in the process of translating genetic information from messenger RNA (mRNA) into proteins. tRNA molecules function as adaptors that match specific amino acids to corresponding codons in the mRNA sequence during protein synthesis.

In particular, the primary function of tRNA is to translate the genetic code from mRNA into a sequence of amino acids, ultimately forming a protein. Each tRNA molecule is "loaded" with its corresponding amino acid by an enzyme called aminoacyl-tRNA synthetase[6], forming an aminoacyl-tRNA complex. The anticodon of the tRNA pairs with the complementary codon on the mRNA strand in the ribosome, whose machinery catalyses the formation of the peptide bond through which the amino acid carried by the tRNA is added to the growing polypeptide chain.

Figure **??** presents the secondary and three-dimensional structure of a tRNA molecule, highlighting its cloverleaf structure. It is worth mentioning that, as shown in the figure, many of the nucleotides of a tRNA molecule are chemically modified in order to tune their interaction with the ribosome or with the mRNA codons. The four main parts of a tRNA molecule are:

1. **The Acceptor Stem:** This is the site where a specific amino acid is attached. The 3' end of the tRNA has a conserved sequence (CCA) where the amino acid binds.
2. **The Anticodon loop:** This contains a sequence of three nucleotides (the anticodon) that is complementary to a specific mRNA codon, ensuring the correct amino acid is added to the growing polypeptide chain.
3. **The D loop:** Named for the presence of dihydrouridine, this arm is involved in tRNA folding.
4. **The TψC loop:** Named for the conserved sequence of thymine, pseudouridine, and cytosine, this arm is important for tRNA recognition by the ribosome.

3.2.1 Dot-paren notation

Dot-paren notation provides a clear and concise way to represent the secondary structure of nucleic acids, showing which nucleotides are paired or unpaired, regardless of their identity. It is a useful tool for storing information about complex secondary structures, and for exchanging this information to and from computational tools.

> **Warning**
> Dot-paren notation is primarily designed for representing the secondary structure of single-stranded nucleic acids, where it is very effective in illustrating the pattern of base pairing. However, for multi-strand systems, the dot-paren notation can become limited and cumbersome.

The dot-paren notation uses dots and parentheses to indicate unpaired and paired nucleotides, respectively.

1. **Unpaired Nucleotides:**
   - Dots (.) represent nucleotides that are not involved in base pairing.
   - Example: `....` represents four unpaired nucleotides.
2. **Paired Nucleotides:**
   - Parentheses indicate base pairs, with matching opening and closing parentheses representing the paired nucleotides.
   - An opening parenthesis `(` signifies the 5' end of a base pair, while a closing parenthesis `)` signifies the 3' end.
   - A well-formed dot-paren representation should have the same number of opening and closing parentheses.
3. **Hairpins:**
   - Hairpin loops are represented with a series of dots inside parentheses.
   - Example: `(((....)))` indicates a stem of three base pairs with a loop of four unpaired nucleotides.
4. **Bulges and Internal Loops:**
   - Bulges are unpaired nucleotides on one side of a stem.
   - Internal loops have unpaired nucleotides on both sides.
   - Example: `((..((...))..))` represents a structure where there is a small internal loop of size 2 within a larger stem, while `((.(...)))` represents a stem-loop with a single bulge.
5. **Multibranch Loops and Pseudoknots:**
   - Multibranch loops involve multiple stems and are, in general, complex to represent.
   - Pseudoknots, which involve base pairs crossing one another, are not well-represented by simple dot-paren notation but can be indicated with additional notation if necessary (see the complex example in the Hint **??**).

> **Two simple examples**
> Consider the following RNA sequence, which forms a hairpin with stem and loop of size 4:
>
> `5' - GGCAUCGUUGCC -3'`
>
> The dot-paren notation for this structure is:
>
> `((((....))))`

If a nucleotide is added in third position the sequence will also feature a bulge. An example of such a sequence and its secondary structure expressed with dot-paren notation would be

```
5' - GGCCAUCGUUGCC -3'
    ((.((....))))
```

A complex example: the input used to generate Figure 3.4
Figure 3.4 has been generated using the following prompt, where the first line is the sequence and the second line is the dot-paren representation of the associated secondary structure:

```
CGCUUCAUAAUGCACAUCCUCAAGCUGAUAGUGUGCUUGGGAAUGUCUGCACCAAGAGCCUUAAACUCUUGUGAUUAUGAAGUG
...((((((.(((...((((.((((.....[[[.)))))))).]]]..))).((((((.......))))))....)))))).
```

Note the square brackets: this is a (non-standard but common) way of representing pseudoknots.

There are other formats to store nucleic acid secondary structures that overcome some of the problems with the dot-paren notation (pseudoknot representation, lack of multi-strand support, poor readability, *etc.*), but none has become a standard yet. You can look at some examples here or here.

3.2.2 Nearest-neighbour models

Nearest-neighbour (NN) models for nucleic acids are computational frameworks used to predict the thermodynamic stability and secondary structure of DNA and RNA molecules. These models operate on the principle that the stability of a base pair is influenced primarily by its immediate neighbors rather than by more distant sequences. As we Section 3.1.6, base stacking and hydrogen bonding are the main drivers for the formation of helical structures. Therefore, assuming that the stability of nucleic acid structures is determined primarily by the local sequence context (the identity and orientation of adjacent base pairs), the complex interactions within nucleic acids can be simplified by considering only the contributions of dinucleotide pairs, hence the "nearest-neighbour" name.

The most used NN models are:

- The SantaLucia model for DNA (SantaLucia and Hicks [2004])
- The Turner model for RNA (Lu et al. [2006])

In addition, here is a useful website where several NN models are briefly described, and their parameters can be downloaded.

In general, a NN model is defined by a list of contributions that make it possible to assign a free-energy cost to the secondary structure of a specific strand (or system of strands) in an additive way: the total free-energy cost of the structure is given by a sum of terms that refer to the sequence and type of each local secondary structure. The specific values that enter into the calculations are being constantly improved upon by means of careful experiments on many different sequences (similar in spirit to those performed to obtain the sequence-dependent stacking strength, see Hint **??**). A recent example is Zuber et al. [2022]). By contrast, the functional forms and the nature of the different free-energy terms are rather stable and did not change much in the last 20+ years.

The contributions are given in terms of ΔH° and ΔS° or ΔG° and ΔH°, which are linked by the relation

$$\Delta G^\circ = \Delta H^\circ - T\Delta S^\circ. \tag{3.4}$$

where the ř superscript signals that these values refer to the free-energy differences estimated at the "standard" strand concentration of 1 molar (*i.e.* one mole per liter), C°. If the strand concentration C is different from C°[8], the final free-energy difference contains the additional entropic term

[8] In DNA nanotechnology the strand concentration is often of the order of 10^{-9} M.

$$\Delta S_C = -R \log C, \tag{3.5}$$

where $R \approx 2$ cal / mol K is the gas constant.

We now analyse the main free-energy contributions to the formation of secondary structures used in NN models.

Nearest-Neighbor Interactions

Double-strand formation is driven by the combined effects of base stacking and hydrogen bonding between adjacent base pairs. In NN models, these are accounted for by summing up the enthalpy and entropy contributions of each dinucleotide base steps. Therefore, for a sequence of length N there are $N - 1$ terms. As an example, consider the two fully-complementary strands

```
5' - TACCTG -3'
3' - ATGGAC -5'
```

In order to estimate the free-energy, we split the sequence into dinucleotide steps:

```
5' - TA AC CC CT TG -3'
3' - AT TG GG GA AC -5'
```

so that the total contributions due to this term are given by

$$\begin{aligned} \Delta H &= \Delta H_{\mathrm{TA/AT}} + \Delta H_{\mathrm{AC/TG}} + \Delta H_{\mathrm{CC/GG}} + \Delta H_{\mathrm{CT/GA}} + \Delta H_{\mathrm{TG/AC}} \\ \Delta S &= \Delta S_{\mathrm{TA/AT}} + \Delta S_{\mathrm{AC/TG}} + \Delta S_{\mathrm{CC/GG}} + \Delta S_{\mathrm{CT/GA}} + \Delta S_{\mathrm{TG/AC}} \end{aligned} \tag{3.6}$$

> **Warning**
> Remember that DNA and RNA strands have a **polarity**: the contributions due to CT/GA and TC/AC base steps differ! Always arrange the strands so that the top strand is listed in the 5' → 3' direction and the bottom strand in the 3' → 5' direction before splitting the sequence.

Terminal penalty

Most NN models requires adding a correction term to the free energy that depends on the base pair at each end of the helix. In most cases the penalty is present only if the terminal base pair is AT (or TA) and it tends to be rather small. Therefore, it is important only for short sequences.

Initiation and Symmetry Terms

The formation of a double helix requires an initial free-energy input to overcome entropy and start the pairing process. The term has to be added once per contiguous double-stranded region or helix.

In the presence of self-complementary duplexes (*i.e.* if the sequence of the two strands composing a duplex is palindromic), there is an additional purely entropic term that takes into account the fact that a strand can pair with a copy of itself. Here is an example:

```
5' - AGCGCT -3'
3' - TCGCGA -5'
```

Loop Free Energies

Forming hairpins, bulges, external, internal or multibranched loops costs both entropy (since we constrain the strand backbone) and enthalpy (since the backbone has to be bent or twisted to some extent). These free-energy penalties are, in general, sequence-dependent, but the number of possible combinations grows exponentially with the loop size. Therefore, the free-energy cost of these motifs is usually approximated to depend only on the loop size (with some exceptions such as hairpins with loops of length three and four, see *e.g.* SantaLucia and Hicks [2004]).

Penalties for Mismatches and Terminal Mismatches

Non-complementary paired bases have a destabilising effect on secondary structure. The specific energetic penalty that applies depend on the sequence of the mismatch, but also whether it occurs at the ends of helices (terminal mismatch), or within the helix (internal mismatches, sometimes referred to as "1x1 internal loops". Both reduce the overall stability of the structure, but to different extents. The following example contains both types of mismatches (look at the first and fifth base pairs):

```
5' - CTACACTG -3'
3' - TATGCGAC -5'
```

Dangling Ends

Dangling ends refer to unpaired nucleotides at the 5' or 3' ends of a helix that can stabilise specific secondary structures such as multibranched and exterior loops through additional stacking interactions. For instance, in this example the top strand has both a 5' and a 3' dangling end:

```
5' - ATACCTGC -3'
3' -  ATGGAC  -5'
```

These terms tend to be sequence-dependent: in the example above, the contribution of the 5' dangling end would be different if the sequence was TT/A or AC/A instead of AT/A. Note that a helix end extended on both strands has a Section 3.2.2 rather than two dangling ends.

Coaxial Stacking Parameters

Coaxial stacking refers to the stacking interactions between adjacent helices in branched or complex secondary structures, such as those found in multibranched loops or multi-strand systems. These interactions can significantly stabilize the overall structure by allowing helices to stack on top of each other in a manner similar to base stacking within a single helix.

In most NN models, two types of coaxial stacking are handled: when two helices are directly adjacent and no intervening unpaired nucleotides are present ("flush coaxial stacking"), or when a single mistmatch occurs between the stacked helices ("mismatch-mediated coaxial stacking"). The following examples (taken from here) show the two cases, where the backbone is explicitly drawn using dashes:

Salt-Dependent Terms

The stability of nucleic acid structures is influenced by the ionic environment, since cations like Na^+ and Mg^{2+} shield the negative charges on the phosphate backbone and reduce electrostatic repulsion between strands. Note that, as far as I know, only the SantaLucia NN model for DNA, presented in SantaLucia and Hicks [2004], takes into account this contribution through the following entropic term:

$$\Delta S_{\text{salt}} = 0.368 \frac{N_p}{2} \log C_S, \tag{3.7}$$

where N_p is the number of phosphates in the duplex, so that $N_p/2 = N$ is, under usual conditions, the duplex length, C_S is the molar concentration of monovalent cations[9], and the resulting contribution is in units of cal / mol K.

3.2.3 The two-state model

Duplex formation

One of the most straightforward applications of any NN model is to model the thermodynamics of duplex formation in a system composed of just two (perfectly or partially) complementary strands,

[9]Magnesium and other multivalent cations are not supported.

A and B. If the sequences are such that the possibility of stable intermediates can be neglected (that is, if the strands spend most of the time either free in solution or bound to each other), hybridisation can be described as a two-state process. The latter, in turn, is formally equivalent to the chemical equilibrium between two reactants and a product, *viz.*

$$A + B \rightleftharpoons AB. \tag{3.8}$$

The two-state model works particularly well for short strands (*i.e.* oligonucleotides) since, if they do not contain repeating patterns or similar "pathological" sequences, are more unlikely to exhibit metastable intermediates, *i.e.*, states with secondary structures whose stability can almost match that of the product.

Under the two-state assumption, equilibrium is described by the law of mass action equation

$$K_{AB} = \frac{C_A C_B}{C_{AB}}, \tag{3.9}$$

where C_X is the (molar) concentration of X and K_{AB} is the dissociation constant associated to the reaction, which is in turn connected to the free-energy difference between the two states, ΔG°_{AB}, through

$$K_{AB} = C^\circ \exp\left(\beta \Delta G^\circ_{AB}\right), \tag{3.10}$$

hence

$$\frac{C_A C_B}{C_{AB}} = C^\circ \exp\left(\beta \Delta G^\circ_{AB}\right). \tag{3.11}$$

The dissociation constant

The dissociation constant, often denoted as K_d, is a measure of the affinity between two molecules in a binding interaction, such as between between DNA strands in a duplex or a protein and its ligand. It is expressed in units of concentration (typically molarity, M), representing the concentration at which half of the strands are bound in duplexes, or half of the binding sites are occupied by the ligand. A lower K_d value indicates higher affinity, meaning the molecules are more likely to remain bound together at lower concentrations. Conversely, a higher K_d suggests weaker binding, indicating that higher concentrations are needed for significant binding to occur.

We define the concentrations of the two isolated strands (*i.e.* when $C_{AB} = 0$) as $C_{A,0}$ and $C_{B,0}$, so that the total strand concentration is $C_0 \equiv C_{A,0} + C_{B,0}$. Without loss of generality we assume that $C_{A,0} \leq C_{B,0}$.

Every time a duplex forms, the number of A and B strands decreases by one each, so that the total strand concentration can be written as $C_0 = C_A + C_B + 2C_{AB}$. We first compute the *melting temperature* T_m, which is the temperature at which half of the duplexes are formed. Under this condition, $C_{AB} = C_A$, since the number of duplexes that can form is controlled by the number of strands in the minority species, so that Eq. (3.11) becomes

$$\frac{C_B}{C^\circ} = \exp\left(\frac{\Delta G^\circ_{AB}}{R T_m}\right), \tag{3.12}$$

which, recalling that $\Delta G^\circ_{AB} = \Delta H^\circ_{AB} - T\Delta S^\circ_{AB}$, can be used to obtain the following expression for T_m:

$$T_m = \frac{\Delta H^\circ_{AB}}{\Delta S^\circ_{AB} + R \log\left(\frac{C_B}{C^\circ}\right)}. \tag{3.13}$$

However, since $C_B = C_0 - C_A - 2C_{AB} = C_0 - 3C_A = C_{A,0} + C_{B,0} - 3C_A$, and, by definition of melting temperature, $C_A = C_{A,0}/2$, so that $C_B = C_{B,0} - C_{A,0}/2$, we find

$$T_m = \frac{\Delta H^\circ_{AB}}{\Delta S^\circ_{AB} + R \log\left(\frac{2C_{B,0}-C_{A,0}}{2C^\circ}\right)}. \tag{3.14}$$

In the common case of equimolarity, *i.e.* when $C_{B,0} = C_{A,0} = C_0/2$, Eq. (3.14) simplifies to

$$T_m = \frac{\Delta H^\circ_{AB}}{\Delta S^\circ_{AB} + R \log\left(\frac{C_0}{4C^\circ}\right)}. \tag{3.15}$$

The C° factor

In most of the cases, the C° factor in Eq. (3.15) is omitted, since $C^\circ = 1$ M. However, as a physicist, you should always distrust equations where the arguments of mathematical functions such as log, exp, cos, *etc.* have physical dimensions. As a rule, you can disregard dimension issues only if you understand where they come from.

The denominator of Eq. (3.15) can be rewritten as $\Delta S^\circ_{AB} + R \log\left(\frac{C_0}{4C^\circ}\right) = \Delta S^\circ_{AB} + R \log\left(\frac{C_{0,A}}{2C^\circ}\right) = \Delta S_{AB} - R \log 2$, where

$$\Delta S_{AB} \equiv \Delta S^\circ_{AB} + R \log\left(\frac{C_{0,A}}{C^\circ}\right) \tag{3.16}$$

is a renormalised entropy difference that takes into account the concentration at which the reaction takes place. Using Eq. (3.16) makes it possible to directly derive the free-energy difference $\Delta G_{AB} = \Delta H^\circ_{AB} - T\Delta S_{AB}$ between the $A + B$ and AB states at any strand concentration. In the general case $C_{A,0} \leq C_{B,0}$, the additional entropic factor is $R \log\left(\frac{2C_{B,0}-C_{A,0}}{C^\circ}\right)$.

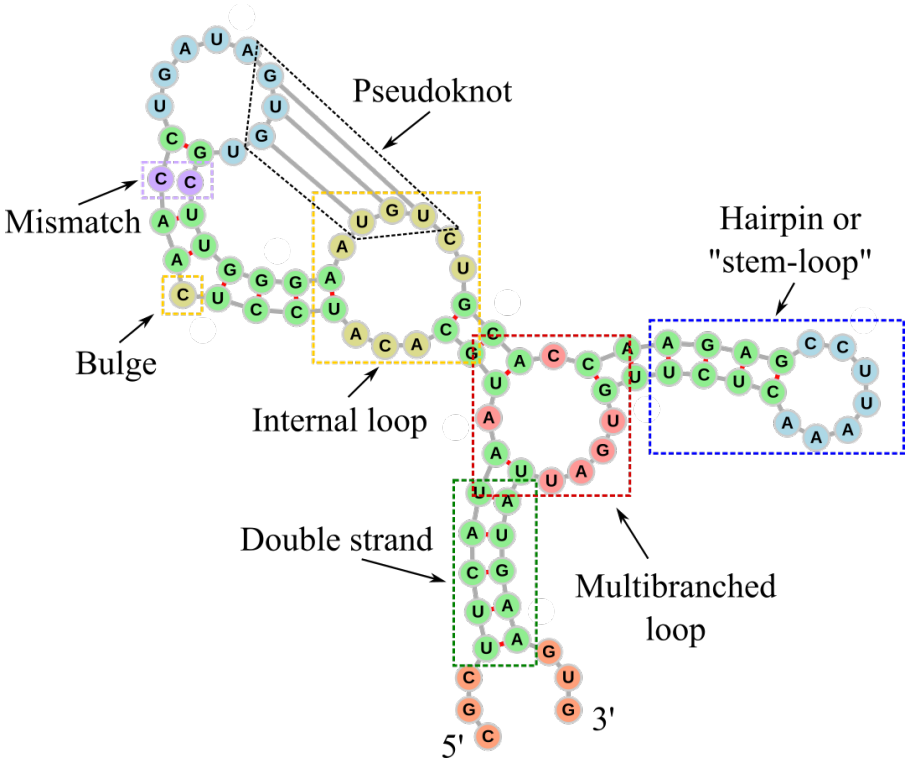

Figure 3.4: The secondary structure of an RNA molecule. Dashed rectangles and labels have been added to the different elements. The double-stranded parts are coloured in green, while the other colours are used to highlight the other main secondary structures: 5' and 3' unpaired regions in orange, internal loops and bulges in yellow, internal loops of size one (also known as *mismatches*) in violet, hairpin loops in blue, multibranched loops in red. The graph representation has been generated with forna.

Figure 3.6 shows a comparison between the experimental and theoretical melting temperatures of hundreds of DNA oligonucleotides predicted with the SantaLucia and Hicks [2004]. The average absolute deviation is smaller than 2.3 K.

Hairpin formation

Hairpin formation can also be modelled as a two-state process, where the equilibrium is between the open (random coil) and closed (stem-loop or hairpin) states:

$$\text{Coil} \rightleftharpoons \text{Hairpin}. \tag{3.17}$$

In this case the overall strand concentration does not play any role[10], and the dissociation equilibrium is given by

$$\frac{C_c}{C_h} = e^{\beta \Delta G^\circ_{ch}}, \tag{3.18}$$

where C_c and C_h are the concentrations of strands in the coil and hairpin conformations, respectively, and ΔG°_h is the free-energy difference between the two states. The condition for the melting temperature is $C_c = C_h$, which yields

$$T_m = \frac{\Delta H^\circ}{\Delta S^\circ}. \tag{3.19}$$

Melting curves

We now consider a generic nucleic acid system where one or more strands can pair and/or fold into a product P. We define the melting curve as the yield of P, in terms of concentration or fraction of formed product, as a function of temperature or another thermodynamic parameter that is changed experimentally, such as pH or salt concentration.

I will now show how the same two-state formalism introduced ealier can be used to predict the melting curve of a system. For the sake of simplicity I will use the $A + B \rightleftharpoons AB$ system with $C_{A,0} = C_{B,0}$. Defining X as the probability that a strand is *not* part of a duplex, the equilibrium concentrations can be written as $C_A = C_B = XC_{A,0}$ and $C_{AB} = (1 - X)C_{A,0}$. Substituting these relations in Eq. (3.11) and simplifying common factors we find

$$\frac{X^2}{1 - X} = \frac{C^\circ}{C_{A,0}} e^{\beta \Delta G^\circ_{AB}} = e^{\beta \Delta G_{AB}}, \tag{3.20}$$

where we have used the renormalised entropy to obtain the latter relation. Resolving for X and taking the positive root we find

$$X = \frac{-e^{\beta \Delta G_{AB}} + \sqrt{e^{2\beta \Delta G_{AB}} + 4e^{\beta \Delta G_{AB}}}}{2} = \frac{-1 + \sqrt{1 + 4e^{-\beta \Delta G_{AB}}}}{2e^{-\beta \Delta G_{AB}}}. \tag{3.21}$$

From X the duplex yield can be computed as $p_b = 1 - X$ (which is the probability that a strand is in a duplex), and the duplex concentration as $C_{AB} = p_b C_{A,0}$.

Figure 3.7 shows the yield of a oligonucleotide of 10 bp as predicted by the SantaLucia NN model and by simulations of two coarse-grained models.

3.3 Tertiary structure

The tertiary structure of nucleic acids is, in general, much simpler than that of proteins. This is due to the more limited variety of building blocks involved (4 *vs* 20), and to the charged (and, in general, hydrophilic) nature of the DNA and RNA backbones, which tend to destabilise the type of "super-secondary structures" that are so common in proteins. Moreover, the lack of tightly-packed structures

[10] If we can neglect the interaction between different strands, *i.e.* if the overall concentration is low enough that we can assume ideal gas behaviour.

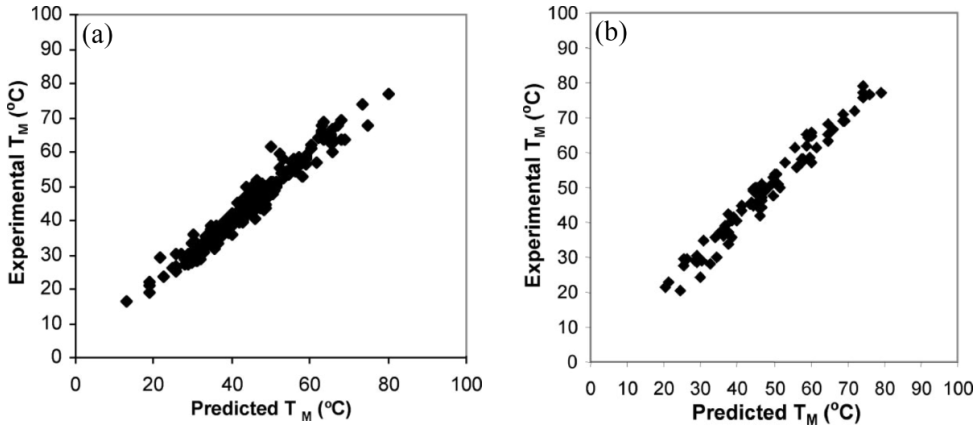

Figure 3.6: Experimental versus predicted melting temperatures of DNA oligonucleotides. (a) Data for 264 duplexes of length 4 to 16 bp in solution with 1 M of NaCl. (b) Data for 81 duplexes of length 6 to 24 bp in solution with variable NaCl concentration ranging from 0.01 to 0.5 M. Adapted from SantaLucia and Hicks [2004].

blurs the difference between secondary and tertiary structures, since most of the times, especially in simple systems, the tertiary structure is straightforwardly implied by the secondary structure.

In general, the most dominant tertiary structure of RNA and DNA is the double helix, and is the main one we will consider going forward. Many different helical structures have been discovered in biological contexts or synthesised artificially under specific conditions. However, here I will present only the three main ("canonical") ones. Note that they I will provide only an average characterisation for each helix, as it is known that their properties can depend (even strongly in some cases) on the local sequence. You can use this webserver (which is presented in Sharma et al. [2023]) to visualise the 3D structures of a given sequence, as estimated with a coarse-grained, realistic model.

3.3.1 Canonical helices

When two complementary strands pair together, they wrap around each other and form a periodic helical structure. The type of the resulting helix depends, in general, on the strand type (DNA or RNA), external conditions (ionic strength, pH, water concentration, *etc.*), and sequence.

B-DNA

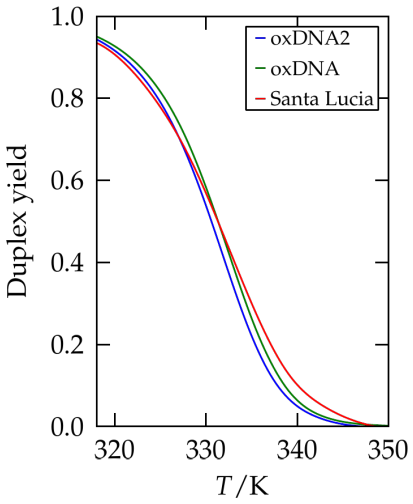

Figure 3.7: The yield of a 10-bp duplex as predicted by SantaLucia (red line), and two coarse-grained simulation models, oxDNA and oxDNA2. Adapted from Snodin et al. [2015].

The single most important conformation of DNA is B-DNA, which is the famous double helix whose

structure was described for the first time by WATSON and CRICK [1953]. B-DNA is a right-handed helical structure that consists of about 10.5 base pairs per turn, with a diameter of approximately 2 nanometers. The structure features major and minor grooves, which are essential for protein-DNA interactions; the major groove is wide and deep, while the minor groove is narrow and shallow. B-DNA is most stable under physiological conditions, making it the prevalent form in biological systems. As described in the box above, the famous experimental photo that was instrumental for Watson and Crick's discovery was taken by a student of Rosalind Franklin, who gave its name not only to B-DNA, but also to A-DNA, which is another canonical form of DNA.

Figure 3.8 shows a perfect B-DNA helix with different representations and from two different points of view. In panel (a) the representation makes it easy to see that

1. The strands run in an anti-parallel fashion: look at the orientation of facing pentagons, or at the way bases are connected to the sugars.

2. Base pairs have essentially zero inclination with respect to the helical axis.

A-DNA and A-RNA

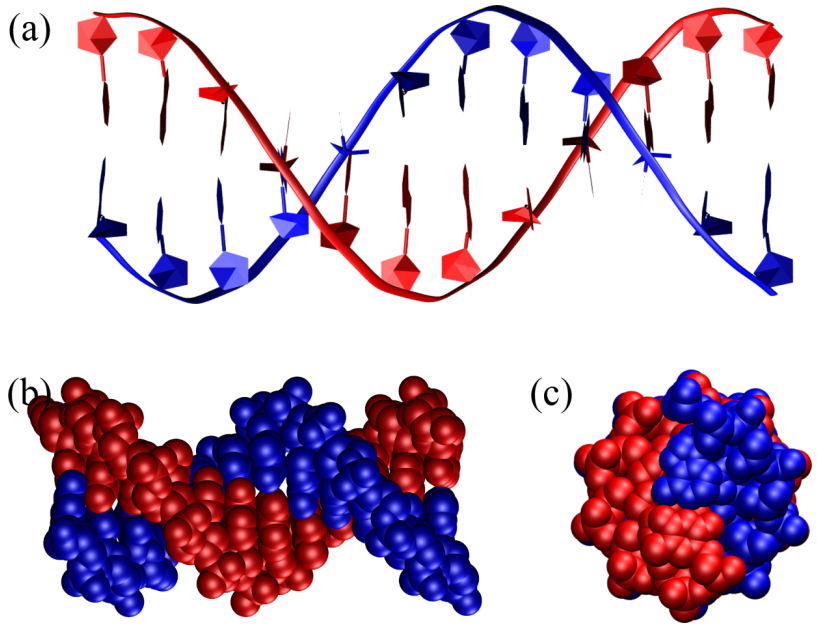

Figure 3.8: A B-DNA double helix composed of 12 base pairs as seen from the side, (a) and (b), and from the top (c). In (a) the backbone is represented as a ribbon, the sugar as a pentagon and the bases are outlined as pentagons and hexagons, while (b) and (c) shows the atoms as van der Waals spheres. The structure has been generated with the 3DNA 2.0 webserver.

DNA turns into its A form when dyhdrated, which can happen as a result of human intervention (*e.g.* when crystalling samples to study with X-rays), but also under natural conditions, when water is scarce or specific proteins bind to the DNA, in place of some of the water molecules[11]. A-DNA is a right-handed helix that is more compact than B-DNA: it has about 11 base pairs per turn and a diameter of approximately 2.3 nanometers. A-DNA features deeper major grooves and shallower minor grooves compared to B-DNA. The helical structure is more tightly wound, with the base pairs tilted by abouy 20° relative to the helix axis, and the distance between base pairs is shorter. Moreover, the sugars are in the C3'-endo conformation rather than in the C2'-endo conformation like in B-DNA. A-DNA is less prevalent in biological systems but can be found in certain DNA-RNA hybrids. Additionally, double-stranded RNA molecules or regions typically adopt the A-form helical structure.

[11] In both cases the A form is thought to protect the genetic material from extreme conditions.

Figure 3.9 shows a perfect A-DNA helix made of the same number of base pairs as the one shown in Figure 3.8. Compared to B-DNA, it is evident that A-DNA is shorter and thicker, and the base pairs are inclined by $\approx 20^{\circ}$ with respect to the helical axis.

Z-DNA

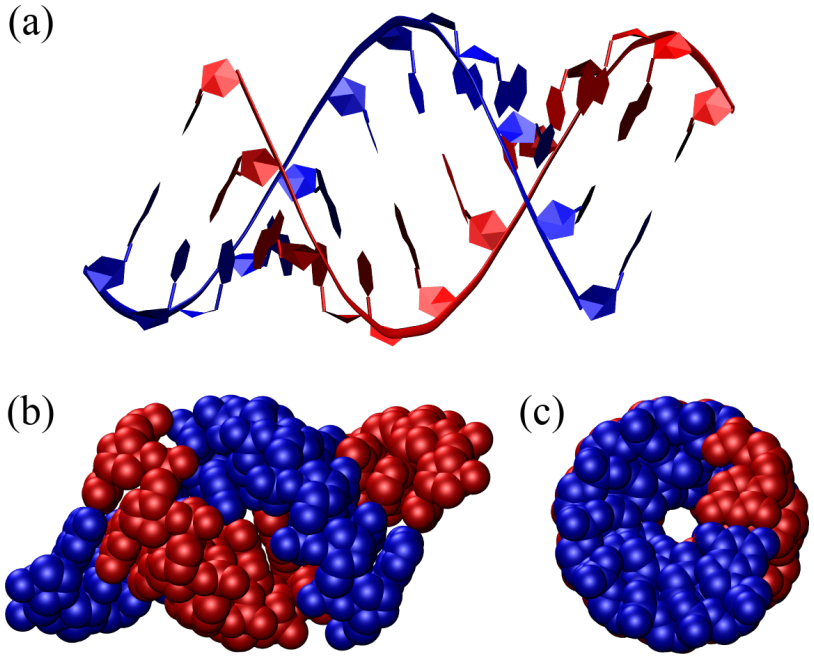

Figure 3.9: An A-DNA double helix composed of 12 base pairs represented as in Figure 3.8. The structure has been generated with the 3DNA 2.0 webserver.

Another important DNA conformation is Z-DNA, which is a less common and structurally distinct form of DNA characterized by a left-handed double helical structure, in contrast to the right-handed helices of A-DNA and B-DNA. It has a zigzag backbone, which gives Z-DNA its name, and consists of about 12 base pairs per turn with a diameter of approximately 1.8 nanometers. The major and minor grooves are less distinct in Z-DNA, and the structure is more elongated and thinner. Z-DNA requires a specific sequence of alternating purines and pyrimidines (*e.g.* repeating GC steps) and can form under high salt conditions or negative supercoiling, *i.e.* twisting of the DNA in the left-handed direction. Though less prevalent than B-DNA, Z-DNA is thought to have important biological functions in gene regulation, genetic recombination, and in the context of certain protein-DNA interactions.

Figure 3.10 shows a perfect Z-DNA helix made of 24 base pairs (12 repeating GC steps) which shows the main features of Z-DNA: the left handedness of the helix, the zig-zag pattern that gives this conformation its name, and the small BP inclination.

A comparison between the average properties of the three main helical conformations is shown in Table **??**.

3.3.2 Other tertiary motifs

Other important tertiary motifs are those that can connect more than two strands together, or two sections of the same strand that are far apart from each other. For instances, in triplexes (*i.e.* triple-stranded DNA or RNA) a third strand can bind in the major groove of a duplex through Hoogsteen base pairings, or in RNA in the minor groove by leveraging the presence of the additional hydroxyl group in the sugar.

Hoogsteen base pairings can also lead to the formation of quadruplexes, which can occur in a variety of patterns, including intramolecular (within a single strand), intermolecular (between different strands), and hybrid types. A strand (or multiple strands) with a high number of consecutive guanine bases folds back on itself (or aligns with other strands) to bring the G bases into proximity. Four

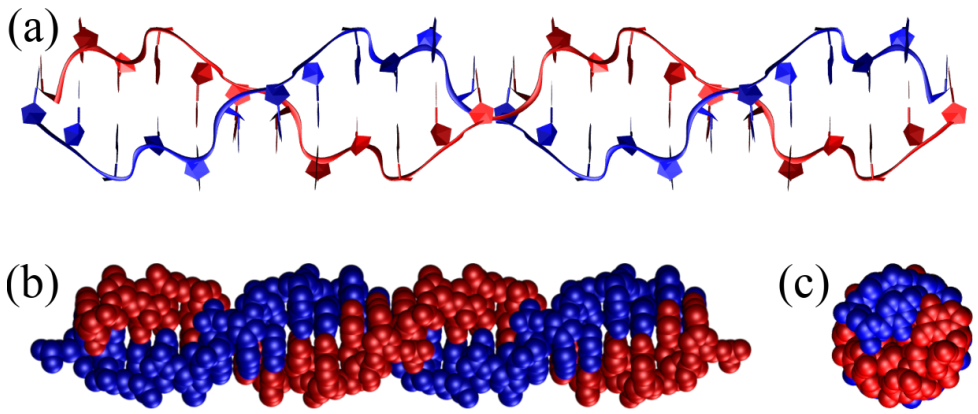

Figure 3.10: A Z-DNA double helix composed of 12 GC base steps (*i.e.* 24 base pairs) represented as in Figure 3.8. Note the zig-zag pattern drawn by the backbone, which gives Z-DNA its name. The structure has been generated with the 3DNA 2.0 webserver.

guanine bases form a planar structure known as a G-tetrad through Hoogsteen hydrogen bonding, and multiple G-tetrads stack on top of each other, stabilized by the $\pi - \pi$interactions between the aromatic rings of the guanine bases.

Finally, as mentioned earlier, pseudoknots and coaxial stacking interactions can also seen as mechanisms the can lead to the formation of particular tertiary structures, since they can stabilise multi-strand (or multi-loop) structures.

Chapter 4

Structure and folding prediction

4.1 Proteins

Tip
The main references for this part are Finkelstein and Ptitsyn [2002] and Bialek [2012]. Note that here I refer mostly to globular proteins.

Let's consider a 100-residue peptide chain with a unique folded state. If each amino acid had only two available conformations, the number of configurations available to the chain would be $2^{100} \sim 10^{30}$. If the time required to switch between any two configurations was 10^{-12} s (a picosecond), and we assume that no configuration is visited twice, it would take approximately 10^{18} seconds to explore all the "phase space" (and therefore, on average, to fold): this is close to the age of the universe! In reality, proteins fold on time scales ranging from microseconds to hours. This is the gist of the famous "Levinthal's paradox", which implies that the search for the folded structure does not happen randomly, but it is guided by the energy surface of residue-residue interactions.

Proteins can be denatured by changing the solution conditions. Very high or low temperature and pH, or high concentration of a denaturant (like urea or guanine dihydrochloride) can "unfold" a protein, which loses its solid-like, native structure, as well as its ability to perform its biological function. Investigations of the unfolding of small globular proteins showed that, as the denaturing agent (*e.g.* temperature or denaturant concentration) increases, many of the properties of the protein go through an "S-shaped" change, which is characteristic of cooperative transitions.

Furthermore, calorimetric studies show that the denaturation transition is an "all-or-none" transition, *i.e.* that the "melting unit" of the transition is the protein as a whole, and not some subparts. This of course applies to single-domain small proteins, or to the single domains of larger proteins. Here "all-or-none" means that the protein exist only in one of two states (native or denaturated), with all the other "intermediate" states being essentially unpopulated at equilibrium. Therefore, this transition is the microscopic equivalent of a first-order phase transition (*e.g.* boiling or melting) occurring in a macroscopic system. Of course, since proteins are finite systems and therefore very far from the thermodynamic limit, this is not a true phase transition, as the energy jump is continuous, and the transition width is finite.

The van't Hoff criterion
Let N and D be two states in chemical equilibrium, *i.e.* $N \rightleftharpoons D$, at a certain temperature T. Recalling the two-state equilibrium concepts derived Section 3.2.3, and defining $p(T)$ as the fraction of, say, state N, we have

$$K_{\text{eff}}(T) = \frac{p(T)}{1 - p(T)} = e^{\beta \Delta G_{ND}}, \tag{4.1}$$

where $K_{\text{eff}}(T)$ is the effective dissociation constant at temperature T and $\Delta G_{ND} = \Delta H_{ND} -$

$T\Delta S_{ND}$ is the free-energy differences between the two states. By simple algebra we have that

$$\Delta H_{ND} = -k_B T^2 \frac{d \log K_{\text{eff}}(T)}{dT}. \tag{4.2}$$

Note that $K_{\text{eff}}(T)$ is connected to $p(T)$, which can be measured experimentally (for instance with Circular Dicroism). The value of ΔH_{ND} obtained can be interpreted as the heat consumed by the "melting unit" associated to the transition.

The enthalpy change can also be estimated by integrating the heat capacity, measured with calorimetry, over the range of temperature associated to the transition. This is the heat consumed by all proteins during the transition, $\Delta H_{\text{tot}} = N_p \Delta H_{\text{cal}}$, where $N_p = m/M$ is the number of proteins, with m being the total mass and M the protein's molecular mass.

If $\Delta H_{\text{cal}} = \Delta H_{ND}$, it means that the "melting unit" is the whole protein, and therefore the transition is of the "all-or-none" type, as it is the case of small globular proteins.

Note that there are Zhou et al. [1999] that should be taken into account when applying this method, but the general concept is still useful, provided that the results are confirmed by independent experimental measurements.

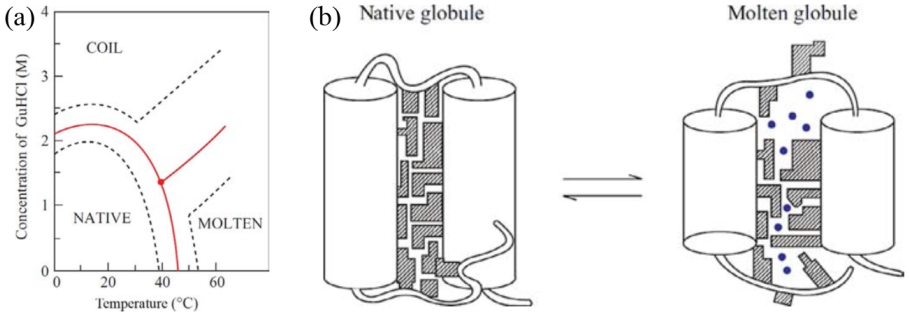

Figure 4.1: (a) Phase diagram of the conformational states of lysozyme at pH 1.7 as a function of denaturant (guanine dihydrochloride) concentration and temperature. The red lines correspond to the mid transition, the dashed lines outline the transition zones. (b) Schematic model of the native and molten globule protein states. Here the protein consists of only two helices connected by a loop, and the side chains are shown as shaded regions. Adapted from Finkelstein and Ptitsyn [2002].

But how does the denatured state look like? As discovered by using a plethora of experimental methods, which often seemed contradicting each other, the answer depends on the denaturing conditions. Figure 4.1(a) shows the phase diagram of lysozyme (a globular small protein) for different temperatures and concentrations of a denaturing agent. Increasing the latter leads to a transition to a disordered coil state where essentially no secondary structure is present. By contrast, the effect of temperature is qualitatively different, as the protein partially melts in a state (the *molten globule*, MG) that retains most of its secondary structure, but it is not solid like and it cannot perform any biological function. A schematic of the difference between the native and molten globule states is shown in Figure 4.1(b).

The cooperative transition that leads to the molten globule is due to the high packing of the side chains, which is energetically favourable but entropically penalised, since it impedes the small-scale backbone fluctuations that trap many of the internal degrees of freedom of the protein. The liberation of these small-scale fluctations, and the resulting entropy gain, requires a slight degree of swelling, which is why the MG is not much larger than the native state. However, such a swelling is large enough to greatly decrease the van der Waals attractions, which are strongly dependent on the distance, and to let solvent (water) molecules in the core.

In general, not all proteins behave like this, as there exist some (usually small) proteins that unfold directly into a coil, others that form the molten globule under the effect of specific denaturants, and coils under the effect of others, *etc.* However, the unfolding transition has nearly always an "all-or-none", cooperative nature.

Coil-globule transition in polymers
The cooperative denaturing of a protein is *very* different from the coil-globule transition observed in polymers, even when no molten globule is involved and the protein turns directly into a coil. The reason is that the coil-globule transition is not a microscopic analog of a first-order phase transition, and cannot be described by two-state models.

Generally speaking, the denaturation of a protein is a reversible process, provided that some experimental conditions are met (the protein is not too large, its chemical structure has not been altered by the denaturation process or by post-translational modifications, *etc.*). The fact that protein folding and unfolding is a reversible process implies that all the information required to build a protein is stored in its sequence, and that its native structure is thermodynamically stable. Therefore, the folding process can be, in principle, understood by using thermodynamics and statistical mechanics. The sequence → structure hypothesis is sometimes called Anfisen's dogma, named after the Nobel Prize laureate Christian B. Anfisen, whose studies on refolding of ribonuclease were key to establish the link between "the amino acid sequence and the biologically active conformation" (Anfinsen et al. [1961], Anfinsen [1973]).

Warning
There are many "exceptions" to this dogma, mostly because large proteins need help to fold, because some native structures require post-translational modifications, and because *in vitro* renaturation is difficult and can be hindered by inter-molecular or (for large proteins) even intramolecular aggregation. Nevertheless, thanks to the validity of this "thermodynamic hypothesis", many common features of denaturation and folding can be understood by means of rather simple arguments which, as I will show, can be used to build simple models.

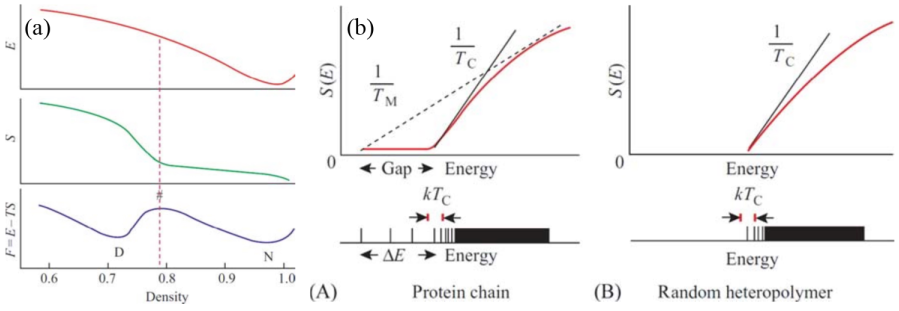

Figure 4.2: (a) Typical curves for the energy, entropy and free-energy of a protein as functions of the (intramolecular) density. The D and N letters mark the positions of the denatured and native states, respectively. The hash pound marks the position of the maximum of the free-energy barrier. (b) A sketch of the energy spectra (bottom panels) and associated $S(E)$ curves (top panels) for (A) a protein and (B) a random heteropolymer. Adapted from Finkelstein and Ptitsyn [2002].

Figure 4.2(a) shows the origin of the "all-or-none" transition observed in protein denaturation in thermodynamic terms. Here I will use the description given by Finkelstein and Ptitsyn [2002] (whence the figure comes), almost word by word. The panel shows the energy E and the entropy S of a protein as a function of its internal density (in some arbitrary units). The energy is at its minimum at close packing, while the entropy S increases with decreasing density. Initially, this increase is slow, since the side chains can only vibrate, but their rotational isomerization is impossible. When the density decreases enough that the latter becomes possible the increase is much steeper. Finally, when free isomerization has been reached the entropy growth becomes slow again.

The nonuniform growth of the entropy is due to the following. The isomerization needs some vacant volume near each side group, and this volume must be at least as large as a CH_3 group. And, since the side chain is attached to the rigid backbone, this vacant volume cannot appear near one side chain only but has to appear near many side chains at once. Therefore, there exists a threshold (barrier) value of the globule's density after which the entropy starts to grow rapidly with increasing volume (decreasing density) of the globule. This density is rather high, about 80% of the density of the native globule, since the volume of the CH_3 group amounts to $\approx 20\%$ of the volume of an amino acid residue. Therefore, the dependence of the resulting free energy $F = E - TS$ on the globule's density ρ has a maximum (the so-called free-energy barrier) separating the native, tightly packed globule (N) from the denatured state (D) of lower density. It is because of the presence of this free energy barrier that protein denaturation occurs as an "all-or-none" transition akin to melting or nucleation, independent of the final state of the denatured protein.

It is useful, when talking about proteins, to also introduce the concept of random heteropolymers, which are polymers with sequences that have not been selected by nature, and therefore have many (degenerate or quasi-degenerate) ground states rather than a single native conformation. Figure 4.2(b) shows the qualitative difference between a protein and a random heteropolymer. The bottom panels of the figure show the typical energy spectra of the two chains, where each line corresponds to a single conformation. At high energy both spectra are essentially continuous. However, the low-energy, discrete parts look very different: the energy difference between the ground state and the continuous part of the spectrum, where the majority of the "misfolded" states lie, is much larger in a protein than in a random heteropolymer. It is this energy gap that creates the free-energy barrier of Figure 4.2(a).

We can also define the entropy $S(E)$, which is proportional to the logarithm of the number of structures having energy E, and the temperature corresponding to the energy E, implicitly defined by the relation $dS(E)/dE = 1/T(E)$. As sketched in Figure 4.2(b), the peculiar shape of the protein energy spectrum gives raise to a concave part of $S(E)$, which means that there is a temperature at which the lowest-energy, native conformation coexists with many high-energy (denatured) ones. In this simple picture, this is the melting temperature T_M. This temperature should be compared with the so-called "glass temperature" T_G, called T_C in the figure, which is the temperature associated to the lowest energy of the continuous band of the spectrum. Above T_G the chain behaves, from the kinetic point of view, like a viscous liquid, while below the glass temperature the behaviour is glassy-like (*i.e.* non-Arrenhius dependence on temperature, aging, non ergodicity, *etc.*).

If $T_M < T_G$, the slow dynamics asssociated to the glass state makes it kinetically impossible to reach the ground state, and the chain remains trapped in higher-energy misfolded states. By contrast, if $T_M > T_G$, folding is possible, and results obtained with minimalistic models show that the fraction T_M/T_G is a good proxy for "foldability": the larger this fraction, the faster the protein folds.

All these concepts are not merely qualitative, but have been grounded in theory by using sophisticated statistical mechanics approaches that have generated the "funnel landscape" folding picture, which is schematically shown in Figure 4.3(b).

The basic idea is to leverage the concept of frustration, which happens when the interactions between the different parts of a system are such that they cannot be satisfied, from the energetic point of view, all at the same time. If this is the case, then there is no single ground state, but rather many low-energy stable states that, in a many-body system, are uncorrelated from each other and separated by (possibly large) energy barriers. An example is provided in Figure 4.3(a). The resulting "energy surface" (or landscape) is said to be rugged (or rough), and generates a dynamics where the system sits in a valley for a long time before being able to jump over the barrier and move to a different stable conformation. This is a hallmark of glassiness.

To draw a parallel with proteins, we can consider a random heteropolymer, where the interactions among the different amino acids will be frustrated, blocking the system from finding a single well isolated folded structure of minimum energy. A candidate principle for selecting functional sequences is thus the minimization of this frustration. Of course even in the absence of frustration, there are still energetic barriers on the path towards the native conformation due to local structural rearrangements which still give raise to a rugged landscape, slowing down the kinetics towards the folded state. This scenario has come to be called a folding "funnel", emphasizing that there is a single dominant valley

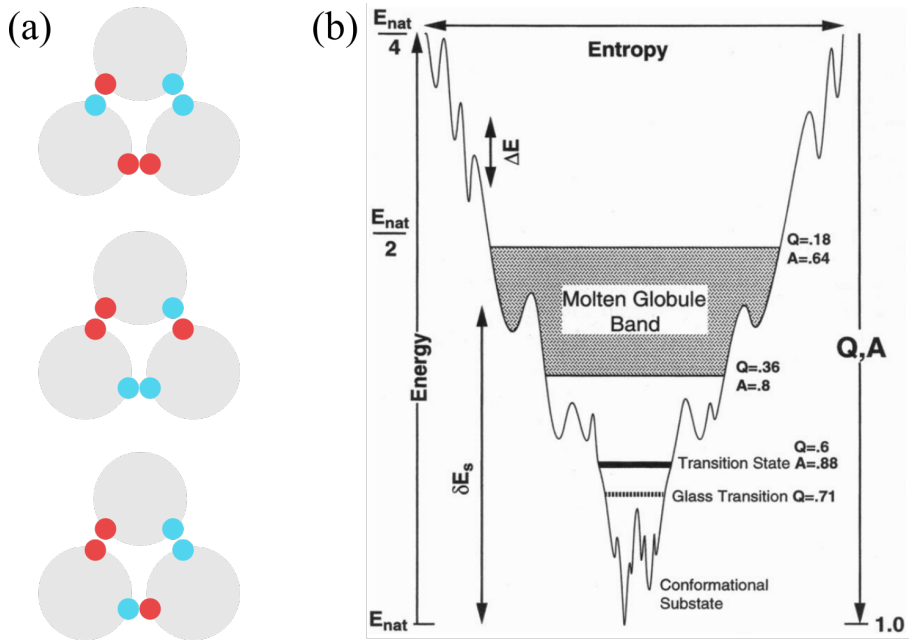

Figure 4.3: (a) An example of frustrated interactions: if the particle-particle interaction is such that only like colours want to be close to each other, there is no way of arranging the three spheres so that all favourable contacts are made. As a result, the ground state is degenerate. (b) A schematic energy landscape of a 60 amino-acid helical protein. A and Q are the fractions of correct dihedral angles in the backbone and correct native-like contacts, respectively, ΔE is the "ruggedness" of the landscape, and δE_s is the energy gap. Taken from Onuchic et al. [1995].

in the energy landscape, into which all initial configurations of the system will be drawn.

A classic funnel diagram, such as the one shown in Figure 4.3(b), represents an energy-entropy landscape, with the width being a measure of the entropy, whereas the depth represents both the energy and two correlated order parameters Q and A, which are the fractions of native contacts and correct dihedral angles in the protein backbone. Although in reality the landscape is multidimensional, here the projection attempts to retain its main features, such as the barrier heights, which are a measure of the "ruggedness" or "roughness" of the landscape. The typical height of these barriers, together with the energy gap and the number of available conformations, are the three main parameters of the Random Energy Model, which is one of the main theoretical tools in this context.

4.1.1 The Zimm-Bragg model for the helix-to-coil transition

As we discussed, polypeptides can, in general, be reversibly denatured (with temperature or pH, for instance). Here we theoretically investigate this phenomenon in the context of secondary structure with the Zimm and Bragg [1959], which is a toy model for a chain that can exhibit a continuous coil-to-helix transition. Note that it is possible to synthesise polypeptide chains that undergo such a transition, see *e.g.* Doty and Yang [1956].

Consider a polymer chain composed of N residues. Each residue can be in one of two states, "coil" (c) or "helix" (h), so that a microscopic state of the chain can be expressed as a one-dimensional sequence, like `ccchhhhhcchhhchcc`. In equilibrium, the probability (or statistical weight) to observe a given chain microstate c is just the Boltzmann factor, $e^{-\beta \Delta F_c}$, where ΔF_c is the free-energy cost of the microstate, so that the total (configurational) partition function is

$$Q = \sum_{c \in \text{config}} e^{-\beta \Delta F_c}. \tag{4.3}$$

Since (free) energies are always defined up to a constant, we define the free-energy of a coil of any length to be zero by definition, so that the statistical weight of a coil residue is always 1. By contrast, forming a sufficiently long helical segment should be advantageous, otherwise no transition

would occur. However, we know that in Section 2.6.1 formed by polypeptides, each j-th amino acid is hydrogen-bonded to the $(j+k)$-th, where $k=4$ for a α-helix. As a consequence, since one full helical turn needs to be immobilised before seeing any benefit from the formation of hydrogen bonds, the formation of a helix incurs a large entropic penalty that needs to be paid every time a coil segment turns into a helical one, f_{init}. However, once a helix is formed and the initial price has been paid[1], adding an H contributes a fixed amount f_h to the helix free energy. As a result, the free-energy contribution of a helical stretch of size n is $\Delta F = f_{\text{init}} + nf_h$, so that is statistical weight is

$$\exp(-\beta\Delta F) = \exp(-\beta f_{\text{init}})\left[\exp(-\beta f_h)\right]^n \equiv \sigma s^n, \tag{4.4}$$

where we have defined the cooperativity (or initiation) parameter, $\sigma \equiv \exp(-\beta f_{\text{init}})$, and the helix-elongation (or hydrogen-bonding) parameter, $s \equiv \exp(-\beta f_h)$. Note that, since f_{init} is purely entropic, $\beta f_{\text{init}} = -S_{\text{init}}/k_B$ and therefore σ does not depend on temperature.

Warning
Strictly speaking, it is not correct to add the nf_h term is $n < 4$, since at least one turn (≈ 4 amino acids) should be present in order to stabilise an α-helix. There are some more complicated models that take this into account (see *e.g.* the Lifson-Roig model or the Badasyan et al. [2013]), but here we will keep it simple and study the simplest model that exhibits the main features of the transition.

Following Finkelstein and Ptitsyn [2016], we define two accessory quantities: C_i is the partition function of a chain of size i, where the last monomer is in the coil state, and H_i is the partition function of a chain of size i, where the last monomer is in the helix state. With this definition, the partition function of the chain is $Q_i = C_i + H_i$. We can explicitly compute the first terms:

- For $i=1$, the sequence is either `c` or `h`, so that $C_1 = 1$ and $H_1 = \sigma s$.
- For $i=2$, we have two sequences ending with c, which are `hc` and `cc`, and two sequences ending with h, which are `hh` and `ch`. Each partition function is given by a sum of the statistical weights of the allowed microstates, so that we have $C_2 = 1 + \sigma s$ and $H_2 = \sigma s^2 + \sigma s$.
- For $i=3$, there are 8 available sequences, which are built by taking each $i=2$ sequence and adding either `c` or `h` at its end. Each of the resulting sequence will have a statistical weight that is that of the base $i=2$ sequence, multiplied by 1 if the sequence ends with `c`, and by s or σs if the sequence ends with `h` and the preceeding character is `h` or `c`, respectively. Summing up the contribution we obtain $C_3 = C_2 + H_2 = 1 + 2\sigma s + \sigma s^2$ and $H_3 = C_2\sigma s + H_2 s = \sigma s + \sigma s^2 + \sigma^2 s^2 + \sigma s^3$.

In general the operations carried out to compute the $i=3$ partial partition functions can be applied to any other value of i, meaning that we can write down recursive relationships connecting H_i and C_i to H_{i-1} and C_{i-1}. These relationships can be neatly expressed in matricial form:

$$(C_i, H_i) = (C_{i-1}, H_{i-1}) \begin{pmatrix} 1 & \sigma s \\ 1 & s \end{pmatrix}. \tag{4.5}$$

It is convenient to define the matrix

$$\mathbf{M} \equiv \begin{pmatrix} 1 & \sigma s \\ 1 & s \end{pmatrix} \tag{4.6}$$

so that the the total partition function of a chain of N monomers can be written as[2]

$$Q_N = (1,0)\mathbf{M}^N \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tag{4.7}$$

[1] In another context, that of first-order phase transition, we would say "once a nucleus forms".

[2] Note that in the original Zimm-Bragg model the first three segments are always in the coil state, so that the exponent is $N-3$ rather than N.

where the first vector-matrix multiplication on the right-hand side gives (C_N, H_N), which multiplied by the $(1,1)$ column vector yields $C_N + H_N$. The partition function can be written in terms of the eigenvalues of $\mathbf{M}$ if we note that

$$\mathbf{M}^N = \mathbf{M}\mathbf{M}\dots\mathbf{M} = \mathbf{A}\mathbf{\Lambda}\mathbf{A}^{-1}\mathbf{A}\mathbf{\Lambda}\mathbf{A}^{-1}\dots\mathbf{A}\mathbf{\Lambda}\mathbf{A}^{-1} = \mathbf{A}\mathbf{\Lambda}^N\mathbf{A}^{-1}, \tag{4.8}$$

where $\mathbf{\Lambda} = \mathbf{A}^{-1}\mathbf{M}\mathbf{A} = \begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}$, and λ_1 and λ_2 (with $\lambda_1 > \lambda_2$) are the eigenvalues of $\mathbf{M}$, which can be found by solving the secular equation, *viz.*:

$$\det\begin{pmatrix}1-\lambda & \sigma s\\ 1 & s-\lambda\end{pmatrix} = (1-\lambda)(s-\lambda) - \sigma s = 0, \tag{4.9}$$

whence

$$\lambda_1 = 1 + \frac{s-1}{2} + \sqrt{\left(\frac{s-1}{2}\right)^2 + \sigma s} \tag{4.10}$$

$$\lambda_2 = 1 + \frac{s-1}{2} - \sqrt{\left(\frac{s-1}{2}\right)^2 + \sigma s}. \tag{4.11}$$

Note that the two eigenvalues are linked by the relation $\lambda_1 + \lambda_2 = 1 + s$.

Since they can be of any length, the two eigenvectors $\vec{A}_1$ and $\vec{A}_2$ of a 2x2 matrix each have a single independent element, a_k (where $k = 1, 2$), which is given by

$$\begin{pmatrix}1-\lambda_k & \sigma s\\ 1 & s-\lambda_k\end{pmatrix}\begin{pmatrix}a_k\\ 1\end{pmatrix} = 0. \tag{4.12}$$

We obtain $a_1 = \lambda_1 - s = 1 - \lambda_2$ and $a_2 = \lambda_2 - s = 1 - \lambda_1$, and therefore $\vec{A}_1 = \begin{pmatrix}1-\lambda_2\\ 1\end{pmatrix}$ and $\vec{A}_2 = \begin{pmatrix}1-\lambda_1\\ 1\end{pmatrix}$. The two eigenvectors are used to construct the diagonalising matrix and its inverse[3]

$$\mathbf{A} = \begin{pmatrix}1-\lambda_2 & 1-\lambda_1\\ 1 & 1\end{pmatrix}, \mathbf{A}^{-1} = \frac{1}{\lambda_1-\lambda_2}\begin{pmatrix}1 & \lambda_1-1\\ -1 & 1-\lambda_2\end{pmatrix}. \tag{4.13}$$

We can now write Eq. (4.7) explicitly in terms of λ_1 and λ_2:

$$Q_N = (1,0)\mathbf{A}\begin{pmatrix}\lambda_1^N & 0\\ 0 & \lambda_2^N\end{pmatrix}\mathbf{A}^{-1}\begin{pmatrix}1\\ 1\end{pmatrix} = \tag{4.14}$$

$$= \frac{1}{\lambda_1-\lambda_2}\left[(1-\lambda_2)\lambda_1^{N+1} - (1-\lambda_1)\lambda_2^{N+1}\right]. \tag{4.15}$$

From the partition function we can compute any observable we need. The most interesting quantity is the average fraction of residues that are in the helical state h, which can be compared to experiments. First of all, we note from the C_i and H_i we explicitly calculated and from Eq. (4.5) that the partition function is a polynomial in s, and therefore can be written as

$$Q_N = \sum_k g_k s^k, \tag{4.16}$$

where g_k are coefficients that do not depend on s, but only on σ and on the multiplicity of each state with k helical residues. Therefore, by definition the probability that the chain has exactly k helical residues is $P_N(k) = \frac{g_k s^k}{Q_N}$, which means that the average number of helical residues is

[3]There is a typo in the Appendix B of Finkelstein and Ptitsyn [2016]: the $(2,1)$ element of A^{-1} should be -1 and not 1.

$$\langle n_h \rangle = \frac{\sum_k k g_k s^k}{Q_N} = \frac{1}{Q_N} \sum_k s g_k \frac{ds^k}{ds} = \frac{s}{Q_N} \frac{d}{ds} \sum_k g_k s^k = \frac{s}{Q_N} \frac{\partial Q_N}{\partial s}, \tag{4.17}$$

since $\frac{ds^k}{ds} = k s^{k-1}$ and only s^k depends on s. Defining the fraction of helical residues, $\theta_N \equiv \langle n_h \rangle / N$, and exploiting the known property of logarithms we find

$$\theta_N = \frac{s}{N} \frac{\partial \log Q_N}{\partial s} = \frac{1}{N} \frac{\partial \log Q_N}{\partial \log s}. \tag{4.18}$$

Substituting Eq. (**??**) it is possible to write θ_N in the following closed, albeit rather cumbersome, form :

$$\theta_N = \frac{\lambda_1 - 1}{\lambda_1 - \lambda_2} \frac{1 + \left(\frac{\lambda_2}{\lambda_1}\right)^{N+1} - \frac{2}{N} \frac{\lambda_2}{\lambda_1 - \lambda_2} \left[1 - \left(\frac{\lambda_2}{\lambda_1}\right)^N\right]}{1 + \frac{\lambda_1 - 1}{1 - \lambda_2} \left(\frac{\lambda_2}{\lambda_1}\right)^{N+1}} \tag{4.19}$$

For sufficiently long chains, $(\lambda_2/\lambda_1)^N \ll 1$ and the expression can be greatly simplified:

$$\theta_N \approx \frac{\lambda_1 - 1}{\lambda_1 - \lambda_2} \left(1 - \frac{2}{N} \frac{\lambda_2}{\lambda_1 - \lambda_2}\right) \xrightarrow[N \to \infty]{} \frac{\lambda_1 - 1}{\lambda_1 - \lambda_2} = \frac{1}{2} + \frac{s - 1}{\sqrt{\left(\frac{s-1}{2}\right)^2 + \sigma s}}. \tag{4.20}$$

The model as described and solved takes into account the cooperativity of the transition through the parameter σ. It is instructive to look at the non-cooperative solution, which can be obtained by setting $\sigma = 1$. In this case $\lambda_1 = 1 + s$ and $\lambda_2 = 0$, so that

$$\theta_N = \frac{s}{s + 1}, \tag{4.21}$$

which is independent of N and describes the simple chemical equilibrium of a $c \rightleftharpoons h$ reaction. Indeed, defining the equilibrium constant $K_{hc} \equiv [h]/[c]$, where $[x]$ is the concentration of x, we find that the fraction of residues in the helical state is

$$\theta = \frac{[h]}{[h] + [c]} = \frac{K_{hc}}{K_{hc} + 1}, \tag{4.22}$$

which, if compared with Eq. (4.21), shows that $s = K_{hc}$.

A comparison between the Zimm-Bragg theory and experiments on poly-γ-benzyl-L-glutamate of different lengths are reported in Figure 4.4. It is clear that there is a strong N-dependence that shows the cooperative nature of the transition. Moreover, in fitting the curves it is assumed that σ does not depend on T, since it has a purely entropic origin.

The Zimm-Bragg *vs.* 1D Ising model

The Zimm-Bragg model is often said to be equivalent to the 1D Ising model, where the coupling between the spins is linked to the initiation free energy and the external field is linked to the hydrogen-bonding contribution. However, this is not correct, as it has been formally demonstrated that the Zimm-Bragg model is equivalent to a Potts-like model, which is a generalisation of the Ising model that can be, in turn, mapped into a 1D Ising model with a temperature-dependent coupling (see Badasyan et al. [2010] for additional details).

However, the regular 1D Ising model can still be used to study the coil-helix transition. Check the notes of the Soft and Biological Matter by Prof. Sciortino, where the model is solved with a similar method as the one used here (the transfer matrix method).

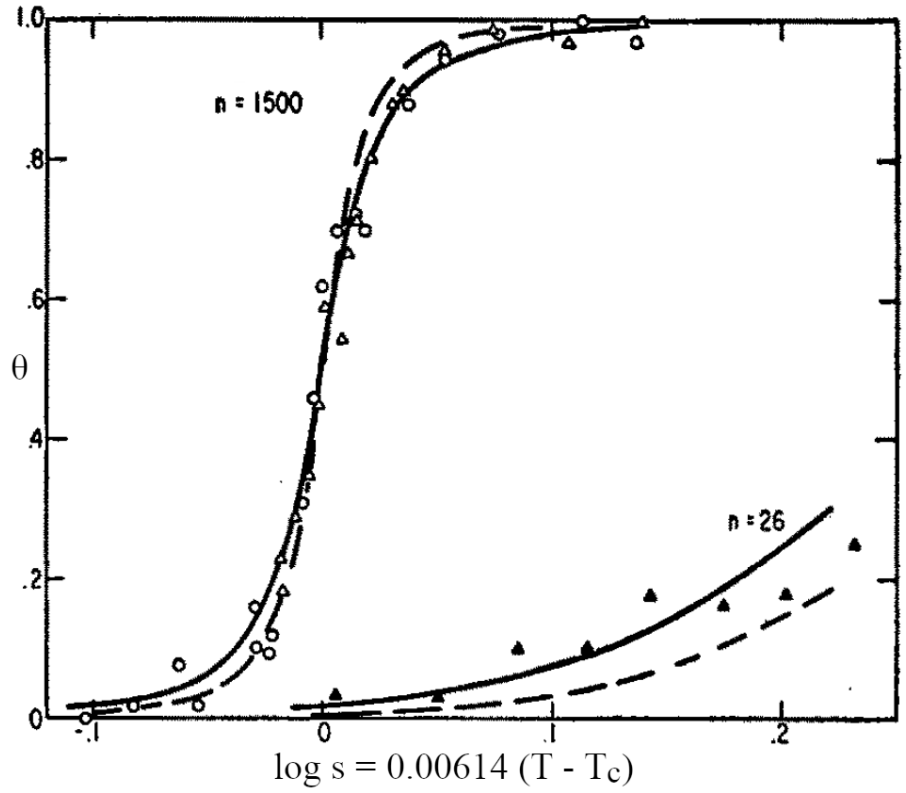

Figure 4.4: Comparison between experimental data (points) and theoretical fits to Eq. (4.19) (lines). The number of monomers is $N = 1500$ and $n = 26$ for the top and bottom curves, respectively. s is fixed by the relation $\log s = 0.00614(T - T_c)$, where T_c is the temperature at which $\theta = 0.5$. Solid and dashed lines correspond to $\sigma = 2 \times 10^{-4}$ and $\sigma = 10^{-4}$, respectively. Adapted from Zimm and Bragg [1959].

Python implementation
Head over here for a Python notebook where the average helicity is computed exactly with Eq. (4.19), and numerically by performing a simulation. The simulation makes use of the Monte Carlo Metropolis method, which is also introduced.

4.1.2 The HP model

Globular proteins exhibit a unique characteristic compared to other isolated polymer molecules: their native structure is unique, meaning that the path formed by the backbone is essentially fixed, bar some small fluctuations. The question arises: what forces, determined by the amino acid sequence, contribute to this remarkable specificity? Of course, as discussed before, a significant constraint is provided by the fact that native structures are compact. However, even for compact chain molecules, many conformations are possible, with the number of maximally compact conformations increasing Chan and Dill [1989]. Among the various physically accessible compact conformations, there are in principle many different interactions that can impact their thermodynamic stability and thus select the native structure. However, it is now well accepted that the interaction type that most contribute to this conformational reduction is hydrophobicity: given a specific amino acid sequence, the number of compact conformations that it can take is greatly reduced by the constraint that residues with hydrophobic side chains should reside inside the globule, while polar and hydrophilic residues should be on the protein's surface (see *e.g.* Finkelstein and Ptitsyn [2002] and Dill [1990]).

A simple model that has been used to demonstrate the importance of hydrophobic interactions in selecting the native structure is the Lau and Dill [1989]. In this model, proteins are abstracted as sequences composed solely of hydrophobic (H) and polar (P) amino acids. In the HP model, the protein sequence is mapped onto a grid, or lattice, which can be two-dimensional (2D) or three-dimensional (3D). Common lattice types used include square lattices for 2D models and cubic lattices for 3D models, but other choices are possible.

Here I will present the 2D square lattice version of the model[4]. Each of the N amino acids of the protein is assigned a type: "H" means hydrophobic, and "P" means polar, so that the sequence of the protein is then the ordered list of amino acids (*e.g.* "HHPHPPPHHP"). The i-th amino acid in the sequence is identified by its index i and occupies a point on this grid, and two amino acids cannot occupy the same grid point. Two amino acids i and j for which $|j-i|=1$ are "connected neighbours" and share a backbone edge representing the peptide bond, which on the square lattice can be either horizontal or vertical. By contrast, two amino acids i and j for which $|i-j|>1$ that are adjacent in space but not along the sequence are said to be "topological neighbours". Every pair of HH topological neighbours form a topological contact that contributes an amount of free energy $\epsilon \equiv \beta\Delta F_{HH} < 0$, while the other contacts give ϵ_{HP} and ϵ_{PP}. In the simplest version of the model $\epsilon_{HP} = \epsilon_{PP} = 0$. In this case, the total (dimensionless) free energy of a conformation is

$$\beta E_i = m\epsilon, \tag{4.23}$$

where m is the number of HH topological contacts. Following Lau and Dill [1989], the lowest-energy (highest-m) conformations of a chain with a specific sequence and a length N are called "native".

The concept of topological contacts is also useful to describe the degree of compactness of a conformation, defined as

$$\rho \equiv \frac{t}{t^{\max}}, \tag{4.24}$$

where t is the number of topological contacts, irrespective of the type of neighbouring residues, and $t^{\max}$ is the largest possible number of topological contacts for a chain of size N, which is

[4]For the chain lengths usually considered, the ratio of the numbers of residues exposed to the solvent or buried in the protein core is closer to that of real proteins compared to 3D lattices (see MorenoHernández and Levitt [2012] and references therein).

$$t^{\max} = N + 1 - \frac{P_{\min}}{2}, \tag{4.25}$$

where $P_{\min}$ is the perimeter of the smallest box that can completely surround the protein. By this definition, (maximally) compact conformations have $\rho = 1.0$.

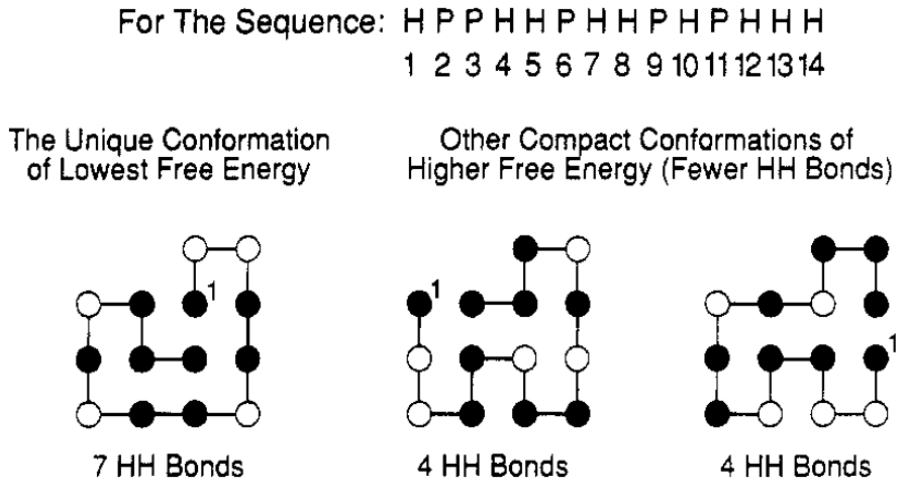

Figure 4.5: The sequence "HPPHHPHHPHPHHH" has a single conformation with 7 topological contacts (shown on the left). All other conformations, like the two on the right, have fewer hydrophobic contacts. Hydrophobic and polar residues are shown as black and white circles, respectively. Taken from Dill [1990].

As an example, Figure 4.5 shows the single lowest-energy conformation of a given sequence, together with two other compact conformations of higher free energy. It is rather evident that the native conformation has a core that is richer in hydrophobic residues compared to the other two.

In order to be more quantitative, some useful observables can be introduced. In general, the thermodynamic properties of a chain of length N and a given sequence can be computed from the partition function:

$$Q = \sum_{i=1}^{\Omega_N} e^{-E_i} \tag{4.26}$$

where Ω_N is the number of distinct conformations of an N-chain, so that the sum runs over all the conformations. If the length of the chains is not too large[5], it is possible to numerically perform a complete (exhaustive) enumeration of all possible chain conformations, from which any quantity of interest can be computed as an ensemble average, *viz.*

$$\langle A \rangle = \frac{1}{Q} \sum_{i}^{\Omega_N} A e^{E_i}. \tag{4.27}$$

The partition function can also be written as a sum over conformations of fixed number of HH topological contacts m:

$$Q = \sum_{m=0}^{m^{\max}} g(m) e^{-m\epsilon}, \tag{4.28}$$

where $m^{\max}$ is the maximum number of HH topological contacts and $g(m)$ is the number of distinct chain conformations having m. Note that by this definition, the native conformations are those for which $m = m^{\max}$. We also define $G(m, u)$ as the number of conformations that have m HH topological contacts and $u = t - m$ non-HH topological contacts, which is connected to $g(m)$ by

[5]Which values are "too large" depend on the hardware and algorithms used to write the code.

$$g(m) = \sum_{u=0}^{t^{\max}-m} G(m,u). \tag{4.29}$$

Since partition functions are defined up to a multiplicative constant, we multiply Q by $e^{m^{\max}\epsilon}$ to obtain[6]

$$Q = \sum_{m=0}^{m^{\max}} g(m) e^{(m^{\max}-m)\epsilon}. \tag{4.30}$$

With this definition it is straightforward to take averages over native conformations only. Indeed, if we take the $\epsilon \to -\infty$ limit, $e^{(m^{\max}-m)\epsilon}$ is non-zero only for $m = m^{\max}$, and the partition function becomes

$$Q_\infty = g(m^{\max}). \tag{4.31}$$

We now define some useful average values to characterise the different states. The average compactness over all conformations is

$$\langle \rho \rangle = \frac{1}{Q} \sum_{m=0}^{m^{\max}} \sum_{u=0}^{t^{\max}-m} \frac{u+m}{t^{\max}} G(m,u) e^{(m^{\max}-m)\epsilon}, \tag{4.32}$$

and the average compactness over the native conformations is

$$\langle \rho \rangle_{ns} = \frac{1}{Q_\infty} \sum_{u=0}^{t^{\max}-m^{\max}} \frac{u+m^{\max}}{t^{\max}} G(m^{\max},u) \tag{4.33}$$

where we used the fact that

$$\lim_{\epsilon \to -\infty} \sum_{m=0}^{m^{\max}} G(m,u) e^{(m^{\max}-m)\epsilon} = G(m^{\max},u) \tag{4.34}$$

The average energy is proportional to the average number of HH contacts, which is

$$\langle m \rangle = \frac{1}{Q} \sum_{m=0}^{m^{\max}} m g(m) e^{(m^{\max}-m)\epsilon} \tag{4.35}$$

Finally, we know already that most of the hydrophobic residues are buried in the core of globular proteins. In the HP model we define the core as the set of residues that are completely surrounded by other residues. Let the number of core residues be n_i, and the number of hydrophobic core residues be n_{hi}, then the degree of hydrophobicity of a given conformation can be estimated as

$$x \equiv \frac{n_{hi}}{n_i}, \tag{4.36}$$

and its averages over all and native conformations can be computed as

$$\langle x \rangle = \frac{1}{Q} \sum_{i=0}^{\Omega_N} x e^{(m^{\max}-m_i)\epsilon} \tag{4.37}$$

$$\langle x \rangle_{ns} = \frac{1}{g(m^{\max})} \sum_{i=1}^{g(m^{\max})} x. \tag{4.38}$$

The ensemble averages of these quantities, as well as of the partition function, which is a measure of the number of available conformations, as a function of $-\epsilon$ for two sequences are shown in Figure 4.6. Looking at the top panels it is clear that if the HH interaction is strong the preferred conformations

[6]Here I'm abusing the notation by using the same symbol Q to represent two different (but very related) quantities.

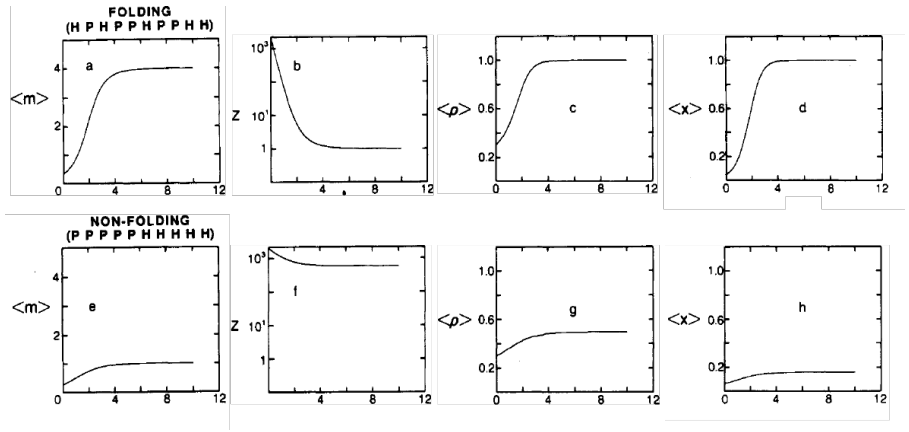

Figure 4.6: Ensemble averages of folding (top) and non-folding (bottom) sequences. Here Z is the value of the partition function, which I call Q. All quantities are plotted as a function of $-\epsilon$. Adapted from Lau and Dill [1989].

of the HPHPPHPPHH sequence have low energy, are few ($Q \to 1$), compact, $\langle\rho\rangle = 1$, and have a purely hydrophobic core, $\langle x\rangle = 1$: this is a "folding" sequence.

By contrast, the bottom panels show a sequence that behaves very differently: the low-energy (native) conformations are not compact and large in number. Looking at the sequence itself, PPPPPHHHHH, it is clear that the hydrophilic "tail" does not make it possible to form a conformation with a hydrophobic core, and the number of HH topological contacts itself cannot be too high. These are all clear signs of a "non-folding" sequence.

Note that the quantities reported in Figure 4.6 are somewhat correlated: folding sequences tend tend to have few (sometimes one) low-energy compact conformations with hydrophobic cores, whereas non-folding sequences tend not to have any of these attributes.

The HP model (or its extensions) can be used to understand some general or even specific aspects of protein folding because of its simplicity. The 2-letter alphabet and the greatly reduced residue conformations due to the underlying lattice makes it possible to fully explore (at the same time) the conformational space (*i.e.* the set of all possible internal conformations of a molecule) and the sequence space (*i.e.* the set of all possible sequences of H and P residues). Both spaces have sizes that grow exponentially with N, and therefore the maximum length of chains that can be investigated by exhaustive enumeration is somewhat limited, but there are computational techniques that can be used to circumvent this issue.

Folding HP sequences

If we now use a stricter definition of a "folding" sequence as a sequence that has a single native conformation, we can draw a comparison between lattice folding sequences and real proteins. First of all, how many folding sequences there are?

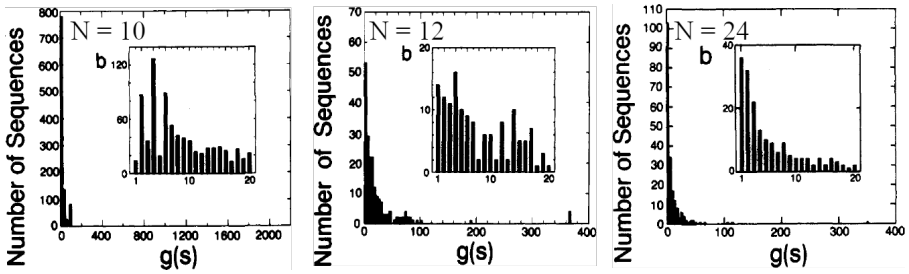

Figure 4.7: The distribution of the number of sequences having $g(s)$ native conformations for chain length $N = 10, 12$ and 24. In the figure $s = m^{\max}$. Adapted from Lau and Dill [1989].

Figure 4.7 shows the distributions of the number of native conformations $g(m^{\max})$, *i.e.* how many

sequences have the given degeneracy, for different chain lengths. As N increases, the distribution becomes more and more peaked on 1: far more sequences have a single lowest-energy conformation rather than, say, 10 or 100. These results suggest that hydrophobicity alone is enough to greatly reduce the number of compact configurations into a small set of folding candidate structures (Dill [1990]), and confirms that the hydrophobic force is a major driving force for protein folding (see *e.g.* Finkelstein and Ptitsyn [2002]).

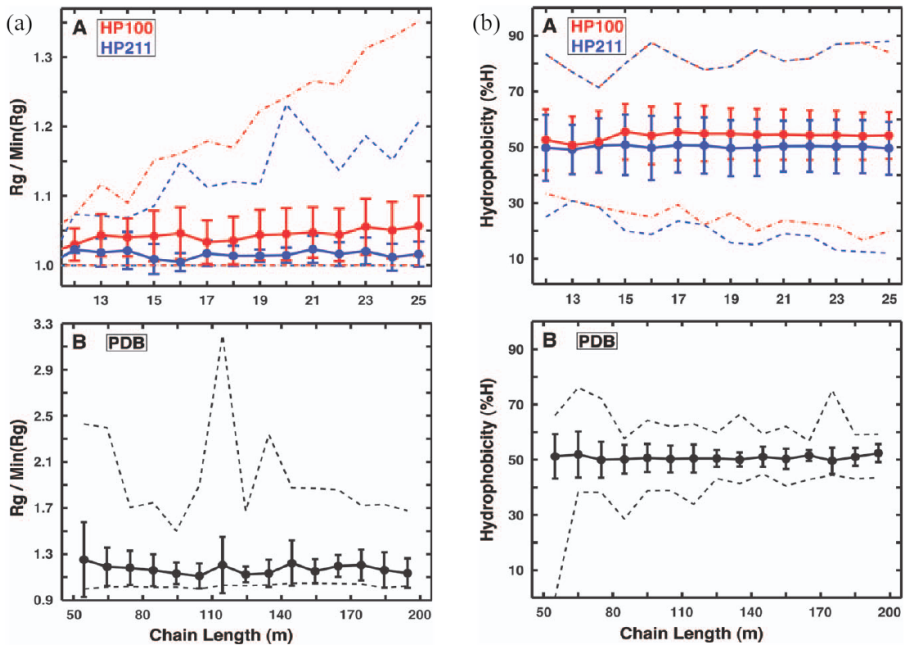

Figure 4.8: (a) Compactness and (b) hydrophobicity as a function of the chain length for two HP lattice models and for real proteins. Dashed lines show the range of values. Adapted from MorenoHernández and Levitt [2012].

But to what extent lattice and real proteins are similar? I will present some results reported in MorenoHernández and Levitt [2012]. Therein, an exhaustive enumeration of all conformations of chains of lengths up to 25 has been carried out with two HP models:

- In HP100, which is the one described above, only HH contacts contribute to a conformation's free energy.
- In HP211 topological contacts between H and P and P and P contribute -1 (*i.e.* $\epsilon_{HP} = \epsilon_{PP} = -1$) to the free energy, while HH contacts contribute $\epsilon = -2$. As a result, compared to the HP100 model, all residues feel an attraction that makes compact conformations more favourable.

Some interesting results pertaining to folding sequences are reported in Figure 4.8. Figure 4.8(a) shows, as a proxy for compactness, the normalised radius of gyration $R_g/\min(R_g)$, where $\min(R_g)$ is the smallest radius of gyration among the structures of a given length, for the lattice and real proteins. For real proteins the authors have considered a non-redundant set of 2401 single-domain proteins from the PDB with lengths between 50 and 200 residues long. By comparing the two lattice models, it is evident that the additional attraction between non-hydrophobic residues enhances the compactness in HP211 more, but the variation in compactness, given by the error bars, is very small in both cases, and the compactness itself does not depend on the chain length. The same behaviour is observed in real proteins.

Figure 4.8(b) shows the percentage of residues that are hydrophobic in the lattice structures and for the same real proteins used for the radius of gyration. Both lattice and real proteins have an average hydrophobicity of $\approx 50\%$, independent of m, and the error bars and total range (dashed lines) are comparable in the two cases (although both are a bit larger for the HP model). Folding sequences in the HP model are "protein-like".

Designable HP conformations

Now we focus on compact conformations. How many sequences have a given compact conformation as their native (*i.e.* lowest-energy) state? Does the answer depend on some properties of the selected conformation itself? In the context of the HP model, some of the first answers were provided Li et al. [1996] with a HP model with $\epsilon = -2.3$, $\epsilon_{HP} = -1$ and $\epsilon_{PP} = 0$, chosen in order to fulfill the following physical constraints, rooted in the analysis of real proteins:

1. Compact conformations have lower energies compared to non-compact ones.
2. Hydrophobic residues should have the tendency to be buried in the core ($\epsilon_{PP} > \epsilon_{HP} > \epsilon$).
3. Different types of monomers should tend to segregate ($2\epsilon_{HP} > \epsilon_{PP} + \epsilon_{HH}$).

The output of the computation was to enumerate all the native structures of each possible sequence on a 3D 3x3x3 cube and on 2D 4x4, 5x5, 6x5 and 6x6 boxes. The result of this complete enumeration is the list of all possible sequences that can "design" a given structure, or, in other words, that have that structure as their unique native state. The size of this list, N_s is a measure of the designability of a given structure.

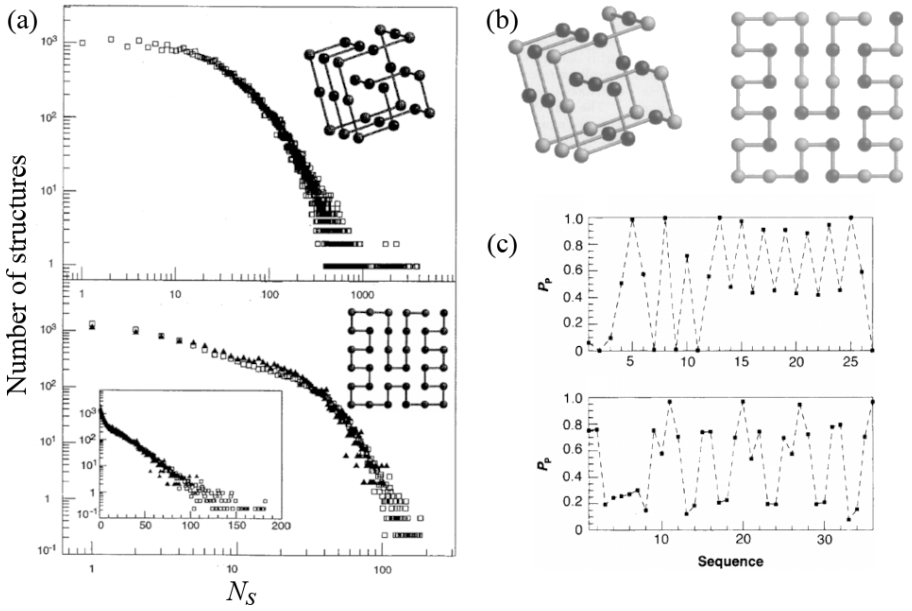

Figure 4.9: (a) The number of structures with the given N_s for the (top) 3x3x3 cube and (bottom) 6x6 square. In the bottom panel, the number of structures goes below 1 for large N_s, which, given its definition, should not happen. Perhaps the authors normalised the data in some way that, as far as I understand, is not specified in the original paper. (b) The most designable (A) 3D and (B) 2D structures. Hydrophobic and polar residues are coloured in black and grey, respectively. (c) The probability of finding a polar residue as a function of the sequence index for the same two structures. Adapted from Li et al. [1996].

Compact structures differ markedly in terms of their designability: there are structures that can be designed by a large number of sequences, and there are "poor" structures that can be designed by only a few or even no sequences. In fact, for $\approx 10\%$ of the conformations there is no sequence that has that structure as its ground state. The majority of the sequences ($\approx 95\%$ out 2^{27} total sequences in 3D) have degenerate ground states, *i.e.* more than one compact conformation of lowest energy, which means that in 3D there exist almost 7 million sequences of length 27 that have a unique compact ground state. Moreover, the number of structures with a given N_s value decreases continuously and monotonically as N_S increases. The data for the 3D and largest 2D cases is shown in Figure 4.9(a), where the long tails of the distributions, with some structures being the ground states of thousands of sequences, highlight the presence of "highly-designable" compact conformations.

Structures with large N_S exhibit specific motifs (*i.e.* secondary structures) that small N_S compact structures lack. For instance, the 3D compact structures with the 10 largest N_S values contain parallel running lines packed regularly and eight or nine strands (three amino acids in a row), sensibly more than the average compact structure. Figure 4.9(b) shows the most designable structures in 3D and in 2D, where it is easy to spot the regularity of the "secondary structures" and, for the 3D structure, also the "strands".

With the simple lattice model is also possible to directly assess the effect of mutations, which in real proteins is particularly important in the context of homologous sequences (sequences related by a common ancestor). Indeed, focussing on highly designable structures and referring to the N_s different sequences that fold into them as "homologous", it is possible to observe phenomena that are qualitatively similar to those observed in real proteins. For example, sequences that differ by more than half of their residues can design the same structure (see Figure 4.20, which will be discussed later, for a real-protein example). Looking at the effect of mutations, the rightmost panel of Figure 4.9(c) shows that some in very designable conformations residues are highly mutable, whereas others are highly conserved, with the conserved sites being those sites with the smallest or largest number of sides exposed to water.

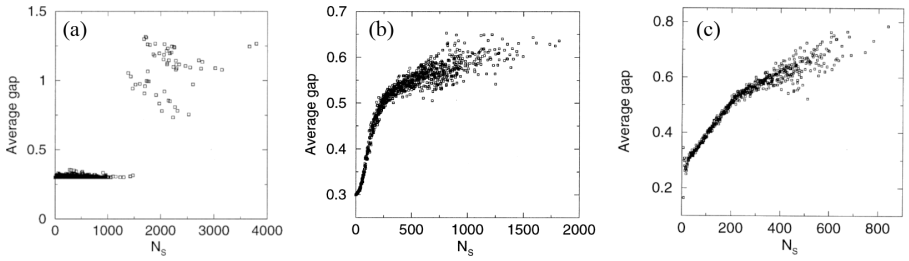

Figure 4.10: The average energy gap between the lowest-energy and first excited states of the (a) 3D 3x3 cube and (b) 2D 6x6 HP models, as well as of the (c) 2D 6x6 MJ model. Adapted from Helling et al. [2001]).

Another important feature of proteins that is also reproduced by the model is the large energy gap between the lowest-energy and first excited conformations, which stabilises the native structures from the thermodynamic point of view. In the lattice model, the average energy gap $\bar{\delta}_S$ is defined as the minimum energy required to change a native conformation to a different compact structure, averaged over all the N_S sequences that design that native conformation. Figure 4.10 shows that there is a sudden jump (in 3D) or change of slope (in 2D) at a specific value of N_S. Therefore, the conformations that are highly designable also have large energy gaps, making them highly stable from a thermodynamic point of view.

Note

In Helling et al. [2001] the energy gap was defined as the energy difference between the ground and first excited states of a sequence. However, a much better indicator for "good foldability" and overall thermodynamic stability is the so-called stability gap, which is defined as the difference between the energy of the native state and the **average energy** of compact (misfolded) states (see *e.g.* Onuchic et al. [1997]).

As shown in the rightmost panel of Figure 4.10, the same behaviour is also observed when a 20-letter (rather than 2-letter) alphabet is used, with the interactions between the residues being modelled using the Miyazawa and Jernigan [1985], which was derived by analysing the contact frequencies of amino acids from protein crystal structures.

Although the ingredients in the simple lattice model are minimal, lacking many of the factors that contribute to the structure of proteins, the results that have been obtained with it contributed to the understanding that there is a link between designability and thermodynamic stability: "From an evolutionary point of view, highly designable structures are more likely to have been chosen through

random selection of sequences in the primordial age, and they are stable against mutations" (Li et al. [1996]).

Python implementation
Head over here for a Jupyter notebook containing the code to exhaustively explore the conformation and sequence space of the 2D HP model.

4.1.3 Sequence alignment

Tip
The main references for this part are Kellis [2016] and Miklós [2016].

Simple models are useful to understand the underlying physics of some particular phenomena. However, how can we understand something very specific, like what is the 3D structure of a particular sequence? The simplest way is to look for similarities: if we already have a list of sequence → structure connections, we can try to look whether the new sequence, for which the 3D structure is unknown, is similar, and to what degree, to another for which the 3D structure is already known. This operation is called "sequence alignment" (SA).

Sequence alignment is a fundamental technique in bioinformatics. The primary goal of sequence alignment is to identify regions of similarity that may indicate functional, structural, or evolutionary relationships between the sequences being compared. In the context of proteins, sequence alignment can be useful for reasons that go beyond the 3D structure prediction:

- It helps predict the function of unknown proteins based on their similarity to known proteins.
- It aids in understanding evolutionary relationships by identifying conserved sequences among different species, allowing the construction of phylogenetic trees.
- It can identify conserved domains that are crucial for the function of proteins.
- It helps in identifying potential drug targets by finding unique sequences in pathogens that differ from the host.
- It assists in annotating genomes by finding homologous sequences, thus providing insights into gene function and regulation.

Turning a biological problem into an algorithm that can be solved on a computer requires making some assumptions in order to obtain a model with relative simplicity and tractability. In practice, for our sequence alignment model we ignore realistic events that occur with low probability (*e.g.* duplications) and focus on the following three mechanisms through which sequences vary:

1. A (point) mutation, or substitution, occurs when some amino acid in a sequence changes to some other amino acid during the course of evolution.
2. A deletion occurs when an amino acid is deleted from a sequence during the course of evolution.
3. A insertion occurs when an amino acid is added to a sequence during the course of evolution.

There are many possible sequences of events that could change one genome into another, and we wish to establish an optimality criterion that allows us to pick the "best" series of events describing changes between sequences. We choose to invoke Occam's razor and select a maximum parsimony method as our optimality criterion[7]: we wish to minimize the number of events used to explain the differences between two sequences. In practice, it is found that point mutations are more likely to occur than insertions and deletions, and certain mutations are more likely than others. We now develop an algorithmic framework where these concepts can be incorporated explicitly by introducing parameters that take into account the "biological cost" of each of these changes.

[7]Note that this is not the only possible choice: we could choose a probabilistic method, for example using Hidden Markov Models (HMMs), that would assign a probability measure over the space of possible event paths and use other methods for evaluating alignments (*e.g.*, Bayesian methods).

Homology

While here my focus is on protein structure, I also want to point out that in bioinformatics one of the key goals of sequence alignment is to identify homologous sequences (*e.g.*, genes) in a genome. Two sequences that are homologous are evolutionarily related, specifically by descent from a common ancestor. The two primary types of homologs are orthologous and paralogous. While they are outside the scope of these notes, I note here that other forms of homology exist (*e.g.*, xenologs, which are genes in different species that have arisen through horizontal gene transfer[8] rather than through vertical inheritance from a common ancestor, and often perform similar functions in their respective organisms despite their distinct evolutionary origins).

- Orthologs arise from speciation events, leading to two organisms with a copy of the same gene. For example, when a single species A speciates into two species B and C, there are genes in species B and C that descend from a common gene in species A, and these genes in B and C are orthologous (the genes continue to evolve independent of each other, but still perform the same relative function).
- Paralogs arise from duplication events within a species. For example, when a gene duplication occurs in some species A, the species has an original gene B and a gene copy B', and the genes B and B' are paralogus.

Generally, orthologous sequences between two species will be more closely related to each other than paralogous sequences. This occurs because orthologous typically (although not always) preserve function over time, whereas paralogous often change over time, for example by specializing a gene's (sub)function or by evolving a new function. As a result, determining orthologous sequences is generally more important than identifying paralogous sequences when gauging evolutionary relatedness.

Definitions

Following Kellis [2016], we introduce a simplified version of the alignment problem and make it more complex by adding the required ingredients. First, some definitions:

Sequence In mathematics (and combinatorics) a sequence is a series of characters taken from an alphabet Σ. For what concerns us, DNA molecules are sequences over the alphabet $\{A, C, G, T\}$, RNA sequences over the alphabet $\{A, C, G, U\}$, and proteins are sequences over the alphabet $\{A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$.

Substring A consecutive part of a sequence. A sequence of length N has N substrings of length 1, $N(N-1)/2$ substrings of length 2, ..., 2 substrings of length $(N-1)$. Given the sequence "SUPERCALIFRAGILISTICHESPIRALIDOSO", "FRAGILI" is a substring, but "SUPERDOSO" is not.

Subsequence A set of characters of a sequence. The characters do not have to be consecutive, but they should be ordered like in the original sequence. Given the sequence "SUPERCALIFRAGILISTICHESPIRALIDOSO", "FRAGILI" and "SUPERDOSO" are two subsequences, but "SOSUPER" is not. Formally, given a sequence $S = (s_1, s_2, \ldots, s_N)$, $K = (z_1, z_2, \ldots, s_n)$, with $n \leq N$, is a subsequence of S if there exists a strictly increasing sequence $i_1 < i_2 < \ldots < i_n$ of indices of S such that $S_{i_j} = K_j$ for each $j : 1 \leq j \leq n$.

Sequence alignment An alignment of two sequences S and T, defined over the same alphabet Σ, is a $2 \times L$ table containing either a character from Σ, or the gap symbol "-". In the table there is no column in which both characters are "-", and the non-gap characters in the first line give back sequence S, while the non-gap characters in the second line give back sequence T.

[8] Horizontal gene transfer is the movement of genetic material between organisms in a manner other than traditional reproduction (vertical inheritance). This process allows genes to be transferred across different species, leading to genetic diversity and the rapid acquisition of new traits, and can occur through several mechanisms.

Problem formulation

We warm up by considering the following simple problem: given two sequences, S and T, what is their longest common substring? We take the two following DNA fragments as examples: `ACGTCATCA` and `TAGTGTCA`. The problem can be solved by aligning the first characters of the two sequences, and then shifting one, say T, by an integer amount $i \in \mathcal{N}$. By computing the number of matching characters for every i we can find the optimal alignment, which in this case is given by $i = -2$:

```
- -ACGTCATCA
 - -xx|||| - - -
TAGTGTCA - - -
```

The first and third rows of the diagram above describe the alignment itself, as introduced earlier, showing that gaps are characters that align to a space, while the central row shows the character matches: here `x` and `|` stand for mismatches and matches, respectively. In this example, the longest common substring is `GTCA`. The algorithm I just described has a complexity of $\mathcal{O}(n^2)$, where n is the length of the shortest sequence.

A more complicated problem is to find the longest common *subsequence* (LCS) between two sequences. In the language we are introducing, this means allowing internal gaps in the alignment. The LCS between the S and T sequences defined earlier is

```
-ACGTCATCA
 -| -||x -|||
TA -GTG -TCA
```

In this case the LCS is `AGTTCA`, which is longer than the longest common substring.

The LCS problem as formulated here is a particular case of the full sequence-alignment problem. In order to generalise the problem, we introduce a cost function that makes it possible to recast it in terms of an optimisation problem. Given two sequences S and T, we define $\delta(S, T)$ as the "biological cost" of turning S into T. For the case just considered, I implicitly used as a cost function the Levenshtein (or edit) distance, which is defined as the minimum number of single-character edits required to change one string into another. In other words, the problem has been formulated by implicitly considering that all possible operations (mutations, insertions and deletions) are equally likely. In a more generalised formulation, each edit operation is associated to a specific cost (penalty) that should reflect its biological occurrence, as discussed Section 4.1.3.

What about gaps? In general, the cost of creating a gap depends on many variables. There are varying degrees of approximations that can be taken in order to simplify the problem. At the zeroth order each gap can be assumed to have a fixed cost, as we implicitly did above. This is called the "linear gap penalty". An improvement can be done by considering that, biologically, the cost of creating a gap is more expensive than the cost of extending an already created gap. This is taken into account by the "affine gap penalty" method, whereby there is a large initial cost for opening a gap, and then a small incremental cost for each gap extension. There are other (more complicated and more context-specific) methods that can be considered and that we will not analyse further here, such as the general gap penalty, which allows for any cost function, and the frame-aware gap penalty, which is applied to DNA sequences and tailors the cost function to take into account disruptions to the coding frame[9].

The Needleman-Wunsch algorithm

Once we allow for gaps, the enumeration of all possible alignments (as done for the longest common substring method) becomes unfeasible. Indeed, the number of non-boring alignments, *i.e.* alignments where gaps are always paired with characters, in a sequence of size N containing G gaps (where $N \approx G$) can be estimated as

[9] Indels (shorthand for "insertions/deletions") that cause frame-shifts in functional elements generally cause important phenotypic modifications.

$$\binom{N+G}{G} = \frac{(N+G)!}{N!G!} \approx \frac{(2N)!}{(N!)^2} \approx \frac{2^{2N}}{\sqrt{\pi N}}, \tag{4.39}$$

where we used the second-order Stirling's approximation, $\log(N!) \approx N\log(N) - N + \frac{1}{2}\log(2\pi N)$. Note that this number grows **very** fast: for $N = 30$ it is already larger than 10^{17}. Considering that we also have to compute the score of each alignment, it is clear that the problem is untractacle with a brute-force method. Here is where dynamic programming enters the field.

Suppose we have an optimal alignment for two sequences S, of length N, and T, of length M, in which S_i matches T_j. The alignment can be conceptually split into three parts:

1. Left subalignment: The alignment of the subsequences $(S_1, \ldots, S_{i-1})$ and $(T_1, \ldots, T_{j-1})$.
2. Middle match/mismatch: The alignment of S_i with T_j (this could be a match or a mismatch).
3. Right subalignment: The alignment of the subsequences $(S_{i+1}, \ldots, S_N)$ and $(T_{j+1}, \ldots, T_M)$.

The overall alignment score is the sum of the scores from these three parts: the score of the left subalignment, the score for aligning S_i with T_j, and the score of the right subalignment. Therefore, if the overall alignment is optimal, both the left and right subalignments must themselves be optimal. This follows from a cut-and-paste argument: imagine that one of the two subalignment was not optimal. This would mean there exists another alignment of these subsequences with a higher score. If such a better alignment for the subsequences existed, we could "cut" the suboptimal subalignment from the original alignment and "paste" in this better alignment. This would result in a new alignment for S and T with a higher total score than the supposed "optimal" alignment. This is a contradiction because the original alignment was assumed to be optimal. Since the same argument applies to both subalignments, the overall alignment's optimality depends on the optimality of both its left and right subalignments. Of course, this is true only if the scores are additive, so that the score of the overall alignment is the sum of the scores of the alignments of the subsequences. The implicit assumption is that the sub-problems of computing the optimal scoring alignments of the subsequences are independent.

Let $F_{i,j}$ be the score of the optimal alignment of $(S_1, \ldots, S_i)$ and $(T_1, \ldots, T_j)$. Since $i \in [0, N]$ and $j \in [0, M]$, the matrix $\hat{F}$ storing the solutions (*i.e.* optimal scores) of the subproblems has a size of $(M+1) \times (N+1)$.

We can compute the optimal solution for a subproblem by making a locally optimal choice based on the results from the smaller subproblems. Thus, we need to establish a recursive function that shows how the solution to a given problem depends on its subproblems and can be used to fill up the matrix $\hat{F}$. At each iteration we consider the four possibilities (insert, delete, substitute, match), and evaluate each of them based on the results we have computed for smaller subproblems.

We start by considering the linear gap penalty model, and define d, with $d < 0$, as the cost of a gap. We are now equipped to set the values of the elements of the matrix. Let's consider the first row: the value of $F_{0,j}$ is the cost of aligning a sequence of length 0 (taken from S) to a sequence of length j (taken from T), which can be obtained only by adding j gaps, yielding $F_{0,j} = jd$. Likewise, for the first column we have $F_{i,0} = id$. Then, we traverse the matrix element by element. Let's consider a generic element F_{ij}: this is the cost of aligning the first i characters of S to the first j characters of T. There are three ways we can obtain this alignment:

- Align S_i to a gap (*i.e.* a gap is added to T): the total cost is $F_{i-1,j} + d$;
- Align T_j to a gap (*i.e.* a gap is added to S): the total cost is $F_{i,j-1} + d$;
- S_i and T_j are matched: the total cost is $F_{i-1,j-1} + s(S_i, T_j)$, where $s(x, y)$ is the cost of (mis)matching x and y.

Leveraging the cut-and-paste argument sketched above, the optimal cost is given by the largest of the three values. Formally,

$$F_{i,j} = \max \begin{cases} F_{i-1,j} + d \\ F_{i,j-1} + d \\ F_{i-1,j-1} + s(S_i, T_j). \end{cases} \tag{4.40}$$

This is a recursive relation: computing the value of any $F_{i,j}$ requires the knowledge of the values of its left, top, and top-left neighbours. Therefore, the fill-in phase amounts to traversing the table in row or column major order, or even diagonally from the top left cell to the bottom right cell. After traversing the matrix, the optimal score for the alignment is given by the bottom-right element, F_{MN}. In order to obtain the actual alignment we have to traceback through the choices made during the fill-in phase. It is helpful to maintain a pointer for each cell while filling up the table that shows which choice was made to get the score for that cell. Then the pointers can be followed backwards to reconstruct the optimal alignment.

The complexity analysis of this algorithm is straightforward. Each update takes $\mathcal{O}(1)$ time, and since there are MN elements in the matrix $\hat{F}$, the total running time is $\mathcal{O}(MN)$. Similarly, the total storage space is $\mathcal{O}(MN)$. For the more general case where the update rule is more complicated, the running time may be more expensive. For instance, if the update rule requires testing all sizes of gaps (*e.g.* the cost of a gap is not linear), then the running time would be $\mathcal{O}(MN(M+N))$.

A simple example

Consider the two DNA sequences $S = AAGC$ and $T = AGT$, a gap penalty $d = -2$, and $s(x, y) = \pm 1$, where the plus and minus signs are used for matches and mismatches, respectively.

Figure **??** shows how the dynamic programming table of the problem looks when initialised, halfway through the fill-in phase, and after being fully traversed. Once the full table has been computed, the bottom-right box contains the optimal score, while the solutions (*i.e.* the optimal alignments) can be reconstructed by tracing back the matrix, following the arrows. Note that sometimes the maximum value computed from Eq. (4.40) is degenerate, *i.e.* the same value can be obtained by performing several operations (see Figure **??**(b)). In this case there are multiple optimal alignments. For instance, for the simple example shown here there are two optimal alignments:

```
A -GT
| -|x
AAGC

 -AGT
 -||x
AAGC
```

Local alignment: the Smith-Waterman algorithm

The Needleman-Wunsch algorithm find the best possible alignment across the entire length of two sequences. It tries to align every character from the start to the end of the sequences, which means both sequences are considered in their entirety. This is called "global alignment", and it is most useful when the sequences being compared are of similar length and are expected to be homologous across their entire length.

Local alignment, on the other hand, focuses on finding the best alignment within a subset of the sequences. It identifies regions of similarity between the two sequences and aligns only those regions, ignoring the parts of the sequences that do not match well. Local alignment is particularly useful when comparing sequences that may only share a segment of similarity, such as when comparing domains within proteins, detecting conserved motifs, or identifying homologous regions in sequences that may not be overall similar[10], which is why is very useful for the prediction of the 3D structure of proteins (or protein subdomains).

The most used method for local alignment is the Smith-Waterman algorithm, which is a modification of the Needleman-Wunsch algorithm. The key difference between the two lies in how the

[10]It is also valuable in cases where one sequence may be a subsequence of another, like when searching for a gene within a cromosome or a whole genome.

scoring matrices are constructed and scored. In Needleman-Wunsch, every cell in the matrix is filled to reflect the best global alignment, with the final alignment score found in the bottom-right corner of the matrix. Smith-Waterman, on the other hand, sets any negative scores to zero, which allows the algorithm to "reset" when the alignment quality dips. The highest score in the matrix indicates the end of the best local alignment, which is then traced back to a zero to identify the optimal aligned subsequence. This approach ensures that only the most relevant, highest-scoring local alignments are highlighted. Here is how the Smith-Waterman algorithm looks like in practice:

1. Initialisation: since a local alignment can start anywhere, the first row and column in the matrix are set to zeros, *i.e.* $F_{0,j} = jd$, $F_{i,0} = id$.

2. Iteration $\forall(i, j)$: this step is modified so that the score is never allowed to become negative but it is reset to zero. This is done by slightly modifying Eq. (4.40) as follows:

$$F_{i,j} = \max \begin{cases} 0 \\ F_{i-1,j} + d \\ F_{i,j-1} + d \\ F_{i-1,j-1} + s(S_i, T_j). \end{cases} \tag{4.41}$$

3. Trace-back: starts from the position of the maximal number in the table and proceeds until a zero is encountered.

The Needleman-Wunsch algorithm

For reference, this is summary of the Needleman-Wunsch algorithm:

1. Initialisation: $F_{0,j} = jd$, $F_{i,0} = id$.

2. Iteration $\forall(i, j)$: Eq. (4.40).

3. Trace-back: starts from the bottom-right value and stops at the top-left corner.

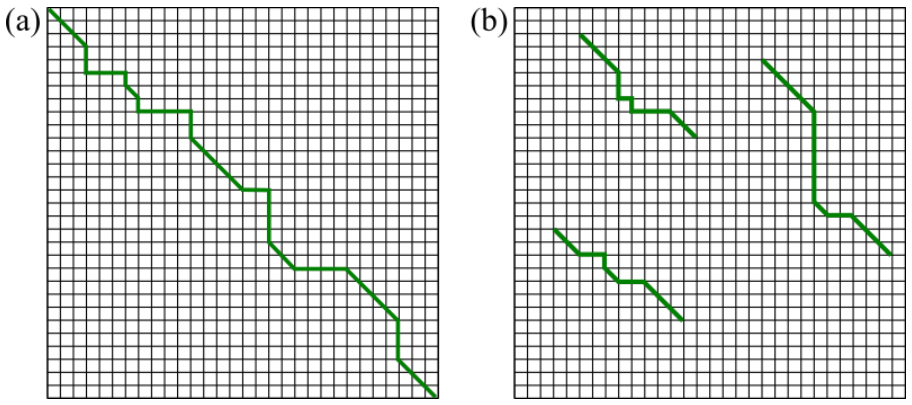

Figure 4.12: Example trace-back solutions generated by (a) global and (b) local alignment methods.

A visual comparison between the tables generated by global and local alignment algorithms is shown in Figure 4.12.

Affine gap penalty

For both the global and local alignment algorithms introduced we have used a linear gap penalty: the cost of adding a gap is constant, regardless of the nature of the aligned character, or of the length of

the gap. From the biological point of view, this means that an indel of *any* length is considered as the result of independent insertions or deletions. However, in reality long indels can form in single evolutionary steps, and in these cases the linear gap model overestimates their cost. To overcome this issue, more complex gap penalty functions have been introduced. As mentioned before, a generic gap penalty function would result in an algorithmic complexity worse than $\mathcal{O}(N^2)$ (where for simplicity I'm considering two sequences of the same length). Let's see why. Any gap penalty function can be implemented in the Needleman-Wunsch or Smith-Waterman algorithms by changing the recursive rule, which can be done trivially for element $F_{i,j}$ by evaluating terms such as $\max_{0\leq k\leq i}\{F_{k,j} + d_{i-k}\}$ and $\max_{0\leq k\leq j}\{F_{i,k} + d_{j-k}\}$, where d_x is the cost of a gap of size x. However, this means that the update of every cell of the dynamic programming matrix would take $\mathcal{O}(N)$ instead of $\mathcal{O}(1)$, bringing the algorithmic complexity up to $\mathcal{O}(N^3)$ (for $N = M$ sequences).

However, the computational cost can be mitigated by using particular gap penalty functions. Here I will present the most common variant, which is known as the *affine* gap penalty. In this model, the cost of a gap of size k is

$$d_k = o + (k-1)e, \tag{4.42}$$

where o and e are the opening and extension penalties, respectively, and $o > e$. To incorporate affine-gap penalties into the Needleman-Wunsch or Smith-Waterman algorithms in an efficient manner, the primary adjustment involves tracking whether consecutive gaps occur in the alignment. This requires the alignment process to be split into three distinct cases: insertions, deletions, and matches/mismatches. Instead of using a single dynamic programming table as in the linear-gap penalty approach, three separate tables are used: $\hat{I}$ for insertions, $\hat{D}$ for deletions, and $\hat{F}$ for the overall score.

In this setup, each entry in the $\hat{I}$ table, denoted by $i_{i,j}$, stores the best alignment score when the last column includes an insertion (*i.e.* a gap in the first sequence), and each entry in the $\hat{D}$ table, $d_{i,j}$, captures the best score for alignments ending in a deletion (*i.e.* a gap in the second sequence). As before, $F_{i,j}$ records the overall score. The recursive update rules of the three tables are

$$I_{i,j} = \max\begin{cases} F_{i-1,j} + o \\ I_{i-1,j} + e \end{cases} \tag{4.43}$$

$$D_{i,j} = \max\begin{cases} F_{i,j-1} + o \\ D_{i,j-1} + e \end{cases} \tag{4.44}$$

$$F_{i,j} = \max\begin{cases} F_{i-1,j-1} + s(S_i, T_j) \\ I_{i,j} \\ D_{i,j}. \end{cases} \tag{4.45}$$

Note that with this algorithm each cell update is $\mathcal{O}(1)$, and therefore the overall complexity remains the same ($\mathcal{O}(NM)$ or $\mathcal{O}(N^2)$ for same-length sequences).

Python implementation
Head over here for Jupyter notebook containing code implementing the Needleman-Wunsch and Smith-Waterman algorithms, with linear and affine gap penalty functions.

Substitution matrices

How is the substitution matrix $s(x, y)$ determined? One possibility is to leverage what we know about the biological processes that underlie mutations. For instance, in DNA the biological reasoning behind the scoring decision can be linked to the probabilities of bases being transcribed incorrectly during polymerization. We already know that of the four nucleotide bases, A and G are purines, while C and T are pyrimidines. Thus, DNA polymerase is much more likely to confuse two purines or two

pyrimidines since they are similar in structure. As a result, a simple improvement over the uniform cost function used above is the following scoring matrix for matches and mismatches:

	A	G	T	C
A	+1	-0.5	-1	-1
G	-0.5	+1	-1	-1
T	-1	-1	+1	-0.5
C	-1	-1	-0.5	+1

a mismatch between like-nucleotides (*e.g.* A and G) is less expensive than one between unlike nucleotides (*e.g.* T and C).

However, we can be more quantitative by taking a probabilistic approach. Here I will describe how the widely-used BLOSUM matrices are built (see also the Henikoff and Henikoff [1992]). We assume that the alignment score reflects the probability that two similar sequences are Section 4.1.3. Thus, given two sequences S and T of the same length[11] for which we have an ungapped alingment, we define two hypotheses:

1. The alignment between the two sequences is due to chance and the sequences are, in fact, unrelated.

2. The alignment is due to common ancestry and the sequences are actually related.

Under the rather strict assumption ("biologically dubious, but mathematically convenient", as aptly put in Eddy [2004]) that pairs of aligned residues are statistically independent of each other, so that the likelihoods associated to the two hypotheses, $P(S,T|R)$ and $P(S,T|U)$, can be expressed as products of probabilities, *viz.*

$$P(S,T|R) \propto \prod_i p_{S_i T_i} \tag{4.46}$$

$$P(S,T|U) \propto \prod_i q_{S_i} q_{T_i} \tag{4.47}$$

where p_{xy} is the likelihood that residues x and y are aligned because correlated, while the product $q_x q_y$ is the likelihood that the two residues are there by chance: their occurrence is unrelated and therefore the likelihood factorises in two terms that account for the average probability of observing those two residues in any protein.

In order to compare two hypotheses, a good score is given by the logarithm of the ratio of their likelihoods (the so-called log-odds score). Therefore, for our case the alignment score is

$$A \equiv \log \frac{P(S,T|R)}{P(S,T|U)}, \tag{4.48}$$

so that, thanks to the properties of logarithms and probabilities, the overall alignment score is the sum of the log-scores of the individual residue pairs. Considering 20 amino acids, there are 400 such log-scores, which form a 20x20 substitution (score) matrix. Each entry of the matrix takes the form

$$s(x,y) = \frac{1}{\lambda} \log \frac{p_{xy}}{q_x q_y}, \tag{4.49}$$

where λ is an overall scaling factor that is used to obtain values that can be rounded off to integers. Note that the previous definition can be formally substantiated (see *e.g.* Karlin and Altschul [1990]).

If $p_{xy} > q_x q_y$ (and therefore $s(x,y)$ is positive), the substitution $x \to y$, and therefore their alignment in homologous sequences, occurs more frequently than would be expected by chance, suggesting that this substitution is favored in evolution. These substitutions are called "conservative". As noted in Eddy [2004], this definition is purely statistical and has nothing directly to do with amino acid

[11] It is clear that in this context S and T are part of larger sequences.

structure or biochemistry. Likewise, substitutions with $p_{xy} < q_x q_y$, and therefore negative values of $s(x, y)$, are termed "nonconservative".

The procedure applied to compute the p_{xy} and q_x values starts with a collection of protein sequences.

1. The sequences are grouped into families based on their evolutionary relatedness. A common source for these families is the BLOCKS database, which contains aligned, ungapped regions of proteins that are highly conserved across members of the family.

2. Within these protein families, highly conserved regions (blocks) are identified. These regions are short segments of amino acid sequences that are believed to be important for the protein's function and are conserved across different species.

3. Once blocks are identified, the sequences within each block are clustered, choosing a threshold value V: two sequences that share more than this percentage of identical amino acids identity are clustered together, and only one representative from each cluster is used to avoid over-representation of very similar sequences.

4. Within each block, the actual counts of amino acid pairs are made.
   - The background frequencies q_x are estimated by computing the overall frequency of each amino acid x in the sequences, not considering any substitutions.
   - To evaluate the p_{xy} terms, for each position in the aligned block, we count how many times one amino acid (say, Alanine) is substituted by another amino acid (say, Glycine) in the aligned positions across the different sequences. For instance, if there is an alignment of 5 sequences, for each single position we make $\binom{5}{2} = 10$ comparisons.

Of course, the final scores depend on V: if V is large, protein blocks that are still rather similar will be considered belonging to different clusters, and therefore compared to each other to derive the scores; the resulting matrix will be more sensitive in identifying closely related sequences but less effective for more distantly related sequences (and vice versa for small values of V). Common values for V are 45%, 62%, and 80%, which yield the matrices BLOSUM45, BLOSUM62, and BLOSUM80, with BLOSUM62 being the de-facto standard (and default) one.

	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
A	0	1	0	4																	A
G	-3	0	-2	0	6																G
P	-3	-1	-1	-1	-2	7															P
D	-3	0	-1	-2	-1	-1	6														D
E	-4	0	-1	-1	-2	-1	2	5													E
Q	-3	0	-1	-1	-2	-1	0	2	5												Q
N	-3	1	0	-2	0	-2	1	0	0	6											N
H	-3	-1	-2	-2	-2	-2	-1	0	0	1	8										H
R	-3	-1	-1	-1	-2	-2	-2	0	1	0	0	5									R
K	-3	0	-1	-1	-2	-1	-1	1	1	0	-1	2	5								K
M	-1	-1	-1	-1	-3	-2	-3	-2	0	-2	-2	-1	-1	5							M
I	-1	-2	-1	-1	-4	-3	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-1	-4	-3	-4	-3	-2	-3	-3	-2	-2	2	2	4					L
V	-1	-2	0	0	-3	-2	-3	-2	-2	-3	-3	-3	-2	1	3	1	4				V
W	-2	-3	-2	-3	-2	-4	-4	-3	-2	-4	-2	-3	-3	-1	-3	-2	-3	11			W
Y	-2	-2	-2	-2	-3	-3	-3	-2	-1	-2	2	-2	-2	-1	-1	-1	-1	2	7		Y
F	-2	-2	-2	-2	-3	-4	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	1	3	6	F
	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	

Figure 4.13: The BLOSUM62 matrix. The amino acids are grouped and coloured based on Margaret Dayhoff's classification. Non-negative values are highlighted. Credits to Ppgardne via Wikimedia Commons.

The BLOSUM62 matrix is shown in Figure 4.13. First of all, note that substitution matrices are symmetric, because the biological process of amino acid substitution is symmetric: as empirically

> The rarer the amino acid is, the more surprising it would be to see two of them align together by chance. In the homologous alignment data that BLOSUM62 was trained on, leucine/leucine (L/L) pairs were in fact more common than tryptophan/tryptophan (W/W) pairs ($p_{LL} = 0.0371$, $p_{WW} = 0.0065$), but tryptophan is a much rarer amino acid ($q_L = 0.099$, $q_W = 0.013$). Run those numbers (with BLOSUM62's original $\lambda = 0.347$) and you get +3.8 for L/L and +10.5 for W/W, which were rounded to +4 and +11.
>
> Eddy [2004]

observed, there is no preferred direction when one amino acid replaces another. Secondly, it is evident that conservative substitutions tend to be those between amino acids that are similar, as made evident by grouping the amino acids according to the classification introduced by Margaret Dayhoff.

Margaret Dayhoff's classification

According to Margaret Dayhoff, one of the pioneers of biophysics and bioinformatics, amino acids can be classified in the following 6 groups, listed in the order with which they are shown in Figure 4.13:

Amino acids	One-letter code	Property	Dayhoff
Cysteine	C	Sulfur polymerization	a
Glycine, Serine, Threonine, Alanine, Proline	G, S, T, A, P	Small	b
Aspartic acid, Glutamic acid, Asparagine, Glutamine	D, E, N, Q	Acid and amide	c
Arginine, Histidine, Lysine	R, H, K	Basic	d
Leucine, Valine, Methionine, Isoleucine	L, V, M, I	Hydrophobic	e
Tyrosine, Phenylalanine, Tryptophan	Y, F, W	Aromatic	f

Why the values of the matrix diagonal, representing the scores of "substituting" one amino acid with itself, are all different?

BLAST

The sheer volume of sequence data generated by modern-day high-throughput sequencing technologies presents a significant challenge. Databases now contain millions of nucleotide and protein sequences, each potentially spanning thousands of characters. When comparing a new sequence against these massive databases, traditional pairwise alignment methods like the ones we just discussed become computationally demanding. Indeed, performing a global or even local alignment between a query sequence and every sequence in a large database can require huge computational resources and time, especially as the number of sequences and their lengths continue to grow exponentially.

The need for a more efficient method resulted in the Altschul et al. [1990], where BLAST (Basic Local Alignment Search Tool), now a critical tool in bioinformatics, was introduced. BLAST operates by finding regions of local similarity between sequences, which is more computationally feasible and faster than aligning entire sequences globally. The algorithm requires a query sequence and a target database. First of all, the query sequence is broken down into smaller fragments (called words or W-mers). For each W-mer, a list of similar words is generated, and only those with a similarity measure

that is higher than a threshold T are retained add added to the final W-mer list. The similarity is evaluated by using a Section 4.1.3 (BLOSUM62 is a common choice). Then, the target database is searched for matches to these words, extending the matches in both directions to find the best local alignments. BLAST assigns scores to these alignments based on the degree of similarity *via* a Smith-Waterman algorithm, with higher scores indicating closer matches.

Warning
The Needleman-Wunsch and Smith-Waterman algorithms always find the optimal solution (*i.e.* the global minimum). By contrast, BLAST uses a heuristic approach, trading off some sensitivity for speed.

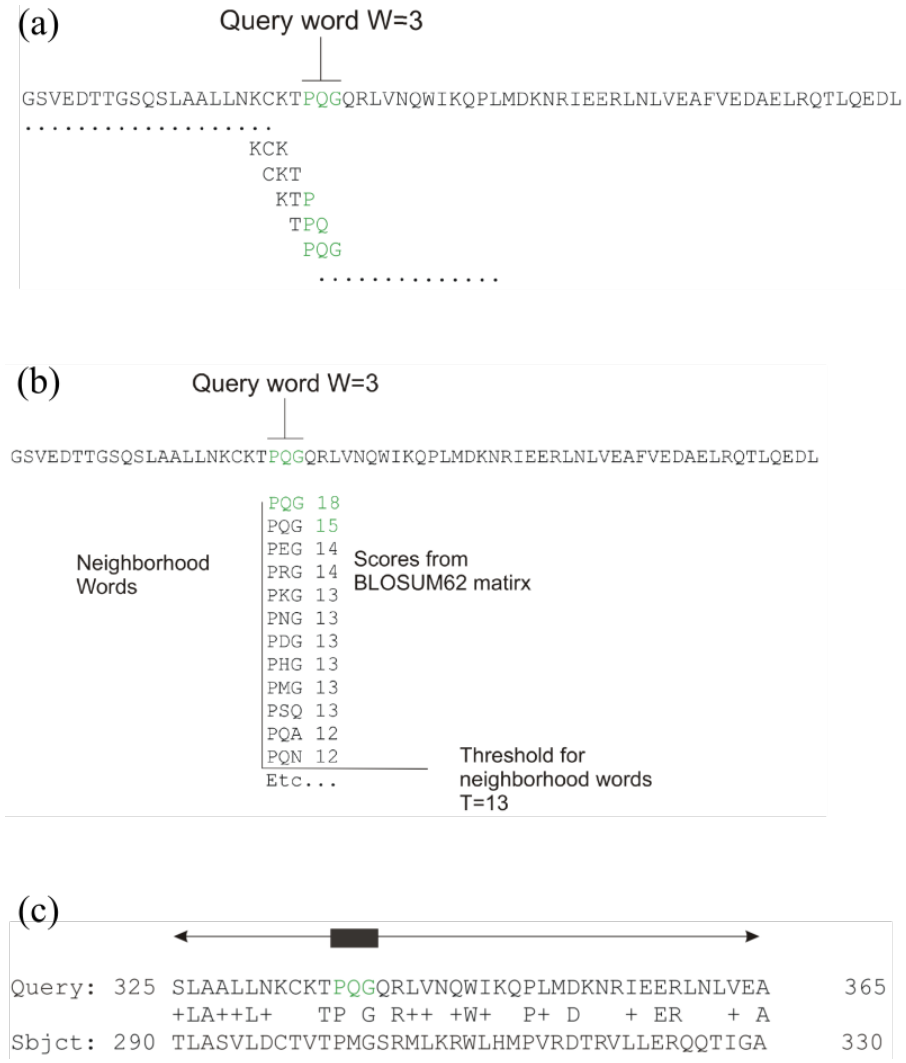

Figure 4.14: The three most important steps of the BLAST algorithm. (a) Generate the initial W-mers ($W = 3$). (b) Make the list of BLAST words using a threshold T. (c) Extend the matches into both directions until the score drops below a predetermined threshold.

The BLAST algorithm can be broken down into the following steps[12]

1. **Optional**: *remove low-complexity region or sequence repeats in the query sequence.* Here "Low-complexity region" means a region of a sequence composed of few kinds of elements. These regions might give high scores that confuse the program to find the actual significant sequences

[12]I took most of this description from Wikipedia, which in my opinion provides one of the thorough explanation of the BLAST algorithm

in the database, so they should be filtered out. The regions will be marked with an X (protein sequences) or N (nucleic acid sequences) and then be ignored by the BLAST program. To filter out the low-complexity regions, the SEG program is used for protein sequences and the program DUST is used for DNA sequences. On the other hand, the program XNU is used to mask off the tandem repeats in protein sequences.

2. *Make a W-letter word list of the query sequence.* The words of length W (W-mers) in the query sequence are listed "sequentially", until the last letter of the query sequence is included. The method is illustrated in TODO. W is usually 3 and 11 for a protein and a DNA sequence, respectively.

3. *List the possible matching words.* A Section 4.1.3 (*e.g.* BLOSUM62) is used to match the words listed in step 2 with all the 20^W W-mers. For example, the score obtained by comparing PQG with PEG and PQA is respectively 15 and 12 with the BLOSUM62 matrix[13]. After that, a neighborhood word score threshold T is used to reduce the number of possible matching words. The words whose scores are greater than the threshold T will remain in the possible matching words list, while those with lower scores will be discarded. For example, if $T = 13$ PEG is kept, but PQA is abandoned.

4. *Organize the remaining high-scoring words into an efficient search tree.* This allows the program to rapidly compare the high-scoring words to the database sequences.

5. *Repeat step 3 to 4 for each W-mer in the query sequence.*

6. *Scan the database sequences for exact matches with the remaining high-scoring words.* The BLAST program scans the database sequences for the remaining high-scoring words, such as PEG. If an exact match is found, this match is used to seed a possible ungapped alignment between the query and database sequences.

7. *Extend the exact matches to high-scoring segment pair (HSP).* The original version of BLAST stretches a longer alignment between the query and the database sequence in the left and right directions, from the position where the exact match occurred. The extension does not stop until the accumulated total score of the HSP begins to decrease. To save more time, a newer version of BLAST, called BLAST2 or gapped BLAST, has been developed. BLAST2 adopts a lower neighborhood word score threshold to maintain the same level of sensitivity for detecting sequence similarity. Therefore, the list of possible matching words list in step 3 becomes longer. Next, exact matched regions that are within distance A from each other on the same diagonal are joined as a longer new region. Finally, the new regions are then extended by the same method as in the original version of BLAST, and the scores for each HSP of the extended regions are created by using a substitution matrix as before.

8. *List all of the HSPs in the database whose score is high enough to be considered.* All the HSPs whose scores are greater than the empirically determined cutoff score S are listed. By examining the distribution of the alignment scores modeled by comparing random sequences, a cutoff score S can be determined such that its value is large enough to guarantee the significance of the remaining HSPs.

9. *Evaluate the significance of the HSP score.* Karlin and Altschul [1990] showed that the distribution of Smith-Waterman local alignment scores between two random sequences is

$$p\left(S \geq x\right) = 1 - \exp\left(-KMNe^{-\lambda x}\right), \tag{4.50}$$

where M and N are the length of the query and database sequences[14], and the statistical parameters λ and K depend upon the substitution matrix, gap penalties, and sequence composition

[13] For DNA words, common parameters are +5/-4 or +2/-3 for matches and mismatches.

[14] It is possible to derive expressions that use effective rather than true sequence lengths, to compensate for edge effects (an alignment that starts near the end of the query or database sequence is likely not to have enough sequence to build an optimal alignment).

(the letter frequencies) and are estimated by fitting the distribution of the ungapped local alignment scores of the query sequence and of a lot of (globally or locally) shuffled versions of a database sequence. Note that the validity of this distribution, known as the Gumbel extreme value distribution (EVD), has not been proven for local alignments containing gaps yet, but there is strong evidence that it works also for those cases. The expect score E of a database match is the number of times that an unrelated database sequence would obtain a score S higher than x by chance. The expectation E obtained in a search for a database of total length N

$$E = KMNe^{-\lambda S} \tag{4.51}$$

This expectation or expect value E (often called E-score, E-value or e-value) assessing the significance of the HSP score for ungapped local alignment is reported in the BLAST results. The relation above is different if individual HSPs are combined, such as when producing gapped alignments (described below), due to the variation of the statistical parameters.

10. *Make two or more HSP regions into a longer alignment.* Sometimes, two or more HSP regions in one database sequence can be made into a longer alignment. This provides additional evidence of the relation between the query and database sequence. There are two methods, the Poisson method and the sum-of-scores method, to compare the significance of the newly combined HSP regions. Suppose that there are two combined HSP regions with the pairs of scores $(65, 40)$ and $(52, 45)$, respectively. The Poisson method gives more significance to the set with the maximal lower score $(45 > 40)$. However, the sum-of-scores method prefers the first set, because $65 + 40 = 105$ is greater than $52 + 45 = 97$. The original BLAST uses the Poisson method; BLAST2 uses the sum-of scores method.

11. *Show the gapped Smith-Waterman local alignments of the query and each of the matched database sequences.* The original BLAST algorithm only generates ungapped alignments including the initially found HSPs individually, even when there is more than one HSP found in one database sequence. By contrast, BLAST2 produces a single alignment with gaps that can include all of the initially found HSP regions. Note that the computation of the score and its corresponding E-value involves use of adequate gap penalties.

The E-value is the single most important parameter to rate the quality of the alignments reported by BLAST. For instance, if $E = 10$ for a particular alignment with score S, it means that there are 10 alignments with score $\geq S$ that can happen by chance between any query sequence and the database used for the search. Therefore, this particular alignment is most likely not very significant. By contrast, values much smaller than 1 (*e.g* 10^{-3} or even smaller) are likely to signal that the sequences are homologous. On most webserver, it is possible to filter out all matches that have an E-value larger than some threshold. For instance, on the NCBI webserver, the "expect threshold" defaults to 0.05.

Note that the sequence database used to search for matches is preprocessed first, which increases further the overall computational efficiency. With BLAST, the algorithmic complexity of searching the database for a sequence of length N is only $\mathcal{O}(N)$. An online webserver[15] to run BLAST can be found here.

Using BLAST
A short tutorial on how to use BLAST from the command line can be found here.

Multiple sequence alignment

Pairwise sequence alignment suffers from some issues that are intrinsic to it, and becomes glaring when trying to align sequences that are distantly related. In particular, all pairwise methods depend in some way or another on a number of parameters (scoring matrix, gap penalties, *etc.*), and it is

[15] There are many!

hard to tell what is the "best" alignment if the method finds multiple alignments with the same score. Moreover, a pairwise alignment is not necessarily informative about the evolutionary relationship (and therefore about possibly conserved amino acids or whole motifs) of the sequences that are compared.

In order to overcome these issues, a number of multiple sequence alignment (MSA) methods have been developed. MSA extends the concept of pairwise protein alignment to simultaneously align three or more sequences, providing a broader view of evolutionary relationships, structural conservation, and functional regions among a group of proteins. Aligning multiple sequences make it possible to identify conserved amino acids that may be critical for protein function or stability and therefore are biologically significant. This, in turn, can help to predict structural features, functional motifs, and evolutionary patterns across species, providing a fundamental tool to build phylogenetic trees or inform the classification of proteins into families.

In principle, MSA can be carried out with the dynamic programming algorithms we already introduced. For instance, the recursive rule of the Needleman-Wunsch algorithm for globally aligning three sequences S, T, and U, which extends Eq. (4.40), is

$$F_{i,j,k} = \max \begin{cases} F_{i-1,j,k} + s(S_i, -, -) \\ F_{i,j-1,k} + s(-, T_j, -) \\ F_{i,j-1,k} + s(-, -, U_k) \\ F_{i-1,j-1,k} + s(S_i, T_j, -) \\ F_{i-1,j,k-1} + s(S_i, -, U_k) \\ F_{i,j-1,k-1} + s(-, T_j, U_k) \\ F_{i-1,j-1,k-1} + s(S_i, T_j, U_k), \end{cases} \tag{4.52}$$

where the dynamic programming "table" is now three-dimensional, and $s = s(a, b, c)$ is the cost function of aligning a, b, and c, which can be either residues or gaps. It is straightforward to see that the dimensionality of the table grows linearly with the number of sequences M, and therefore the algorithmic complexity is $\mathcal{O}(N^M)$. The exponential dependence on M makes this approach practically unfeasible when working with real-world examples.

Unfortunately, better-performing exact methods, *i.e.* methods that provide the global optimal solution by construction, do not exist (yet). As a result, we have to rely on heuristic methods, which only find local minima. One commonly used approach for multiple sequence alignment is called progressive multiple alignment. Here the prerequisite is that we need to know the evolutionary tree, often called the *guide tree*, that connects the sequences we wish to align, which is usually built by using some low-resolution similarity measure, often based on (global) pairwise alignment. Then, we start by aligning the two most closely related sequences in a pairwise fashion, creating what is known as the seed alignment. Next, we align the third closest sequence to this seed, replacing the previous alignment with the new one. This process continues, sequentially adding and aligning each sequence based on its proximity in the tree, until we reach the final alignment. Note that this is done using a "once a gap, always a gap" rule: gaps in an alignment are not modified during subsequence alignments. These methods have a computational complexity of $\mathcal{O}(M^2)$, which makes it possible to align thousands of sequences.

One of the most commonly used tools to perform MSAs is Sievers et al. [2011], which is a command-line tool, but it is also available as a webserver. Clustal Omega builds the guide tree using an efficient algorithm (adapted from Blackshields et al. [2010]) which has an algorithmic complexity of $\mathcal{O}(M \log M)$, making it possible to generate multiple alignments of hundreds of thousands sequences. An example of MSA is shown in Figure 4.15.

4.1.4 Threading

Tip

Most of the text of this part has been adapted from Torda [2005] and Finkelstein and Ptitsyn [2016].

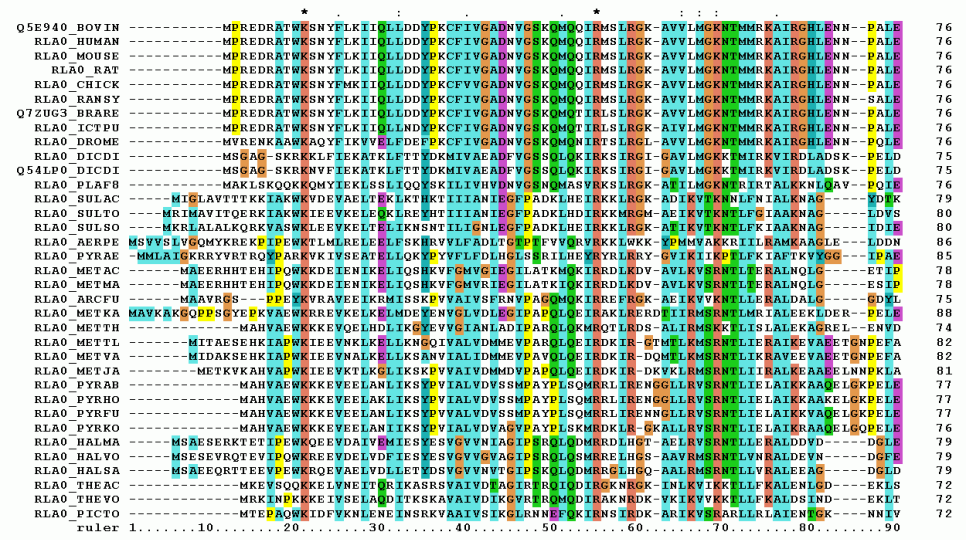

Figure 4.15: Representation of a protein multiple sequence alignment produced with ClustalW (which has been superseded by Clustal Omega). The sequences are instances of the acidic ribosomal protein P0 homolog (L10E) encoded by the Rplp0 gene from multiple organisms. The protein sequences were obtained from SwissProt searching with the gene name. Only the first 90 positions of the alignment are displayed. The colours represent the amino acid conservation according to the properties and distribution of amino acid frequencies in each column. Note the two completely conserved residues arginine (R) and lysine (K) marked with an asterisk at the top of the alignment. Credits to Miguel Andrade via Wikipedia Commons.

If pair alignment tools find that a given query sequence is found to share more than 30% of its sequence with another, then it is common to think that a reasonable model for that sequence can be built. By contrast, an alignment yielding a similarity of 20% - 25% could be purely coincidental. In reality, things are more complicated, as it has been shown that proteins with rather high sequence identity could be very different from a structural point of view (Brenner et al. [1998]). This and other results show that a single quantity, such as sequence identity, is not enough to determine the 3D similarity between two proteins, and more numbers (such as the length of the chain or of the well-aligned regions) are required to build reliable models. For instance, it makes sense that for a short 50-residue protein a 40% sequence identity would be required to generate a good match, while 25% may be enough for 250 residues. However, these numbers have a purely statistical value.

A better approach is to go beyond pairwise alignments by using sequence database searching programs such as BLAST which, as we have seen, provide E-values or similar quantities that estimate the reliability of a sequence match by looking at it in the context of the whole library of sequence scores. In addition, more sophisticated BLAST versions (such as PSI-BLAST) make it possible to obtain good matches with less than 20% sequence identity.

Thanks to these tools, and to the ever-growing number of sequences stored in databases, “simple” database searches are often enough to build a model of the query sequence out of a reliable homolog of known structure. However, if no matches are found, or if an independent confirmation is required, we need methods that do not rely on sequence homology. In this case we recast the problem of protein structure prediction as a problem of choice of the 3D structure best fitting the given sequence among many other possible folds. However, what can be the source of “possible” structures?

While an a priori classification is sometimes possible (see, *e.g.*, Murzin and Finkelstein [1988]) or Ptitsyn et al. [1985]), a more practical answer is the Protein Data Bank (PDB) where all the solved and publicly available 3D structures are collected. However, using structures stored in the PDB turns a “prediction” problem into a “recognition” one: a fold cannot be recognized if the PDB does not contain an already solved analog. This limits the power of recognition. Nevertheless, there is an important advantage associated to this procedure: if the protein fold is recognized among the PDB-stored structures, one can hope to recognize also the most interesting feature of the protein, namely

its function—by analogy with that of an already studied protein.

Certainly, not all protein folding patterns have been collected in PDB yet; however, it most likely already includes the majority of all the folding patterns existing in nature. This hope, substantiated by Chothia [1992], is based on the fact that the folds found in newly solved protein structures turn out to be similar to already known folds more and more frequently. Extrapolation shows that perhaps about 1500–2000 folding patterns of protein domains exist in genomes, and we currently know more than half of them (including the majority of the most common folds).

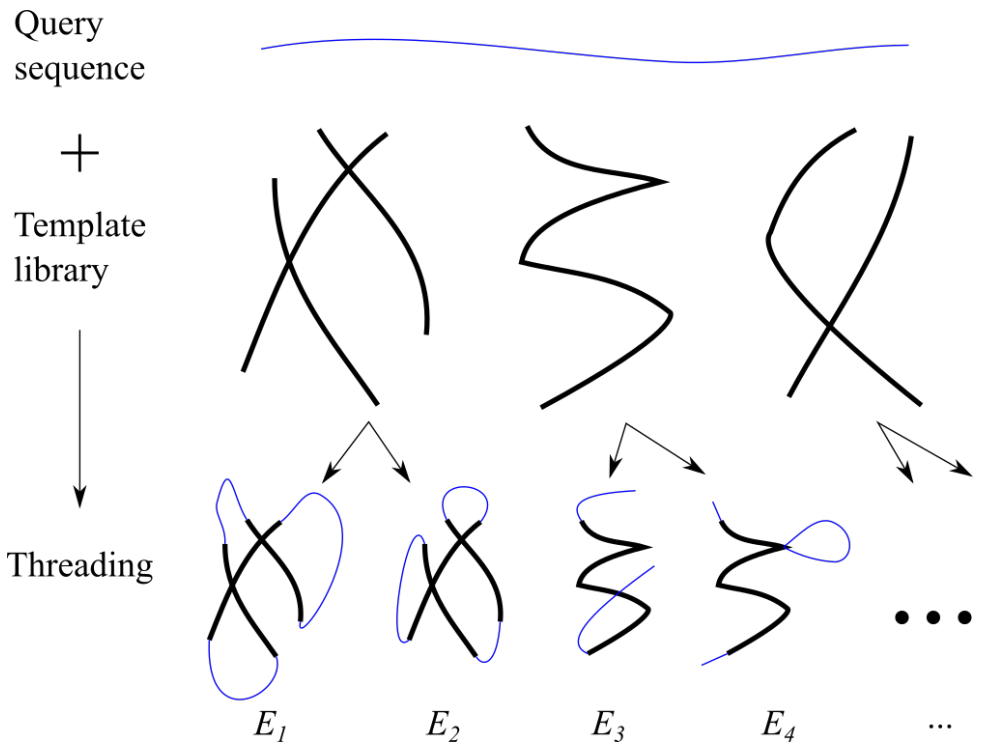

Figure 4.16: The basic idea of threading: a sequence is "threaded" through templates extracted from a database of known folds (*e.g.* the PDB), and the resulting structure is assigned an energy. The structure having the lowest energy is taken as the optimal candidate.

To recognize the fold of a chain having no visible homology with already solved proteins, one can use various superpositions of the chain in question onto all examined (taken from an a priori classification or from PDB) 3D folds in search of the lowest-energy chain-with-fold alignment, as sketched in Figure 4.16. This is called the *threading method*. When a chain is aligned with the given fold, it is threaded onto the fold's backbone until its energy (or rather, free energy) is minimized, including both local interactions and interactions between remote chain regions. The threading alignment allows "gaps" in the chain and in the fold's backbone (the latter are often allowed for irregular backbone regions only). Many different algorithms have been proposed for finding the correct threading of a sequence onto a structure, though many make use of dynamic programming in some form. Indeed, in principle, threading is similar to a homology search; the difference is that only sequences are aligned in a homology search, while threading aligns a "new" sequence with "old" folds.

Being physics-inspired, threading can be a powerful tool for structural biologists and biophysicists. It uses mostly statistics-derived pseudopotentials rather than actual energies, which are based on the contact statistics between amino acids as found in known protein structures[16].

As with any other structure-prediction model, there are some issues with threading, the main ones being:

1. The conformations of the gapped regions remain unknown, together with all interactions in these regions.

[16]Similar, in spirit, to the way Section 4.1.3 are computed.

2. Even the conformations of the aligned regions and their interactions are known only approximately, since the alignment does not include side-chain conformations (which may differ in "new" and "old" structures). Estimates show that threading takes into account, at best, half of the interactions operating in the protein chain, while the other half remain unknown to us. Thus, again, the protein structure is to be judged from only a part of the interactions occurring in this structure. Therefore, this can only be a probabilistic judgment.

3. The number of possible alignments is enormous (remember Levinthal's paradox?), which makes it very hard to sort out all possible threading alignments and single out the best one (or ones). Here there are many methods that can be employed, one of which being the statistical mechanics of one-dimensional systems, and the related dynamic programming techniques.

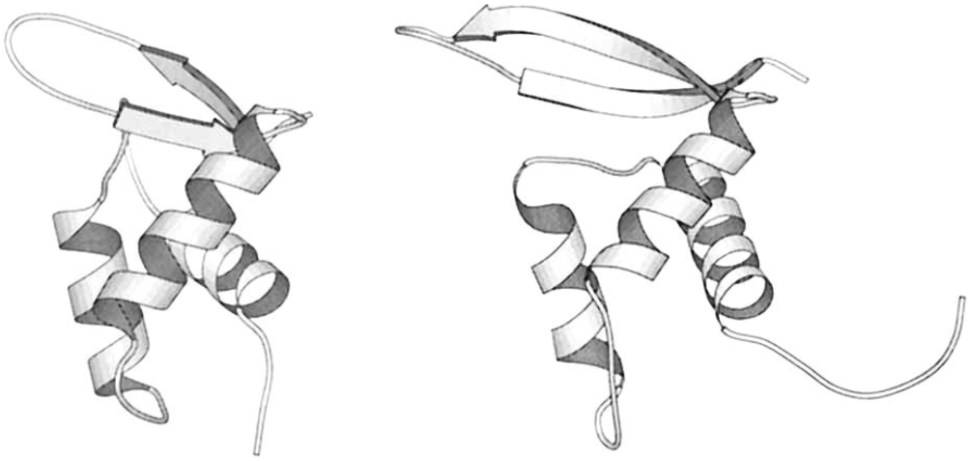

Figure 4.17: The 3D structures of (left) chicken histone H5 and (right) replication terminating protein (rtp). Note that the C-terminal helix of rtp is not shown, since it has no analoug in the histone. The root mean-squared deviation between 65 equivalent C^{α} positions in the two structures is 2.4 Å. Taken from Flöckner et al. [1995].

As an example, Finkelstein and Ptitsyn [2016] shows the structure prediction done for the replication terminating protein (rtp) by threading. Having threaded the rtp sequence onto all PDB-stored folds, Flöckner et al. [1995] used threading with statistics-based pseudopotentials to show that the rtp fold must be similar to that of H5 histone. This a priori recognition, which is shown in Figure 4.17, turned out to be correct.

However, it also turned out that the alignment provided by threading deviates from the true alignment obtained from superposed 3D structures of rtp and H5 histone. On the one hand, this shows that even a rather inaccurate picture of residue-to-residue contacts can lead to an approximately correct structure prediction. On the other hand, this shows once again that all the mentioned flaws (insufficiently precise interaction potentials, uncertainty in conformations of nonaligned regions, of side chains, *etc.*) make it possible to single out only a more-or-less narrow set of plausible folds rather than one unique correct fold. Indeed, the set of the "most plausible" folds can be singled out quite reliably, but it still remains unclear which of these is the best. The native structure is a member of the set of the plausible ones, it is more or less close to the most plausible (predicted) fold, but this is all that one can actually say even in the very best case.

Threading methods became a tool for a tentative recognition of protein folds from their sequences. The advantage of these methods is that they formulate a recipe: do this, this and this, and you will obtain a few plausible folds, one of which has a fairly high chance of being correct.

4.1.5 AlphaFold

Tip
The most technical part of this section has been taken/adapted from a great blog post written by Carlos Outeiral.

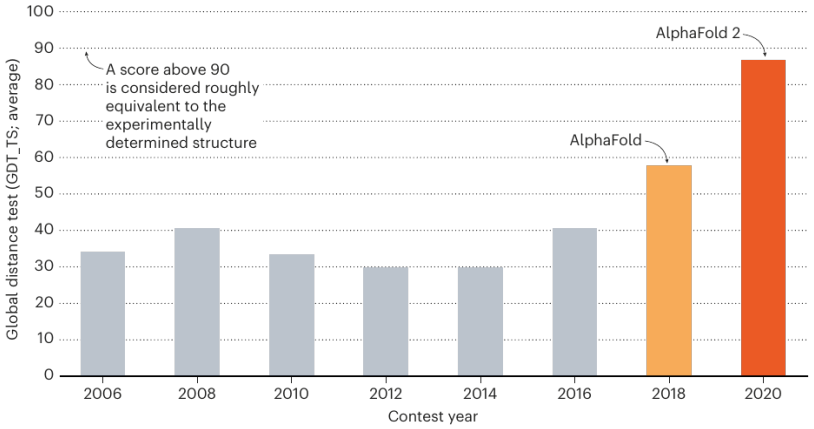

Figure 4.18: The performance of CASP winners in the eight contests up to 2020. Taken from Callaway [2020].

The CASP (Critical Assessment of protein Structure Prediction) contest is a biennial competition established to evaluate the performance of computational methods for protein structure prediction. Since its inception in 1994, CASP has become a benchmark for assessing progress in the field, gathering researchers from across the globe to tackle one of the most complex challenges in computational biology. Participants receive amino acid sequences of proteins for which the 3D structure remains experimentally unknown to them but has been solved by researchers outside the competition. They then attempt to predict these structures using their algorithms, which are later evaluated against the experimentally determined ones. Figure 4.18 shows the score of the winners of the last 8 contests. As can be seen in the figure, in 2020 there was a breakthrough that made it possible to predict the 3D structure of proteins with a precision close to the experimental one.

The breakthrough has been possible thanks to AlphaFold2 (which I will refer to simply as "AlphaFold" from now on), which is a deep learning model developed by DeepMind, presented in Jumper et al. [2021], designed to predict protein structures with remarkable accuracy. The work behind AlphaFold has been awarded half of the 2024 Nobel Prize in Chemistry. The model's architecture is based on neural networks, and integrates several advanced techniques from machine learning and structural biology to predict the 3D structure of a protein out of its sequence (*i.e.*, the *folding problem*). I will briefly describe the internal architecture of Alphafold, and then show how to use it (in a slightly improved version called ColabFold, presented in Mirdita et al. [2022]).

Input and preprocessing

The overall architecture, as presented in the original paper, is shown in Figure 4.19. The first part, which handles the user input and is highlighted in red in Figure 4.19, is a preprocessing pipeline that can be carried out independently of the rest as done, for instance, in Section 4.1.5. First of all, the AlphaFold system uses the input amino acid sequence to query several databases of protein sequences, and constructs a Section 4.1.3 in order to determine the parts of the sequence that are more likely to mutate. The underlying idea is that, if two amino acids (possibly far apart along the sequence) are in close spatial contact, mutations in one of them will be closely followed by mutations of the other, so that the overall 3D structure is preserved. To make an extreme (and somewhat unrealistic) example, suppose we have a protein where an amino acid with negative charge (say, glutamate) is spatially close to an amino acid with positive charge (say, lysine), although they are both far away in the sequence. Most likely, the resulting electrostatic interaction stabilises the structure of the protein. If for evolutionary reasons the first AA mutates into a positively charged amino acid, the second AA

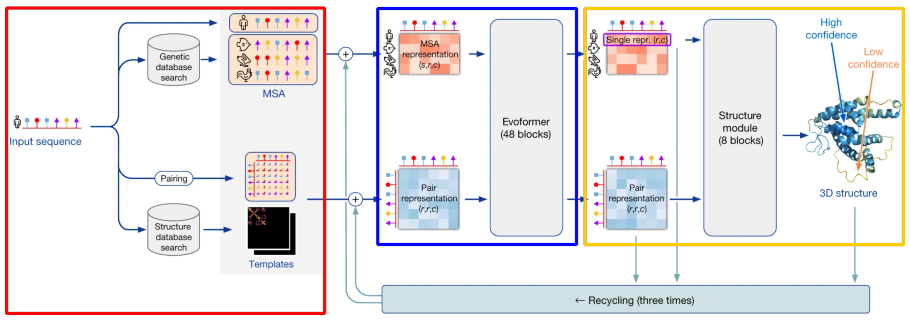

Figure 4.19: The architecture of AlphaFold. Arrows show how the information flows among the components. The input (preprocessing) module, the evoform and the structure module are highlighted in red, blue and yellow, respectively. Adapted from Jumper et al. [2021].

will be under evolutionary pressure to mutate into a negatively charged amino acid in order to preserve the electrostatic attraction and therefore the contribution to the stability of the folded protein. The MSA makes it possible to detect this sort of correlations.

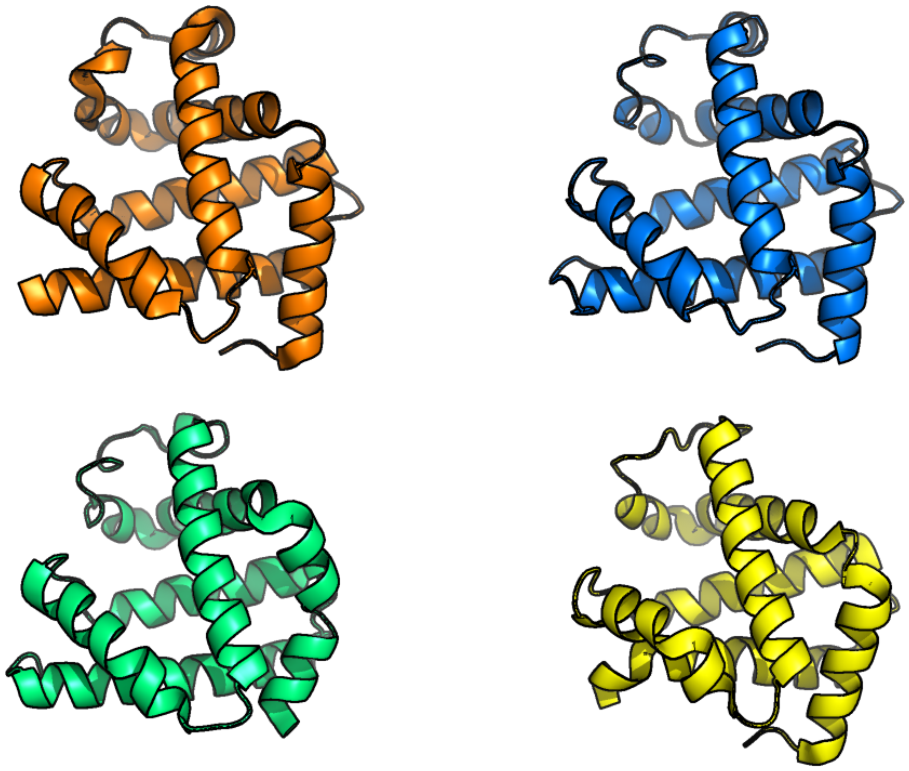

Figure 4.20: The 3D structure of human myoglobin (top left), african elephant myoglobin (top right, 80% sequence identity), blackfin tuna myoglobin (bottom right, 45% sequence identity) and pigeon myoglobin (bottom left, 25% sequence identity). Credits to Carlos Outeiral.

AlphaFold also tries to identify proteins that may have a similar structure to the input ("templates"), and constructs an initial representation of the structure called the "pair representation" which is, in essence, a model of which amino acids are likely to be in contact with each other. Finding templates follows a completely different, but closely related principle: proteins mutate and evolve, but their structures tend to remain similar despite the changes. In Figure 4.20, for example, I display the structure of four different myoglobin proteins, corresponding to different organisms. You can appreciate that they all look pretty much the same, but if you were to look at the sequences, you would find enormous differences. The protein on the bottom right, for example, only has ~25% amino acids in common with the protein on the top left.

The Evoformer

For years, the use of MSAs to detect correlations between amino acids relied on statistical analysis, but its limited accuracy required substantial improvements. Early breakthroughs in coevolutionary analysis helped identify biases in the data and correct them using more advanced statistical techniques. These methods contributed to better but still imperfect predictions of protein structures.

In AlphaFold, the information about the MSA and the templates is jointly analysed by a special type of transformer, which is a deep learning method introduced in Vaswani et al. [2017] that uses a mechanism called attention, allowing the model to focus on different parts of the input data, assigning varying importance to specific regions. This architecture became popular in tasks like natural language processing (NLP) due to its ability to model long-range dependencies in sequences. Instead of processing data sequentially, transformers look at all parts of the input simultaneously, which significantly speeds up learning and makes it more efficient. In bioinformatics, this approach has been adapted to analyze MSAs, improving how relationships between residues are understood. AlphaFold takes the application of transformers a step further with its Evoformer architecture (highlighted in blue in Figure 4.19), which processes both the MSA and the templates, allowing information to flow back and forth between the sequence and the structural representations. Unlike previous deep learning approaches, where geometric proximity was inferred only at the end, AlphaFold continuously updates and refines its structural predictions throughout the process.

Evoformer uses a two-tower transformer model, with one tower dedicated to the MSA (called the MSA transformer) and the other focusing on pairwise residue interactions (pair representation). At each iteration, these representations exchange information, improving both the understanding of the sequence and the predicted pair representation. This iterative refinement, performed 48 times in the published model, allows AlphaFold to adjust its predictions based on both the sequence data and evolving structural hypotheses.

For example, if the MSA transformer identifies a correlation between two residues, it hypothesizes that they are spatially close. This information is passed to the pair representation, which updates the structural model. The pair transformer can then detect further correlations between other residues, perhaps close to the first pair, refining the structure hypothesis. This back-and-forth exchange continues until the model reaches a confident structural prediction.

Note that this step is particularly memory intensive, since the construction of the attention matrix in trasformers has a quadratic memory cost, which can bring down to their knees even powerful GPUs, making it hard to use AlphaFold to predict the structure or long proteins on consumer hardware.

The structure module

The information generated by the Evoformer is taken to the the structure module (highlighted in yellow in Figure 4.19). The idea behind the structure module is conceptually very simple, but its implementation is fairly complex.

The structure module considers the protein as a "residue gas": every AA is modelled as a triangle, representing the three atoms of the backbone. These triangles float around in space, and are moved by the network to form the structure. The transformations are parametrised as 4x4 affine matrices that handle rotations and translations. At the beginning of the structure module, all of the residues are placed at the origin of coordinates. At every step of the iterative process, AlphaFold produces a set of affine matrices that displace and rotate the residues in space. This representation does not reflect any physical or geometrical assumptions, and as a result the network has a tendency to generate structural violations. This is particularly visible in the supplementary videos of Jumper et al. [2021], that display some deeply unphysical snapshots (see Figure 4.21 for a striking example).

The secret sauce of the structure module is a new flavour of attention devised specifically for working with three-dimensional structures which is based on the very simple fact that the L2-norm of a vector is invariant with respect to translations and rotations[17]. Why is this invariance such a big deal? You may understand this as a form of data augmentation: if the model knows that any possible

[17] An explanation of this Invariant Point Attention (IPA) is well-beyond my expertise. If you are interested, I refer to section 1.8 of the Supplementary Information of the original paper.

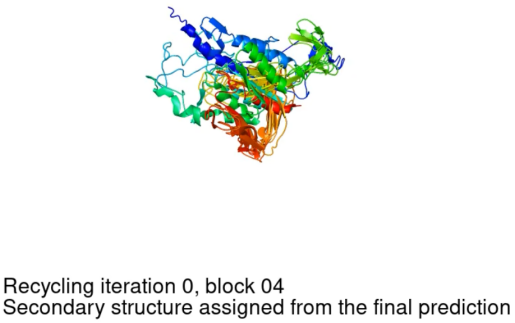

Figure 4.21: The evolving prediction of the CASP14 multi-domain target (863 residues) T1091. Individual domains' structure is determined early, while the domain packing evolves throughout the network. The network explores unphysical configurations throughout the process, resulting in the long "strings" that appear during the evolution. Supplementary Video 4 of Jumper et al. [2021], which can be downloaded here.

rotation of translation of the data will lead to the same answer, it will need a lot less data to pull it away from wrong models and will therefore be able to learn much more.

Finally, it is also worth noting that the Structure Module also generates a model of the side chains. To simplify the system, their positions are parametrised by a list of dihedral angles χ_1, χ_2, *etc.* (depending on the AA), which are predicted in their normal form by the network, and implemented with standard geometric subroutines.

Recycling and loss function

AlphaFold leverages a so-called "recycling mechanism": after generating a final structure, it takes all the information (*i.e.* the MSA and pair representations, as well as the predicted structure) and passes it back to the beginning of the Evoformer blocks. In the original paper the whole pipeline is executed three times.

The quality of the training and of the final output of AlphaFold are determined by a specialized loss function called Frame Aligned Point Error (FAPE), a modified version of RMSD, to align atomic positions, which helps prevent incorrect protein chirality. However, this is only part of a more complex loss function, which is a weighted sum of various auxiliary losses. These include losses calculated from multiple iterations of the structure module and a distogram loss, which compares predicted 2D distance matrices with the true structure.

Another interesting aspect is MSA masking, inspired by self-supervised learning models, where some symbols in the MSA are masked and the model is asked to predict them. An additional trick employed is the so-called self-distillation: in this approach, they took a model trained exclusively on the PDB, and predicted the structures of ~300k diverse protein sequences. They then retrained the full model, incorporating a small random sample of these structures at every training cycle. The claim is that this operation allows the model to leverage the large amount of unlabelled data available in protein sequence repositories.

Output

The single most useful output of AlphaFold is the 3D coordinates of each atom of the final predicted structure. A very important metric that is reported by the model is the predicted local-distance difference test (pLDDT), which provides a per-residue confidence score on a scale from 0 to 100, indicating how certain the model is about the position of each amino acid in the predicted 3D structure.

A higher pLDDT score suggests greater accuracy, with scores above 90 being highly reliable, while scores below 70 indicate regions with lower confidence, possibly due to disorder or flexibility. Finally, AlphaFold also outputs a distogram, which is a histogram of pairwise distances.

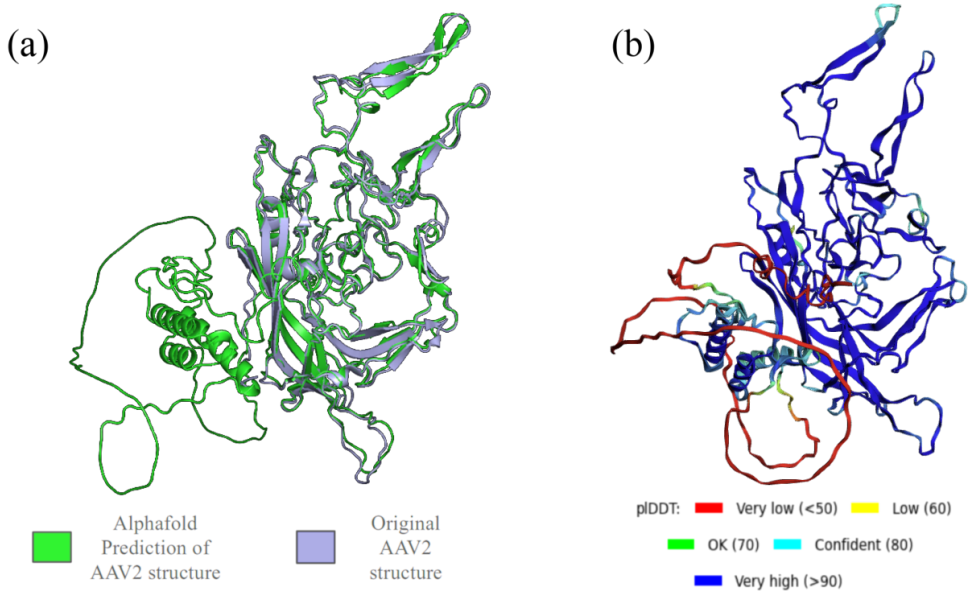

Figure 4.22: (a) The superposition between the experimental and Alphafold-predicted structures of the VP1 protein composing the capsid of a AAV2 virus. (b) The Alphafold-predicted structure, coloured according to the value of the pLDDT of each residue. Courtesy of Mouna Ouattara.

Figure 4.22 shows an example where Alphafold was used to predict the structure of a protein that makes up part of a viral capsid. Since the experimental structure is available, it is possible to superpose it to the predicted structure, which shows that a part of the latter has been hallucinated, *i.e.* made up, by Alphafold. Interestingly, a large part of the hallucinated portion is predicted to be a coil and has a very low pLDDT, but part of it (the two helices) is structured and has a good pLDDT.

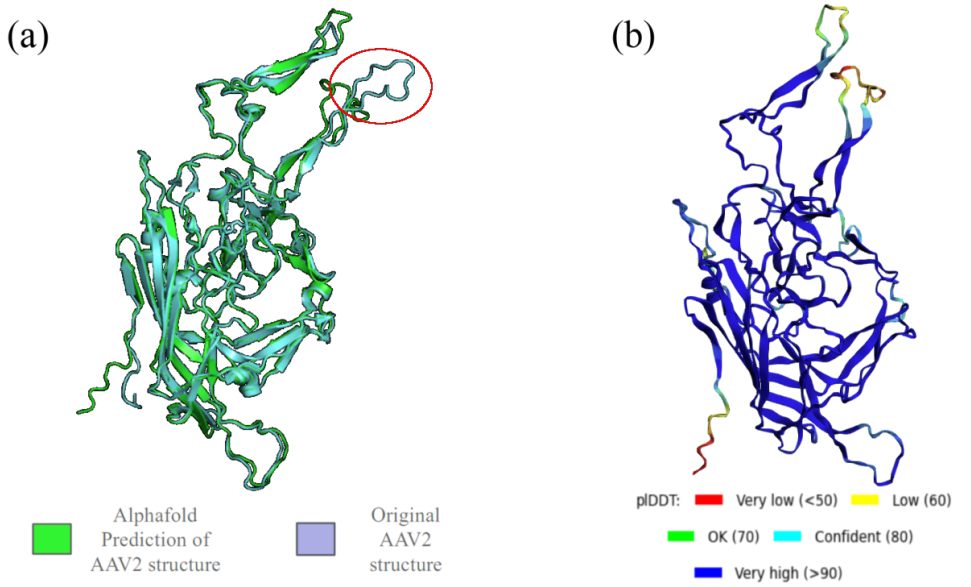

Figure 4.23: Same as Figure 4.22, but for a mutant serotype (7m8, see Dalkara et al. [2013]), which differs from the wildtype for a 10-amino acid peptide insertion in the VR-VIII loop, highlighted with a red ellipse in panel (a). Courtesy of Mouna Ouattara.

By contrast, Figure 4.23 shows the same comparisons for a closely related protein, which is obtained by adding a short peptide (10 amino acid long) to a loop that is far apart from the part of the protein

that Alphafold hallucinates when predicting the structure of the wildtype protein. Nevertheless, in this case the predicted structure is very close to the experimental one.

Using AlphaFold through ColabFold

ColabFold, presented in Mirdita et al. [2022], is an optimized implementation of AlphaFold that runs on Google Colab, providing a more accessible and user-friendly way to perform protein structure predictions. Unlike the "pure" AlphaFold, which can be downloaded here and requires extensive computational resources and a setup that may be challenging for individual users, ColabFold makes AlphaFold's capabilities more widely available by leveraging Google's cloud infrastructure. The ColabFold version is streamlined for easier use and has a faster input preprocessing, with the option to use precomputed databases for sequence searching, that significantly reduces both runtime and storage requirements. ColabFold provides an easy way to customise the most common options of the model, allowing users to adjust parameters, access multi-sequence alignments, and use additional input sequences to improve prediction accuracy. The code behind the notebook is open source (see here), and the notebook itself can be accessed here.

Exercise

Use ColabFold to predict the structure of the human ATPase GET3 (UniProt ID: O43681). First of all, download the experimental PDB structure by clicking on the "Structure" item in the left toolbar, and then download the "8CQZ" entry, which contains the structure of a single chain[18].

To predict the structure, copy the sequence ("Sequence" in the left toolbar, and then "Copy sequence"), paste it into the "query_sequence" field of the ColabFold notebook, and then click on "Runtime" → "Run all". At the end of the procedure (which should take 20-30 minutes) a zip file will be downloaded.

Before opening it, familiarise with all the plots in the output part of the notebook. A few notes:

- By default ColabFold will generate 5 models and rank them according to their average pLDDT.
- The sequence coverage shows the MSA "at a glance". The plot shows the number of homolog sequences that overlap, at least partially, with the query.
- The pLDDT plots provide an immediate visual estimation of the quality of the predictions.

Now unzip the output archive and compare the experimental and predicted structures. In VMD this can be done in this way:

- Open the two files: `vmd -m file1.pdb file2.pdb`
- Change the representations to show the secondary structure of the two proteins
  - Open "Graphics → Representations..."
  - Choose "Molecule" as Coloring Method and "NewCartoon" as Drawing Method
  - In the top menu select the other file and repeat the previous step
- Open the RMSD Tool: "Extensions → Analysis → RMSD Calculator"
- In the top-left text field write "residue 1 to 340", since the two structures do not have the same number of residues
- Click "Align" to superimpose one structure on top of the other. The different colours should make it possible to spot the differences at a glance
- You can play with the text field to see what happens if you choose to align different subsections of the two structures.

4.2 Nucleic acids

As for proteins, the structure of nucleic acids can be studied at different levels. The primary structure is the unidimensional list of nucleotides, which can be obtained fairly easily through sequencing. In DNA and RNA, the secondary structure is the list of base pairs that are formed between different

parts of the strand(s), while the 3D structure is the description of how the polymer bends and twists. Here, contrary, to proteins, attractive interactions beyond base pairing (which includes stacking) are not very strong, and since nucleic acids are hydrophilic, there is no strong tendency of the nucleotides to tightly pack together. As a result, the free energy that stabilises a molecule's conformation mostly comes from the secondary (rather than tertiary, as in proteins) structure. Therefore, there are many useful things that we can learn about a certain RNA or DNA system by just knowing its secondary structure.

Here I will present methods and algorithms that have been developed for single chains of RNA, but they are mostly valid for (or extendable to) multi-stranded and DNA systems. Formally, given a chain of length N, a secondary structure can be represented as a graph where each nucleotide i is a vertex, and each base pair (i, j) is an edge connecting two vertices. The connectivity can be represented by an adjancency matrix $\hat{A}$, where each element of the matrix $A_{i,j}$ is equal to 1 if i and j are linked (either through the backbone or by hydrogen bonds) and 0 otherwise. In the case of the single chain, nucleotides are ordered and the backbone is continuous, thus $A_{i,i+1} = 1$. Moreover, since each nucleotide can be involved in most one base pair, for each i there is at most one j different from $i - 1$ and $i + 1$ for which $A_{i,j} = 1$.

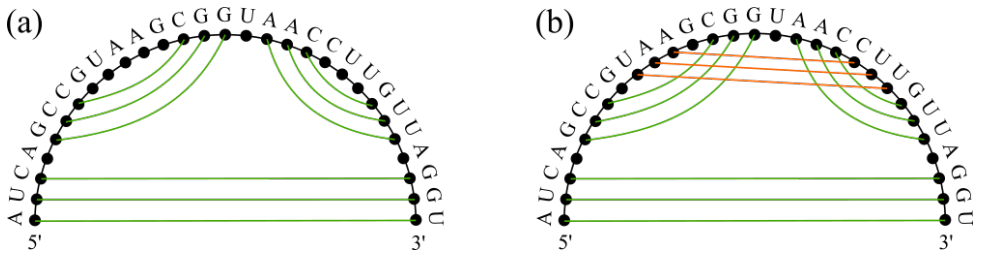

Figure 4.24: Possible secondary structures of an RNA strand of sequence `AUCAGCCGUAAGCGGUAACCUUGUUAGGU`, represented as graphs. The points are the nucleotides (*i.e.* the graph's vertices), while lines that connect them are the graph's edges. The black and coloured lines are the exterior and interior edges, respectively. In (a) no lines cross, while in (b) there are intersections between orange and green lines: a pseudoknot is present.

Note that secondary structure prediction becomes much easier if we assume that no pseudoknots can form. This is a strong assumption, but it is very convenient from an algorithmic point of view. In the graph/matrix representation, this condition is fulfilled if, for any pair of base pairs (i, j) and (k, l),

- if $i < k < j$ then $i < l < j$;
- if $k < i < l$ then $k < j < l$.

In the graph representation, pseudoknots are present if some edges intersect. Figure 4.24 shows the secondary structure (with and without knots) of a short RNA strand, represented as a graph.

Now we want to predict the secondary structure of the RNA, given its sequence. Since here we are dealing with one-dimensional sequences, I hope it will not be surprising to know that dynamic programming can be leveraged to find a solution to this problem, provided that

1. we use a scoring scheme whereby the free-energy contribution that each base pair (or, more generally, "local" secondary structure motif) has on the overall stability of the molecule is additive;

2. we assume that pseudoknots cannot form, so that the RNA can be split into two smaller ones which are independent.

Indeed, under these conditions the solution to the full problem can be built by solving subproblems, which can be done efficiently by applying dynamic programming. The next two sections will describe two algorithms that can be used to obtain the optimal structure, *i.e.* the one with the minimum free-energy (MFE).

4.2.1 Nussinov's algorithm

The first approach I will describe has been introduced by Nussinov et al. [1978]. To keep it simple, I will use a version of the algorithm in which the (free) energy of an unpaired base is 0, and that of a base pair is a fixed value, *i.e.* -1, regardless of the type of nucleotides involved.

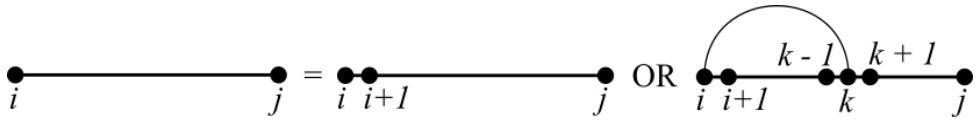

Figure 4.25: The graph representation of Nussinov's recursive relation. Given a subsequence S_{ij}, i is either unpaired (first term after the equal sign), or is paired to some other nucleotide $i < k \leq j$.

The intuition behind Nussinov's algorithm, shown pictorially in Figure 4.25, is the following: for any subsequence $[i, j]$, $S_{i,j}$, the i-th base can either remain unpaired or be paired with some k-th base where $i < k \leq j$. If the i-th base is unpaired, the (free) energy of $S_{i,j}$, denoted as $F_{i,j}$, simply reduces to the energy of the subsequence $S_{i+1,j}$, or $F_{i+1,j}$. This forms the first term of the Nussinov recurrence relation.

On the other hand, if the i-th base pairs with the k-th base, then $F_{i,j}$ consists of the energy contribution of this pairing, denoted as $s_{i,k}$, plus the energies of the two resulting subproblems: the subsequence $S_{i+1,k-1}$, represented by $F_{i+1,k-1}$, and the subsequence $S_{k+1,j}$, represented by $F_{k+1,j}$. Finding the optimal k that minimizes this value gives us the second term in the Nussinov recurrence relation.

> **The original algorithm**
> If you read the original paper, you will notice that the algorithm was conceived a bit differently. For instance, the optimal secondary structure was defined as the structure maximising the number of base pairs rather than the one minimising the free energy. Here I choose to do the latter to make the algorithm a bit more general, since in principle the energy contribution of a base pair can be made to depend on the types of the two nucleotides.

Therefore, the optimal energy of the subsequence $S_{i,j}$ is the minimum of the energy when the i-th base is unpaired and the energy when it is paired with the optimal k-th base, which can be written explicitly as follows:

$$F_{i,j} = \min \left\{ F_{i+1,j}, \min_k \{ F_{i+1,k-1} + F_{k+1,j} + s_{i,k} \} \right\}. \tag{4.53}$$

In this case the dynamic programming table $\hat{F}$ is of size $N \times N$, and is initialised so that its diagonal entries are set to zero, since a nucleotide cannot bind to itself. The other entries are set iteratively by starting from the bottom-right entry, where $i = N-2$ and $j = N-1$, so that the matrix is progressively filled up from left to right and bottom to top. The final score, representing the optimal solution for the entire sequence, is found in the upper right corner of the matrix, corresponding to the subsequence $S_{0,N-1}$. Since each (i, j) entry requires an $\mathcal{O}(N)$ minimisation, and there are $\mathcal{O}(N^2)$ entries, the total algorithmic complexity is $\mathcal{O}(N^3)$.

As always with dynamic programming methods, the secondary structure is obtained by using a traceback matrix $\hat{K}$, which is initialised during the fill-in phase. In particular, given the optimal secondary structure of the subsequence $S_{i,j}$, $K_{ij} = 0$ if i is unpaired, while $K_{ij} = k$, where k is the nucleotide that is paired to i, otherwise. A possible recursive traceback function is

```
DEFINE K as the traceback matrix K
DEFINE P as the list of base pairs

FUNCTION traceback(i, j)
    k = K(i, j)
```

```
IF i > j
    RETURN
IF k == 0
    traceback(i + 1, j)
ELSE
    ADD (i, k) TO P
    traceback(i + 1, k - 1)
    traceback(k + 1, j)
```

The recursion starts from the end, *i.e.* with `i = 0` and `j = N - 1`, and at the end of the procedure the list of base pairs is stored in `P`.

Note that the complexity of the traceback algorithm is $O(N^2)$, since we move through a $N \times N$ matrix and perform a $\mathcal{O}(1)$ operation on each entry we visit.

The Nussinov's algorithm is quite simplistic and comes with several limitations. In its naive implementation, the algorithm does not account for some important factors in RNA folding. Most notably, it does not take into account that, as we already know, stacking interactions between neighboring base pairs are crucial for RNA stability, at least as much as hydrogen bonding. To address this and other limitations, it is fundamental to incorporate biophysical factors into the prediction model. One improvement is to assign energies to structural elements rather than to individual base pairs, so that the total energy of the RNA structure becomes the sum of the energies of these substructures.

Visualising the table, tracebacks and structures
Head over here to try the algorithm. The webpage makes it possible to visualise the dynamic programming table, the traceback procedure and the optimal structure(s).

4.2.2 Zuker's algorithm

Most of the issues mentioned above can be overcome with the algorithm devised by Zuker and Stiegler [1981]. The algorithm is based on the idea that RNA folds into a structure that minimizes its free energy, taking into account not only base pairing, but also other factors. In particular, the algorithm assumes that the RNA folding process can be described by an energy model where base pairs, stacking interactions, and loops (hairpin loops, bulge loops, interior loops, and multi-branch loops) contribute to the total free energy of the molecule. The Lu et al. [2006] energy model is commonly used in implementations, which provides empirical parameters for these contributions.

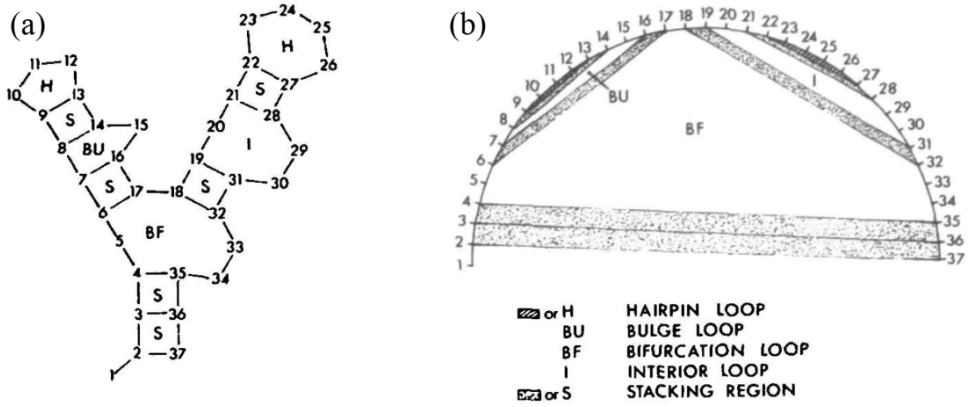

Figure 4.26: Two schematic representations of the secondary structure of a simple RNA molecule. (a) The conventional representation. (b) The same structure, represented as an undirected graph with exterior and interior edges. The legend in the bottom right applies to both panels. Note that here "bifurcation loop" is used in place of multibranched loop. Taken from Zuker and Stiegler [1981].

The key idea can be fully appreciated by representing an RNA secondary structure as an undirected graph, drawn as a semicircle with chords connecting some edges (see also Figure 4.24). In the following

description I borrow heavily from the original paper, which still is (after more than 40 years) one of the best resources on the subject[19]. Here are a few definitions:

- the N nucleotides are the vertices of the graph;
- the $N-1$ arcs that connect each pair of i and $i+1$ nucleotides are the *exterior edges* and represent the backbone bonds;
- chords connecting vertices represent base pairs, and therefore connect only complementary nucleotides (C-G, A-U or G-U); these chords are called *interior edges* and have the following properties:
  1. interior edges do not touch, since each nucleotide can be involved in no more than one base pair;
  2. interior edges do not cross, since pseudoknots are not allowed.
- a *face* is a region of the graph that is bounded on all sides by edges.

These definitions are pictorially represented in Figure 4.26. The difference with Nussinov's algorithm is that the (free) energy of a structure is associated not with the base pairs, but with the regions that are bounded by the bonds, *i.e.* with the graph faces associated to that structure. Indeed, the possible secondary structure motifs can be classified according to how they can be represented in terms of faces:

- A face bounded by a single interior edge is a hairpin loop, whose length is the number of exterior edges (the number of nucleotides composing the loop is this number minus one);
- A face bounded by two interior edges is further classified into three groups:
  1. if the interior edges are separated by two single exterior edges on both sides, the face is a stacking region (or stacking loop);
  2. if the face is bounded by a single exterior edge on one side, and multiple exterior edges on the other, the face is a bulge loop;
  3. if there are multiple exterior edges on both sides, the face is an interior loop;
- A face that has k ($k > 2$) interior edges is a k-multiloop (or a multibranched loop of order k).

The energy of a given structure is the sum of the energies associated to each of its faces, $E_P = \sum_F E_F$. This is a powerful way of setting the statistical weight of each motif by using experimentally-determined values: for instance, since hairpins with loops shorter than 3 are not possible, we can assign the energy $E_F = \infty$ to hairpin loops having fewer than four exterior edges. Any other specific effect (*e.g.* complicated sequence-dependent effect) can be naturally incorporated in this framework.

Thanks to this decomposition, if we know (or can estimate) the energy contributions associated to each secondary structure motif, we can compute the total energy of any given structure. However, enumerating all possible structures to find the one with the lowest (free) energy would be computationally impossible even for rather small sequences. The problem can be overcome by using dynamic programming, with an approach similar (but not equal) to that of Nussinov.

As before, $S_{i,j}$ is the subsequence $[i, j]$. For each $S_{i,j}$ we define

- $W_{i,j}$ as the minimum free energy of the subsequence;
- $V_{i,j}$ as the minimum free energy of all structures formed by $S_{i,j}$, in which S_i and S_j are paired with each other; if S_i and S_j cannot pair with each other, $V_{i,j} = \infty$.

The constraint on the minimum length of hairpin loops can be enforced by setting $W_{i,j} = V_{i,j} = \infty$ if $j - i \leq 4$. By constrast, if $j - i = d > 4$, $W_{i,j}$ and $V_{i,j}$ can be written in terms of $W_{k,l}$ and $V_{k,l}$ for various pairs (k, l) for which $l - k < d$, *i.e.* in terms of the minimum energies of smaller substructures. The recursive relations for the two matrices, $\hat{V}$ and $\hat{W}$, both of size $N \times N$, can be found by considering the possible structures formed by $S_{i,j}$.

We start with $\hat{V}$. First of all, note that this matrix considers only those substructures that have S_i and S_j paired or, in other words, substructures whose graph representation has the (i, j) interior edge. The presence of this additional edge means that the resulting optimal substructure also has an

[19] It is very common for papers that contributed greatly to a field to be also very well thought out and written. This is only partially true for the paper in question (as mentioned below).

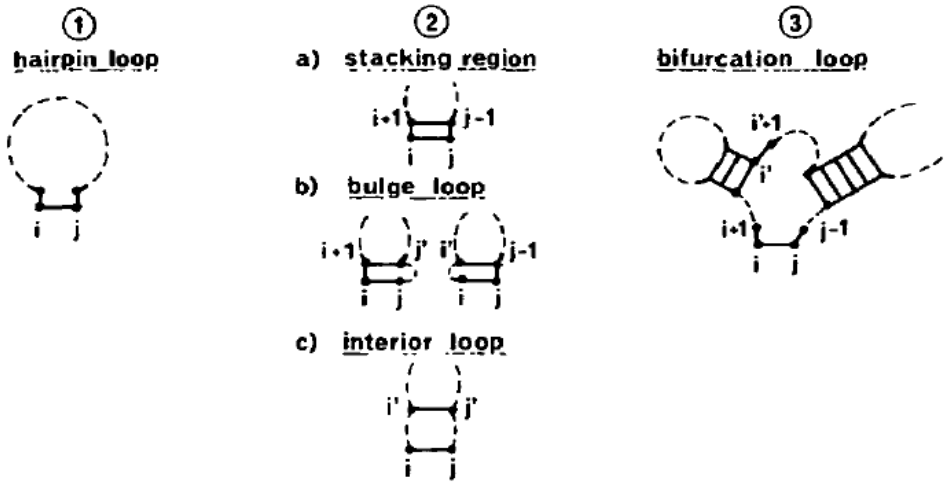

Figure 4.27: Possible substructures for the subsequence S_{ij}, constrained to the presence of an (i, j) edge. Adapted from Zuker and Stiegler [1981].

additional face compared to any other substructure that has been already evaluated. In particular, as shown in Figure 4.27,

1. (i, j) can close a hairpin, thus contributing with an energy $F_H(i, j)$;
2. (i, j) can close a face containing exactly two interior edges, with the other edge being (k, l), with $i < k < l < j$. The face contributes an energy $F_L(i, j, k, l)$ that depends on the face which, as before, can be of three types:
   (a) $k = i + 1$ and $l = j - 1$: the face is a stacking region;
   (b) $k = i + 1$ or $l = j - 1$ (but not both): the face is a bulge region;
   (c) $k > i + 1$ and $l < j - 1$: the face is an interior loop.
3. (i, j) can close a face containing k interior edges, with the other $k - 1$ edges being $(i_1, j_1), \ldots, (i_{k-1}, j_{k-1})$. This is a k-multiloop, and its energy contribution is $F_M(i, j, i_1, j_1, \ldots, i_{k-1}, j_{k-1})$.

In all these cases, the F_H, F_L and F_M penalty functions are model parameters, and can be estimated by experiment (see the section on Section 3.2.2). Of course, there are multiple substructures in which the (i, j) edge is present. We make sure to select the one with the lowest free energy by first obtaining the single optimal substructures that fall into cases 2. and 3., and then to select the optimal substructure among the three possibilities listed above. This procedure translates to the following recursive relation:

$$V_{i,j} = \min \begin{cases} F_H(i,j) \\ F_S(i,j) + V_{i+1,j-1} \\ \min_{i<k<l<j}\{F_L(i,j,k,l) + V_{k,l}\} \\ \min\limits_{k,i<i_1<j_1<\ldots<i_{k-1}<j_{k-1}<j} \{F_M(i,j,i_1,j_1,\ldots,i_{k-1},j_{k-1}) + \\ \qquad\qquad + \sum\limits_{1\le l\le k} V_{i_l,j_l}\}, \end{cases} \tag{4.54}$$

where $F_S(i, j) \equiv F_L(i, j, i + 1, j - 1)$ is the energy associated to the possibility that the new face is a stacking region, for which we do not have to carry out any minimisation, since we already know that $k = i + 1$ and $l = j - 1$.

A simplified model

Strangely enough, the Zuker and Stiegler [1981] contains an excellent description of the simplified model where the energy cost of multibranched loops is zero. However, the authors explicitly state that they use a version of the algorithm where this cost is different from zero, but do not describe it.

In the simplified model described in Zuker and Stiegler [1981], the contribution due to a multibranched loop is not given by a specific energy function F_M, but it is written in terms of the energy of two substructures (which is why Zuker *et al* use the term "bifurcation loop"). In particular, the contribution of a k-multiloop closed by the (i, j) edge is written as

$$W_{i+1,k} + W_{k+1,j-1} \tag{4.55}$$

for some nucleotide k satisfying $i + 1 < k < j - 1$. This is a strong approximation, since it does not assign any penalty to the presence of a multiloop *per se*: its contribution is only due to the energy of the parts that compose it. The updated recursion for $\hat{V}$ thus reads

$$V_{i,j} = \min \begin{cases} F_H(i,j) \\ F_S(i,j) + V_{i+1,j-1} \\ \min_{i<h<k<j}\{F_L(i,j,h,k) + V_{h,k}\} \\ \min_{i+1<k<j-2}\{W_{i+1,k} + W_{k+1,j-1}\} \end{cases} \tag{4.56}$$

This is the version of the model I have implemented.

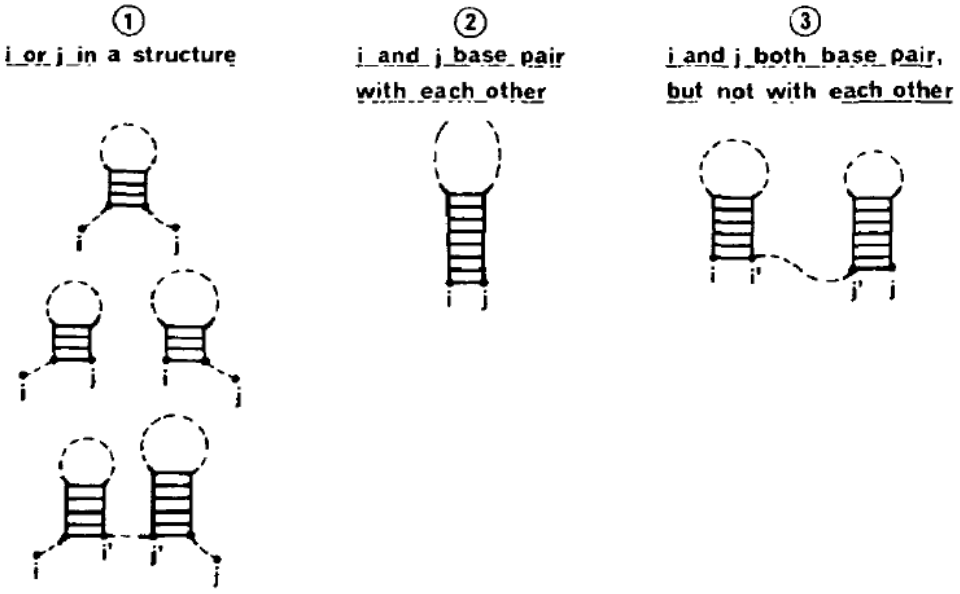

Figure 4.28: Possible substructures for the subsequence S_{ij}, this time with no constraints. Adapted from Zuker and Stiegler [1981].

Now we can write down the recursive relation for $\hat{W}$. There are once again three possibilities, shown schematically in Figure 4.28:

1. S_i and S_j (one of them or both) do not form base pairs in the substructure. Since dangling ends do not contribute to the overall energy, $W_{i,j} = W_{i+1,j}$ or $W_{i,j} = W_{i,j-1}$.
2. S_i and S_j form a base pair. In this case $W_{i,j} = V_{i,j}$, which we have already computed.
3. S_i and S_j base pair, but not with each other. In other words, there exist interior edges (i, k) and (l, j) (with $i < k < l < j$). In this case there is an "open bifurcation", as the energy can be written in terms of two substructures: $W(i, j) = W(i, k) + W(k + 1, j) = W(i, l - 1) + W(l, j)$.

The last case requires a minimisation over k (or, equivalently l), which translates to the following recursive relation:

$$W_{i,j} = \min \begin{cases} W_{i+1,j} \\ W_{i,j-1} \\ V_{i,j} \\ \min_{i<k<j}\{W_{i,k} + W_{k+1,j}\}. \end{cases} \tag{4.57}$$

A simpler (but less obvious) recursive relation for $\hat{W}$

If the the $W_{i,j}$ entries with $j - i \leq 4$ are initialised to zero rather than ∞, Eq. (4.57) can be rewritten as

$$W_{i,j} = \min \begin{cases} W_{i,j-1} \\ \min_{i\leq k<j-4}\{W_{i,k-1} + V_{k,j}\}. \end{cases} \tag{4.58}$$

This can be proven by rewriting the last case of Eq. (4.57) so that the first term of the min argument is $W_{i,k-1}$ rather than $W_{i,k}$, and the minimisation is carried out over the same values of k of Eq. (4.58), yielding

$$\min_{i\leq k<j-4}\{W_{i,k-1} + W_{k,j}\}. \tag{4.59}$$

First of all, note that, by construction, j is always paired with k, so that $W_{k,j} = V_{k,j}$. Moreover, the new relation contains two additional terms compared to the original one: $W_{i+1,j}$ and $(W_{i,i-1} + V_{i,j}) = V_{i,j}$, which are equal to the first and third case of Eq. (4.57), respectively.

The recursive relations for $\hat{V}$ and $\hat{W}$ can be used in a dynamic programming code to obtain the optimal secondary structure of the full sequence S. Since there is the constraint on the minimal loop length, after initialisation we start filling the matrices from the $(N-5, N-1)$ entry, and continue from bottom to top and from left to right. As in Nussinov's algorithm, the energy of the optimal structure of the entire sequence will be stored at the end of the fill-in phase in the $W_{N-1,0}$ entry, and the optimal secondary structure itself can be retrieved by using traceback matrices.

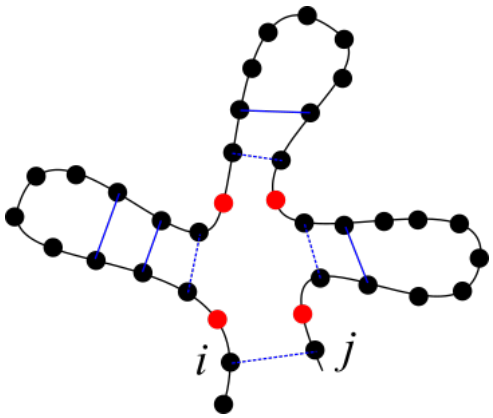

Figure 4.29: An example of a multibranched loop with $k = 4$ and $k' = 4$. Interior edges are coloured in blue, while the interior edges that define the multiloop are drawn with dashed lines. Unpaired nucleotides within the loop are coloured in red.

What is the algorithmic complexity of the method? In the last case of Eq. (4.54), we minimise on the order of the multiloop, and on all its interior edges. This operation has an exponential complexity, and therefore completely kills the computational efficiency. There exist several approximations that can be leveraged to obtain a polynomial complexity. The most extreme one is given by Eq. (4.56),

which completely neglects the energetic penalty of the multiloop. A more realistic approximation is the following:

$$F_M(i, j, i_1, j_1, \ldots, i_{k-1}, j_{k-1}) \approx a + bk + ck', \tag{4.60}$$

where k is the order of the loop, k' is the number of unpaired bases within the loop (see Figure 4.29), and a, b and c are parameters, to be estimated experimentally. This approach makes use of another auxiliary matrix, $\hat{M}$, whose generic entry $M_{i,j}$ is the optimal energy of S_{ij}, with the constraint that S_i and S_j are part of a multiloop. The recursive relation takes into account the possibilities that i and/or j are unpaired, which contributes c, are paired with each other, which contributes b, or are part of two multi-loop substructures, and therefore

$$M_{i,j} = \min \begin{cases} M_{i,j-1} + c \\ M_{i+1,j} + c \\ V_{i,j} + b \\ \min_{i<k<j}\{M_{i,k} + M_{k+1,j}\}. \end{cases} \tag{4.61}$$

We can now rewrite the last case of Eq. (4.54) by making use of the new matrix, obtaining

$$V_{i,j} = \min \begin{cases} F_H(i, j) \\ F_S(i, j) + V_{i+1,j-1} \\ \min_{i<k<l<j}\{F_L(i, j, k, l) + V_{k,l}\} \\ \min_{i<k<j}\{M_{i+1,k} + M_{k+1,j-1} + a\}. \end{cases} \tag{4.62}$$

The algorithmic complexity of the four cases are $\mathcal{O}(1)$, $\mathcal{O}(1)$, $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$. Since there are $\sim N^2$ entries, the overall algorithmic complexity is $\mathcal{O}(N^4)$, and the required storage space is $\mathcal{O}(N^2)$, since only $N \times N$ matrices are used. The computational efficiency can be improved by limiting the size of a bulge or interior loop to some value (often taken to be 30), which brings the complexity of the third case of Eq. (4.62) down to $\mathcal{O}(1)$, and the overall algorithmic complexity down to $\mathcal{O}(N^3)$.

A very nice (but not necessarily easy to read) open-source implementation of the Zuker's algorithm can be found here.

4.2.3 Beyond the MFE: the McCaskill algorithm[20]

The Nussinov's and Zuker's algorithm find the optimal secondary structure of an RNA strand, defined as the structure that minimise the overall (free) energy. However, the conformations of macromolecules that are in thermal equilibrium with their environment are not fixed: in principle *any* allowed structure can be visited, given enough time (remember the concept of ergodicity and phase space?). In particular, if we define $\mathcal{P}$ as the *structural ensemble*, *i.e.* the ensemble of allowed structures, the probability that a macromolecule has a specific secondary structure P is given by

$$p(P) = \frac{e^{-\beta E_P}}{\sum_{P' \in \mathcal{P}} e^{-\beta E_{P'}}} \equiv \frac{e^{-\beta E_P}}{Q}, \tag{4.63}$$

where E_P is the energy of secondary structure P, and Q is the partition function. We now look for a recursive relation to compute the partition function of a sequence S. Given a subsequence S_{ij}, i is either unpaired, or it is paired with the k-th nucleotide, where $i < k \leq j$. For the first case, the partition function is that of the smaller subsequence $[i + 1, j]$, $Q_{i+1,j}$. For the latter case, the (i, k) edge splits the subsequence in two independent subproblems, and the overall partition function is given by the product of their partition functions, $Q_{k+1,j}$ and $Q_{i+1,k-1}q_{i,k}$, where $q_{i,k} \equiv e^{-\beta \Delta G_{i,k}}$ is the statistical weight of the base pair formed by S_k and S_j. Look at Figure 4.25 and you will realise that it is the same splitting! The total partition function will be a sum over all these cases, *viz*

[20] The derivations in this section are taken from Raden et al. [2018].

$$Q_{i,j} = Q_{i+1,j} + \sum_{i<k\leq j} Q_{k+1,j} Q_{i+1,k-1} q_{i,k}. \tag{4.64}$$

This relation makes it possible to write down a dynamic programming code that fill the $\hat{Q}$ matrix with a complexity $\mathcal{O}(N^3)$. As before, the partition function of the sequence is found in the $Q_{0,N-1}$ entry.

The connection with Nussinov's algorithm

Here I'm implicitly using a simplified version of the McCaskill algorithm, where the overall energy is only due to base pairing. If you look closely, you will notice that there is a direct connection between Eq. (4.64) and Eq. (4.53), which is summarised in the following table (adapted from the Appendix C of Finkelstein and Ptitsyn [2016]):

MFE	Partition function
Energy E	Boltzmann factor $e^{-\beta E}$
Minimisation	Summation
Summation	Multiplication

, it is possible to turn any MFE algorithm into an algorithm that can compute partition functions, provided that there is no overlap between the cases appearing in the recursive relations of the former, *i.e.* that some structures are counted more than once. Indeed, ambiguous decompositions (as they are called) are not a problem when carrying out a minimisation, but they are when doing a summation...

How can we use the information contained in the partition function to obtain the equilibrium (structural) ensemble of the strand? The most granular information we can compute is the probability that two nucleotides i and j are base paired, $p(i,j)$. However, in order to do so we first have to introduce the auxiliary matrix $\hat{Q}^{\text{bp}}$, whose generic entry $Q^{\text{bp}}_{i,j}$ stores the partition function of the subsequence $S_{i,j}$, with the constraint that nucleotides i and j form the (i,j) base pair. Its recursive relation is

$$Q^{\text{bp}}_{i,j} = \begin{cases} Q_{i+1,j-1} q_{i,j} & \text{if } i \text{ and } j \text{ are complementary} \\ 0 & \text{otherwise} \end{cases} \tag{4.65}$$

so that the recursive relation of $\hat{Q}$ (Eq. (4.64)) becomes

$$Q_{i,j} = Q_{i+1,j} + \sum_{i<k\leq j} Q_{k+1,j} Q^{\text{bp}}_{i,k}. \tag{4.66}$$

Note that with this formalism it is easier to employ more realistic energy functions by adding contributions (due to interior and multiloops, for instance) to Eq. (4.65), in analogy with the strategy devised by Zuker. See Bernhart et al. [2011] for additional details.

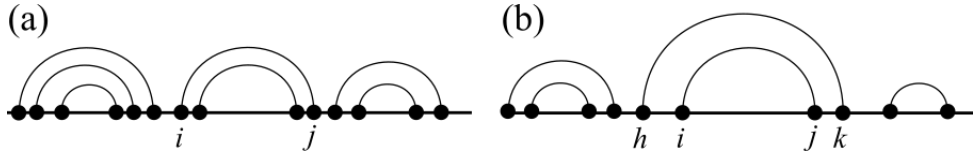

Figure 4.30: A graph where the (i,j) base pair is (a) external and (b) enclosed by an (h,k) base pair.

We can write down the probability $p(i,j)$ recursively, as the sum of two contributions, shown graphically in Figure 4.30:

1. The probability that the (i,j) edge exists and it is external, *i.e.* that there are no edges (k,l) for which $k < i < j < l$. In this case the (i,j) base pair split the graph into three independent

parts, so that the total probability is just

$$p_{\text{ext}}(i,j) = \frac{Q_{0,i-1} Q^{\text{bp}}_{i,j} Q_{j+1,N-1}}{Q_{0,N-1}}. \tag{4.67}$$

2. The probability that the (i,j) edge exists and it is enclosed by other base pairs. This is a sum over all the possible enclosing base pairs (h,k), with $h < i < j < k$, of the probability that the (h,k) edge exists, $p(h,k)$, times the probability that the (i,j) edge is the exterior edge of the $S_{h+1,k-1}$ subsequence, which is given by[21]

$$\frac{Q_{h+1,i-1} Q^{\text{bp}}_{i,j} Q_{j+1,k-1}}{Q_{h+1,k-1}}. \tag{4.68}$$

All in all, this contribution amounts to

$$P_{\text{int}}(i,j) = \sum_{h<i<j<k} p(h,k) \frac{Q_{h+1,i-1} Q^{\text{bp}}_{i,j} Q_{j+1,k-1}}{Q_{h+1,k-1}}. \tag{4.69}$$

The total probability that i and j form a base pair is thus

$$p(i,j) = \frac{Q_{0,i-1} Q^{\text{bp}}_{i,j} Q_{j+1,N-1}}{Q_{0,N-1}} + \sum_{h<i<j<k} p(h,k) \frac{Q_{h+1,i-1} Q^{\text{bp}}_{i,j} Q_{j+1,k-1}}{Q_{h+1,k-1}}. \tag{4.70}$$

Note that the algorithmic complexity of this computation is $\mathcal{O}(N^4)$, since there is the double sum over h and k, which has to be carried out for every i,j pair. However, the complexity can be brought down to $\mathcal{O}(N^3)$ by introducing another auxiliary matrix (see *e.g.* Bernhart et al. [2011]).

Python implementation
Head over here for a Jupyter notebook containing code implementing the algorithms discussed here. My version of Zuker's algorithm is highly simplified so that the code is as short and readable as possible.

4.2.4 Third-party software

You can find several codes online implementing the algorithms we just discussed. However, many of them are a bit rough around the edges and/or incomplete (or wrong). Some interesting and well-done open source implementations are available in different languages. seqfold implements the Zuker algorithm for both DNA and RNA, and it is a rather readable code written in Python. At this website there are several pages where you can test "RNA algorithms" (several of which we already introduced). Interestingly, the authors (from Freiburg University) also released an excellent Raden et al. [2018] detailing their implementations, as well as their code.

The current state-of-the-art software codes for RNA folding are RNAfold, bundled in the ViennaRNA package, and mFold, which was originally developed by M. Zuker and is been since merged with other packages in the UNAFold software. Both softwares can be used online as webservers, but can also downloaded and installed locally. However, ViennaRNA can be downloaded freely, while UNAFold requires a license (although older versions of mFold can be freely downloaded here).

The best online webserver is surely the one based on ViennaRNA, which can be found here. It provides tools to do almost everything related to RNA secondary structures, and I invite you to use it (and possibly read the accompanying papers). The tool that implements the algorithms of Zuker and McCaskill is RNAfold. Try it, and compare its predictions to those of the simple codes we discussed in class.

[21] Eq. (10) of Raden et al. [2018] is written slightly differently, since they use the fact that $Q^{\text{bp}}_{h,k} = Q_{h+1,k-1} q_{hk}$

NUPACK

An online webserver, which is also a Python library, that has a slightly different scope but uses algorithms that are very much inline with those we just studied, is NUPACK. The most recent NUPACK version is described Fornace et al. [2022]. NUPACK has been free-to-use for many years, but it is apparently on the verge of requiring a paid subscription to use its webservices. By contrast, for now the Python library, which has an open-source-like license for non-commercial academic use, is free-to-download here, provided that you are a registered user.

I want to briefly describe what NUPACK does, and why it is one of the most used software by the DNA/RNA nanotechnology community. At its core, NUPACK integrates statistical thermodynamics and advanced algorithms to model the folding, hybridization, and thermodynamic stability of nucleic acids. The main appeal of NUPACK compared to other packages lies in its ability to model multi-strand nucleic acid systems. In contrast to simpler single-strand predictions, many DNA nanotechnology applications involve multiple strands interacting to form complex assemblies. NUPACK handles these multi-stranded configurations by extending the core ideas found in the classical algorithms we studied, generalizing them to more complicated systems where the combinatorial possibilities of base-pairing grow exponentially not only with the number of nucleotides, but also with the number of strands.

In practical terms, NUPACK enables researchers to predict the secondary structures that DNA or RNA strands will adopt under given conditions. NUPACK's algorithms minimize free energy to predict the most stable structure, but also consider an ensemble of possible structures, going beyond the minimum free energy configuration. Indeed, is uses McCaskill-like algorithms to compute the partition function, providing information about the entire landscape of possible configurations. Going beyond folding, NUPACK can also design sequences for specific structures: In nanotechnology, designing DNA or RNA that reliably folds into a desired structure is essential for creating functional nanodevices or molecular machines. NUPACK optimizes nucleotide sequences to favor specific target structures while minimizing the formation of undesired alternatives, often through iterative sequence design techniques rooted in statistical optimization.

Python implementation
Head over here for some examples on how to use the NUPACK library.

Chapter 5

Molecular quantum mechanics

> **Warning**
> This is the part that is the farthest from my own expertise, which is about classical simulations. Use this chapter with caution!

> **Tip**
> The main references for this part are Giustino [2014] and Böttcher and Herrmann [2021].

In this chapter I briefly present the background required to understand the basic algorithms used to run *ab-initio* molecular dynamics (AIMD) simulations. The methods presented here makes it possible to simulate the dynamics of atoms and molecules from first principles, which is what "ab initio" means.

5.1 A minimalistic introduction to quantum mechanics

Quantum mechanics is a theoretical framework that describes the behavior of matter and energy at the smallest scales. Unlike classical mechanics, where objects have definite positions and velocities, quantum mechanics introduces the concept of wave-particle duality, where particles exhibit both wave-like and particle-like properties.

The full information about the quantum state of a given quantum system is contained in a complex-valued function of the spatial and time coordinates $\vec{r}$ and t, called the wavefunction, $\Phi(t, \vec{r})$. Max Born's interpretation of the wavefunction provides a probabilistic view of quantum mechanics. According to Born's rule, the probability of finding a particle at position $\vec{r}$ at time t is proportional to $|\Phi(t, \vec{r})|^2$. Thus, the wavefunction must be normalized, meaning that the total probability of finding the particle anywhere in space must equal 1:

$$\int_V |\Phi(t, \vec{r})|^2 d\vec{r} = \int_V |\Psi(\vec{r})|^2 d\vec{r} = 1, \tag{5.1}$$

where we used the factorisation $\Phi(t, \vec{r}) = \phi(t)\Psi(\vec{r})$ which will be justified in a moment, and the integral is carried out over the system's volume, V. This probabilistic interpretation is one of the fundamental departures from classical mechanics, where particles have deterministic trajectories. Note that, in order for Eq. (5.1) to make sense, wave functions need to be "square-integrable" or, in other words, to belong to the Hilbert space $L^2(\mathbb{R}^3)$. This means that it is also possible to define an inner product between any two wave functions ϕ and ψ:

$$\langle \phi | \psi \rangle = \int \phi^*(\vec{r}) \psi(\vec{r}) d\vec{r}, \tag{5.2}$$

where $\phi^*(\vec{r})$ is the complex conjugate of $\phi(\vec{r})$, and we also introduced the bra-ket notation.

In quantum mechanics, physical quantities (observables) such as energy, momentum, and position are represented by operators that act on the wavefunction. For example, the position operator is

$\hat{\vec{r}} = \vec{r}$, and the momentum operator, is $\hat{\vec{p}} = -i\hbar\nabla$, where $\hbar \equiv h/2\pi$ is the reduced Planck constant. The expectation value of an observable over a wavefunction Ψ, which is the average value measured in an experiment, is given by:

$$\langle \hat{A} \rangle = \langle \Psi | \hat{A} | \Psi \rangle = \int \Psi^*(\vec{r}) \hat{A} \Psi(\vec{r}) d\vec{r}, \tag{5.3}$$

where $\hat{A}$ is the operator corresponding to the observable.

The *Hamiltonian operator*, or simply *Hamiltonian*, of a system, $\hat{H}$, represents the total energy of the system (kinetic and potential) and determines its evolution. The Hamiltonian typically takes the form:

$$\hat{H} = \frac{\hat{p}^2}{2m} + V(\vec{r}) = -\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r}) \tag{5.4}$$

where m is the mass of the particle, $\hat{p} = -i\hbar\nabla$ is the particle's momentum, ∇^2 is the Laplacian operator, representing the kinetic energy contribution, and $V(\vec{r})$ is the potential energy as a function of position. For a system of multiple particles, the Hamiltonian becomes more complex, accounting for the kinetic and potential energies of all particles as well as their interactions.

Given a Hamiltonian, the fundamental equation that determines how the wavefunction of a system evolves over time is the time-dependent Schrödinger equation:

$$i\hbar \frac{d\Phi(t, \vec{r})}{dt} = \hat{H}\Phi(t, \vec{r}). \tag{5.5}$$

If $\hat{H}$ does not depend explicitly on time, the dependence on time and spatial variables of the wavefunction can be separated, *i.e.* $\Phi(t, \vec{r}) = \Psi(\vec{r})\phi(t)$. The time-dependence of Eq. (5.5) can be explicitly integrated, yielding

$$\phi(t) = e^{-iEt/\hbar}, \tag{5.6}$$

where E is the expectation value of the Hamiltonian and will be further discussed below. The time-independent part of the Schrödinger equation is instead

$$\hat{H}\Psi(\vec{r}) = E\Psi(\vec{r}). \tag{5.7}$$

This is a partial differential eigenvalue equation, where an operator acts on a function, called eigenfunction, and returns the same function multiplied by the eigenvalue associated to the eigenfunction. Here the eigenfunction is the wavefunction, and the eigenvalue is its energy, E. Since the kinetic part of the Hamiltonian always contains ∇^2 terms, the Schrödinger equation is a second-order differential equation.

5.1.1 The many-body problem

For systems with multiple interacting particles (*e.g.*, an atom, a molecule or larger objects), the Schrödinger equation becomes a many-body problem. The Hamiltonian for such a system includes terms for the kinetic energy of each particle, the potential energy due to external fields, and the interaction energy between the particles. For instance, for a system of N charged particles, the many-body Hamiltonian can be written as:

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{\hbar^2}{2m_i}\nabla_i^2 + V(\vec{r}_i) \right) + \frac{1}{2}\sum_i \sum_{i \neq j} \frac{q_i q_j}{|\vec{r}_i - \vec{r}_j|}, \tag{5.8}$$

where m_i and q_i are the mass and charge of the i-th particle, respectively. Here the first term represents the kinetic energy of the particles, the second term represents the potential energy due to an external field (*e.g.*, from atomic nuclei if we are considering electrons), and the third term represents the Coulomb interaction. Here we set the Coulomb constant $1/4\pi\epsilon_0$ to 1 to simplify the notation.

Atomic units
A common convention is to use atomic units (a.u.), where the reduced Planck constant $\hbar$, the electron charge e, the electron mass m_e, and the Coulomb constant are all set to 1. This simplifies many of the equations and is a common convention in computational quantum chemistry.

Solving the Schrödinger equation for many-body systems exactly is extremely challenging, if not impossible, for all but the simplest cases, such as the hydrogen atom. Therefore, approximations and numerical methods are essential. This is where *ab initio* methods like Density Functional Theory (DFT) come into play, offering a way to handle the complexities of interacting electrons.

5.2 The Born-Oppenheimer approximation

When dealing with molecules or solids, one faces a many-body problem involving both the nuclei and the electrons. The total Hamiltonian of a molecular system includes terms for the kinetic energy of the nuclei, the kinetic energy of the electrons, and the potential energy due to interactions between all particles (nuclei-nuclei, nuclei-electrons, and electron-electron interactions). This problem is, in general, analitically unsolvable due to the coupling between the electronic and nuclear degrees of freedom.

To make the problem tractable, we introduce the Born-Oppenheimer (BO) approximation. This approximation exploits the large mass difference between nuclei and electrons (≈ 3 orders of magnitude) to decouple their motions, making it possible to simplify the Schrödinger equation and compute electronic structures for fixed nuclear configurations.

For a molecule composed of N electrons of mass m_e and M nuclei, each of mass M_I, the total Hamiltonian can be written as:

$$\hat{H} = \hat{T}_{\text{nuc}} + \hat{T}_{\text{el}} + \hat{V}_{\text{nuc-nuc}} + \hat{V}_{\text{nuc-el}} + \hat{V}_{\text{el-el}}, \tag{5.9}$$

where $\hat{T}_{\text{nuc}} = -\sum_{I=1}^{M} \frac{\hbar^2}{2M_I}\nabla_I^2$ and $\hat{T}_{\text{el}} = -\sum_{i=1}^{N} \frac{\hbar^2}{2m_e}\nabla_i^2$ are the kinetic energy operators for the nuclei and electrons, respectively, while $\hat{V}_{\text{nuc -nuc}}$, $\hat{V}_{\text{nuc -el}}$ and $\hat{V}_{\text{el -el}}$ represent the nucleus-nucleus, electron-nucleus, and electron-electron Coulomb interactions, respectively.

The total wavefunction $\Psi(\{\vec{R}_I\}, \{\vec{r}_i\})$ depends on both the positions of the nuclei $\{\vec{R}_I\}$ and of the electrons $\{\vec{r}_i\}$. Solving for the full wavefunction is complicated because the nuclear and electronic motions are coupled through the potential terms. However, the mass disparity between the nucleus and the electrons suggests that, on the timescale of nuclear motion, the electrons can adjust almost instantaneously to any change in nuclear positions. This allows us to decouple the nuclear and electronic motions by assuming that the electronic wavefunction can be solved for a fixed configuration of nuclei.

We assume that the total wavefunction $\Psi(\{\vec{R}_I\}, \{\vec{r}_i\})$ can be factored into a product of a nuclear wavefunction $\chi(\{\vec{R}_I\})$ and an electronic wavefunction $\psi(\{\vec{r}_i\}; \{\vec{R}_I\})$:

$$\Psi(\{\vec{R}_I\}, \{\vec{r}_i\}) \approx \chi(\{\vec{R}_I\})\psi(\{\vec{r}_i\}; \{\vec{R}_I\}) \tag{5.10}$$

Here, the electronic wavefunction $\psi(\{\vec{r}_i\}; \{\vec{R}_I\})$ is parameterized by the fixed nuclear positions $\{\vec{R}_I\}$, while $\chi(\{\vec{R}_I\})$ describes the motion of the nuclei.

5.2.1 Solving the Schrödinger equation

With the nuclei fixed, we can solve the electronic Schrödinger equation for a given set of nuclear positions:

$$\hat{H}_{\text{el}}\psi(\{\vec{r}_i\}; \{\vec{R}_I\}) = E_{\text{el}}(\{\vec{R}_I\})\psi(\{\vec{r}_i\}; \{\vec{R}_I\}). \tag{5.11}$$

The electronic Hamiltonian $\hat{H}_{\text{el}}$ includes the kinetic energy of the electrons and the potential energy due to electron-electron and electron-nuclei interactions:

$$\hat{H}_{\text{el}} = \hat{T}_{\text{el}} + \hat{V}_{\text{el-el}} + \hat{V}_{\text{nuc-el}}. \tag{5.12}$$

The result of solving this equation is the electronic energy $E_{\text{el}}(\{\vec{R}_I\})$, which depends parametrically on the nuclear positions. This energy forms a potential energy surface (PES) on which the nuclei move.

Once the electronic problem is solved, the nuclear dynamics can be described by an effective nuclear Hamiltonian, where the potential energy surface $E_{\text{el}}(\{\vec{R}_I\})$ from the electronic calculation acts as the potential for the nuclei:

$$\hat{H}_{\text{nuc}}\chi(\{\vec{R}_I\}) = \left(\hat{T}_{\text{nuc}} + \hat{V}_{\text{nuc-nuc}} + E_{\text{el}}(\{\vec{R}_I\})\right)\chi(\{\vec{R}_I\}) \tag{5.13}$$

This equation governs the motion of the nuclei on the potential energy surface created by the electrons. In practice, depending on the temperature and energy scales, this motion can be treated either classically or quantum mechanically.

Adiabaticity

In quantum mechanics, a process is called *adiabatic* if the system remains in the same quantum state (or subspace) throughout the process, provided that the external conditions (like the positions of the nuclei) change sufficiently slowly. Specifically, for the BO approximation, this means that as the nuclei move, the electrons remain in their instantaneous ground state at all times.

More formally, if the nuclei evolve adiabatically, the system stays in the ground-state wavefunction of the electronic Hamiltonian corresponding to the instantaneous positions of the nuclei: the nuclear motion is slow enough to allow the electrons to adjust without ever being excited to higher electronic states.

The assumption of adiabaticity—and therefore the validity of the BO approximation—can break down in certain situations, leading to non-adiabatic effects. These effects occur when the nuclear motion becomes fast enough that the electrons cannot adjust instantaneously. In such cases, the system may undergo transitions between different electronic states, leading to a mixing of electronic and nuclear dynamics.

Situations where non-adiabatic effects are significant include:

- **Conical intersections:** Points in the nuclear configuration space where two or more potential energy surfaces intersect. Near conical intersections, the energy gap between electronic states becomes very small, and even slow nuclear motion can lead to electronic transitions.
- **High-temperature or high-energy regimes:** When the nuclei move with enough kinetic energy, the assumption of slow nuclear motion relative to electronic adjustments may no longer hold.
- **Photochemical reactions:** When a molecule absorbs light, it can be excited to a higher electronic state. The subsequent dynamics often involve non-adiabatic transitions between electronic states as the molecule relaxes back to the ground state.

In these cases, the BO approximation fails, and more sophisticated methods, such as time-dependent DFT, are required to accurately describe the system.

5.2.2 Implications and limitations

The Born-Oppenheimer approximation is widely used in *ab initio* simulations because it allows the separation of electronic structure calculations from nuclear motion, significantly reducing the computational complexity. It underpins most of the computational chemistry methods, including Hartree-Fock theory, post-Hartree-Fock methods, and density functional theory (DFT). However, the approximation has some limitations. For instance, in systems where nuclear and electronic motions are strongly coupled (*e.g.*, near conical intersections or in the presence of non-adiabatic effects, see box above), the Born-Oppenheimer approximation breaks down, and one must account for the simultaneous evolution of electrons and nuclei. Moreover, the approximation often treats nuclei as classical particles moving on a potential energy surface, which may neglect important quantum effects like tunneling or zero-point energy, particularly in light atoms like hydrogen.

5.3 The Hohenberg-Kohn theorems

The Hohenberg-Kohn theorems form the foundation of Density Functional Theory (DFT), a widely used approach in *ab initio* simulations. DFT simplifies the quantum many-body problem by shifting the focus from the many-body wavefunction to the electron density $n(\vec{r})$. These theorems, established by Hohenberg and Kohn [1964], provide the theoretical justification for this approach, proving that all ground-state properties of a system are uniquely determined by its electron density. By leveraging this result, it is possible to reformulate quantum mechanics in terms of the electron density, which depends on only three spatial coordinates, rather than the many-body wavefunction, which depends on the coordinates of all electrons. This leads to a significant reduction in complexity and makes DFT one of the most efficient methods for electronic structure calculations in large systems.

Important
The Hohenberg-Kohn theorems apply to non-degenerate ground states only! Here "ground state" means that the energy corresponding to this state is the lowest possible.

Expliciting the terms of an N-electron Hamiltonian (see Eqs. (5.8) and (5.12)) one obtains

$$\hat{H}_{\text{el}} = \sum_i \frac{p_i^2}{2m_e} + \frac{1}{2}\sum_i \sum_{i\neq j} \frac{e^2}{|\vec{r}_i - \vec{r}_j|} + \sum_i V(\vec{r}_i), \tag{5.14}$$

where the terms can be grouped to yield $\hat{H}_{\text{el}} = \hat{F} + \hat{V}$, where F are the kinetic and electron-electron interaction energies, respectively, and V is the "external" potential energy. The ground state Ψ_0 is uniquely determined by N and $V(\{\vec{r}\})$, and we normalise it so that $\langle\Psi_0|\Psi_0\rangle = N$. Its associated electron density is

$$n_0(\vec{r}) = \langle\Psi_0|\rho(\vec{r})|\Psi_0\rangle = N\int |\Psi_0(\vec{r},\vec{r}_2,\ldots,\vec{r}_N)|^2 d\vec{r}_2\ldots d\vec{r}_N, \tag{5.15}$$

where $\rho(\vec{r}) = \sum_i \delta(\vec{r}-\vec{r}_i)$ is the particle density operator. By using this definition, we can write the total potential energy as

$$\begin{aligned}\langle\Psi_0|\hat{V}|\Psi_0\rangle &= \int \sum_i V(\vec{r}_i)|\Psi_0(\vec{r}_1,\vec{r}_2,\ldots,\vec{r}_N)|^2 d\vec{r}_1 d\vec{r}_2\ldots d\vec{r}_N &(5.16)\\ &= N\int V(\vec{r})|\Psi_0(\vec{r},\vec{r}_2,\ldots,\vec{r}_N)|^2 d\vec{r} d\vec{r}_2\ldots d\vec{r}_N &(5.17)\\ &= \int n_0(\vec{r})V(\vec{r})d\vec{r}. &(5.18)\end{aligned}$$

Theorem 5.3.1 (The First Hohenberg-Kohn Theorem: Existence Theorem)**.** *For any system of interacting electrons in an external potential $V(\vec{r})$, the external potential is uniquely determined by the ground-state electron density $n_0(\vec{r})$, except for a trivial additive constant.*

Proof. The theorem is proven by contradiction. Assume that two different external potentials, $V_1(\vec{r})$ and $V_2(\vec{r})$, and therefore two different Hamiltonians, $\hat{H}_1 = \hat{F} + \hat{V}_1$ and $\hat{H}_2 = \hat{F} + \hat{V}_2$, lead to the same ground-state electron density $n_0(\vec{r})$. The corresponding ground-state wavefunctions, Ψ_1 and Ψ_2, as they correspond to different potentials. Their associated energies, which should also be different, are

$$\begin{aligned}E_1 &= \langle\Psi_1|\hat{H}_1|\Psi_1\rangle &(5.19)\\ E_2 &= \langle\Psi_2|\hat{H}_2|\Psi_2\rangle, &(5.20)\end{aligned}$$

Since both Ψ_1 and Ψ_2 are ground states, E_1 and E_2 are the lowest possible with the corresponding Hamiltonian. Therefore

$$E_1 < \langle \Psi_2 | \hat{H}_1 | \Psi_2 \rangle = \langle \Psi_2 | \hat{H}_2 | \Psi_2 \rangle + \langle \Psi_2 | (\hat{H}_1 - \hat{H}_2) | \Psi_2 \rangle = \tag{5.21}$$

$$= E_2 + \int n_0(\vec{r})[V_1(\vec{r}) - V_2(\vec{r})]d\vec{r} \tag{5.22}$$

$$E_2 < \langle \Psi_1 | \hat{H}_2 | \Psi_1 \rangle = \langle \Psi_1 | \hat{H}_1 | \Psi_1 \rangle + \langle \Psi_1 | (\hat{H}_2 - \hat{H}_1) | \Psi_1 \rangle = \tag{5.23}$$

$$= E_1 + \int n_0(\vec{r})[V_2(\vec{r}) - V_1(\vec{r})]d\vec{r}. \tag{5.24}$$

Adding the two inequalities yeilds

$$E_1 + E_2 < E_2 + E_1, \tag{5.25}$$

which is clearly impossible. Therefore, the assumption must be false, and the external potential must be uniquely determined by the electron density (up to a constant).

□

This theorem implies that the many-body wavefunction $\Psi(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)$ is no longer the central object in quantum mechanics. Instead, the electron density $n(\vec{r})$ can be used to determine all ground-state properties. Specifically, the ground-state energy E_0 is a functional of the electron density, $E[n]$, and any other observable that depends on the external potential, such as forces on nuclei or response functions, is also determined by the electron density. Therefore, if one can find the correct ground-state electron density, one can in principle reconstruct the entire external potential and solve for all other properties of the system.

Theorem 5.3.2 (Second Hohenberg-Kohn Theorem: The Variational Principle)**.** *There exists a universal energy functional $E[n]$ of the electron density such that the correct ground-state density $n_0(\vec{r})$ minimizes this functional*[1]*, yielding the ground-state energy E_0.*

This energy functional can be written as:

$$E[n] = F[n] + \int V(\vec{r})n(\vec{r})\, d\vec{r}, \tag{5.26}$$

where $F[n] = T[n] + U[n]$ includes the kinetic energy of the electrons and the electron-electron interaction energy, and $\int V(\vec{r})n(\vec{r})\, d\vec{r}$ is the external potential energy, representing the interaction with the external field (e.g. the electron-nucleus interaction potential in the Born-Oppenheimer approach). The Hohenberg-Kohn functional $F[n]$ is sometimes called "universal functional" since it does not depend on the specific system under study.

Proof. The second Hohenberg-Kohn theorem is a straightforward application of the variational principle, whose proof I also report here for the sake of completeness. Let ψ_n and E_n be the eigenstates and eigenvalues of a Hamiltonian $\hat{H}$, with $E_0 < E_1 < E_2 < \ldots$, so that ψ_0 is the ground state, ψ_1 is the first excited state, *etc.* The eigenstates are orthonormal, *i.e.* $\langle \psi_i | \psi_j \rangle = \delta_{ij}$. Now consider a (normalised) generic wavefunction ψ, which can be expressed in the basis of the eigenstates as

$$\psi = \sum_i c_i \psi_i, \tag{5.27}$$

where $\sum_i |c_i|^2 = 1$, which implies that

$$|c_0|^2 = 1 - \sum_{i>0} |c_i|^2. \tag{5.28}$$

The expectation value of the Hamiltonian on ψ is

[1] This is analogous to the Rayleigh-Ritz variational principle for wavefunctions.

$$\langle\psi|\hat{H}|\psi\rangle = \langle\sum_i c_i\psi_i|\hat{H}|\sum_j c_j\psi_j\rangle = \sum_{i,j} c_i^* c_j\langle\psi_i|\hat{H}|\psi_j\rangle = \tag{5.29}$$

$$= \sum_{i,j} c_i^* c_j E_j\langle\psi_i|\psi_j\rangle = \sum_j |c_j|^2 E_j = |c_0|^2 E_0 + \sum_{j>0} |c_j|^2 E_j, \tag{5.30}$$

which, by applying Eq. (5.28), becomes

$$\langle\psi|\hat{H}|\psi\rangle = E_0 + \sum_{j>0} |c_j|^2 (E_j - E_0) > E_0, \tag{5.31}$$

since $|c_j|^2 \geq 0$ and $E_j - E_0 > 0 \; \forall j > 0$.

Back to the Hohenberg-Kohn theorem, we know that the electron density $n(\vec{r})$ fully determines the potential $V(\vec{r})$ and the ground state Ψ_0. Consider a trial electron density $n'(\vec{r})$ that is not the true ground-state density $n_0(\vec{r})$. This will be associated to an external potential V' and to a wave function Ψ'. By applying the variational principle, we find that the expectation value of the true Hamiltonian on this wave function is

$$\langle\Psi'|\hat{H}_{\text{el}}|\Psi'\rangle = E[n'] \leq E[n_0] = \langle\Psi|\hat{H}_{\text{el}}|\Psi\rangle, \tag{5.32}$$

where the equality holds if and only if $n'(\vec{r}) = n_0(\vec{r})$.

□

The second Hohenberg-Kohn theorem provides a practical way to find the ground-state electron density by minimizing the energy functional $E[n]$. This is the basis of modern DFT numerical calculations:

1. Start with a trial electron density $n(\vec{r})$.
2. Calculate the total energy functional $E[n]$.
3. Adjust the electron density to minimize the energy.
4. The density that minimizes the energy is the true ground-state density, and the corresponding energy is the ground-state energy.

This approach bypasses the need to solve the many-body Schrödinger equation directly, instead focusing on finding the electron density that minimizes the energy functional. The challenge in DFT is to find an accurate expression for the universal functional $F[n] = T[n] + U[n]$, as this functional is not known explicitly for interacting electrons. As we will now see, various approximations, such as the Local Density Approximation (LDA) and Generalized Gradient Approximation (GGA), have been developed to approximate this functional.

5.4 The Kohn-Sham approximation

The Hohenberg-Kohn theorems laid the foundation for Density Functional Theory (DFT) by demonstrating that all ground-state properties of a many-electron system are determined by the electron density $n(\vec{r})$. However, the exact form of the universal energy functional $E[n]$, which includes the kinetic energy and electron-electron interaction energy, is unknown for interacting electrons. This is the key challenge in practical DFT calculations.

The Kohn-Sham (KS) approximation, introduced by Kohn and Sham [1965], provides a solution to this problem by mapping the interacting electron system onto an auxiliary system of non-interacting electrons that has the same ground-state electron density. This makes it possible to treat the complex interactions in a computationally efficient way while still capturing the essential physics of the many-body problem.

5.4.1 The Kohn-Sham equations

The central idea of the Kohn-Sham approach is to introduce a fictitious system of non-interacting electrons that reproduces the exact ground-state electron density of the real, interacting system. For this non-interacting system, the kinetic energy can be computed exactly in terms of single-particle orbitals, which in this context are called the Kohn-Sham orbitals, ψ_i. The electron density is also constructed from the Kohn-Sham orbitals as:

$$n(\vec{r}) = \sum_{i=1}^{N} |\psi_i(\vec{r})|^2. \tag{5.33}$$

Instead of attempting to directly approximate the total energy functional $E[n]$ for the interacting system, we decompose it into parts that can be computed exactly and parts that require approximations. The total energy functional in the Kohn-Sham formalism is written as:

$$E[n] = T_s[n] + \int V(\vec{r})n(\vec{r})\, d\vec{r} + U_{\text{el-el}}[n] + E_{\text{xc}}[n], \tag{5.34}$$

where $T_s[n]$ is the kinetic energy of the non-interacting electrons, $U_{\text{el -el}}[n]$ is the Coulomb energy, representing the classical electrostatic interaction between electrons, and $E_{\text{xc}}[n]$ is the exchange-correlation energy, which includes all the many-body effects not captured by the other terms (*e.g* the electron-electron effective repulsion due to the Pauli exclusion principle).

To find the ground-state density, the Kohn-Sham approach involves solving a set of single-particle equations, known as the Kohn-Sham equations. These equations describe the motion of non-interacting electrons in an effective potential $v_{\text{eff}}(\vec{r})$, and can be obtained by using the fact that the functional in Eq. (5.34) is minimised by the ground-state electron density. Therefore, $\delta E/\delta n|_{n(\vec{r})=n_0(\vec{r})} = 0$. If we carry out the functional derivative with the constraint that the Kohn-Sham orbitals are orthonormal, we obtain (see Appendix B of Giustino [2014] for the full derivation)

$$\left(-\frac{\hbar^2}{2m}\nabla^2 + v_{\text{eff}}(\vec{r})\right)\psi_i(\vec{r}) = \epsilon_i\psi_i(\vec{r}), \tag{5.35}$$

ϵ_i is the single-particle energy corresponding to ψ_i, and the effective potential, which includes the effects of the external potential, the Hartree potential, and the exchange-correlation potential, is given by

$$v_{\text{eff}}(\vec{r}) = V(\vec{r}) + \int \frac{n(\vec{r'})}{|\vec{r} - \vec{r'}|} d\vec{r'} + v_{\text{xc}}(\vec{r}) \tag{5.36}$$

The Kohn-Sham equations must be solved self-consistently:

1. Start with an initial guess for the electron density $n(\vec{r})$.
2. Construct the effective potential $v_{\text{eff}}(\vec{r})$.
3. Solve the Kohn-Sham equations to obtain the orbitals $\psi_i(\vec{r})$.
4. Calculate a new electron density from the orbitals.
5. Repeat until the electron density converges.

5.4.2 The exchange-correlation functional

The exchange-correlation functional $E_{\text{xc}}[n]$ plays a crucial role in DFT, as it captures the many-body effects of electron exchange and correlation that are not accounted for in the other terms of the Kohn-Sham functional. This term is responsible for the accuracy of DFT calculations, and its exact form is unknown.

Several approximate forms for the exchange-correlation functional have been developed, each offering a different balance between accuracy and computational cost. Here I list the most common approximations.

Local Density Approximation

In the Local Density Approximation (LDA), the exchange-correlation energy density at each point in space is assumed to depend only on the local electron density $n(\vec{r})$:

$$E_{\text{xc}}^{\text{LDA}}[n] = \int \epsilon_{\text{xc}}(n(\vec{r}))n(\vec{r})d\vec{r}. \tag{5.37}$$

The LDA is based on the exchange-correlation energy of a homogeneous electron gas. While it can be surprisingly accurate for systems with slowly varying densities (*e.g.*, bulk solids), it fails, sometimes dramatically, in cases where the electron density varies rapidly, such as in molecules or surfaces.

Generalized Gradient Approximation

The Generalized Gradient Approximation (GGA) improves upon the LDA by incorporating not only the local electron density but also its gradient $\nabla n(\vec{r})$:

$$E_{\text{xc}}^{\text{GGA}}[n] = \int \epsilon_{\text{xc}}(n(\vec{r}), \nabla n(\vec{r}))n(\vec{r})d\vec{r}. \tag{5.38}$$

GGA functionals, such as the Perdew et al. [1996] functional, are more accurate for systems with significant density variations, such as molecules, surfaces, and low-dimensional materials. They are widely used in practical DFT calculations.

Hybrid Functionals

Hybrid functionals go beyond the local and semi-local approximations by mixing a portion of exact exchange energy from Hartree-Fock theory with the exchange-correlation energy from DFT. The most well-known hybrid functional is Stephens et al. [1994], which is popular in quantum chemistry for molecular systems. Hybrid functionals generally provide higher accuracy but at a significantly increased computational cost.

5.4.3 Implications and limitations

The Kohn-Sham approximation is a powerful and practical method for implementing Density Functional Theory. By mapping the interacting electron problem onto a non-interacting system with the same electron density, it makes DFT calculations computationally tractable. In fact, DFT with the Kohn-Sham approximation can be applied to systems with hundreds or even thousands of atoms, from molecules to solids, and can provide a good balance between computational cost and accuracy, particularly with carefully chosen exchange-correlation functionals. Unfortunately, the choice of the functional greatly impacts the accuracy of the DFT calculations, and no single functional is universally accurate.

Moreover, for some systems (*e.g.*, strongly correlated materials), DFT may fail to provide reliable results. For instance, the most approximate functionals (*e.g.*, LDA and GGA) suffer from self-interaction errors, where an electron incorrectly interacts with itself. This can lead to inaccuracies, particularly in systems with localized states, such as transition metal oxides and open-shell molecules. Moreover, DFT is primarily a ground-state theory, and its application to excited states is limited. Time-dependent DFT extends DFT to excited states, but it also inherits the limitations of the exchange-correlation functionals used.

5.5 The Hellmann-Feynman Theorem

In the sections above I have sketched a way of solving the electronic problem and obtain the ground-state electron density. However, in *ab initio* simulations we have to also evolve the positions of the nuclei. How do we connect $n(r)$ to the forces acting on the nuclear degrees of freedom? The Hellmann-Feynman theorem provides a convenient and efficient way to compute these forces within

the framework of quantum mechanics, without requiring the explicit calculation of the derivatives of the wavefunction.

The theorem states that when a system is in an eigenstate of its Hamiltonian, the forces acting on a nucleus can be calculated directly from the electron density and the external potential. This result greatly simplifies force calculations in Density Functional Theory (DFT) and other quantum mechanical methods, making it possible to efficiently perform simulations of atomic motion.

Theorem 5.5.1 (The Hellmann-Feynman Theorem)**.** *Consider a quantum system described by a Hamiltonian $\hat{H}(\lambda)$, which depends on a parameter λ (e.g., the position of a nucleus in a molecule or solid). If $\psi(\lambda)$ is an eigenstate of the Hamiltonian with eigenvalue $E(\lambda)$, then the derivative of the eigenvalue with respect to the parameter λ is given by the expectation value of the derivative of the Hamiltonian:*

$$\frac{dE(\lambda)}{d\lambda} = \left\langle \psi(\lambda) \left| \frac{d\hat{H}(\lambda)}{d\lambda} \right| \psi(\lambda) \right\rangle. \tag{5.39}$$

Proof. Consider the time-independent Schrödinger equation for a system with a Hamiltonian $\hat{H}(\lambda)$ that depends on a parameter λ:

$$\hat{H}(\lambda)\psi(\lambda) = E(\lambda)\psi(\lambda) \tag{5.40}$$

Taking the derivative of both sides of this equation with respect to λ we obtain[2]

$$\frac{d\hat{H}}{d\lambda}\psi + \hat{H}\frac{d\psi}{d\lambda} = \frac{dE}{d\lambda}\psi + E\frac{d\psi}{d\lambda}. \tag{5.41}$$

If we multiply both sides by ψ^* and integrate over all space we get

$$\left\langle \psi \left| \frac{d\hat{H}}{d\lambda} \right| \psi \right\rangle + \left\langle \psi \left| \hat{H} \right| \frac{d\psi}{d\lambda} \right\rangle = \frac{dE}{d\lambda} + E \left\langle \psi \left| \frac{d\psi}{d\lambda} \right\rangle \right. . \tag{5.42}$$

Since ψ is an eigenstate of $\hat{H}$, $\hat{H}\psi = E\psi$, so that the second term can be simplified:

$$\left\langle \psi \left| \frac{d\hat{H}}{d\lambda} \right| \psi \right\rangle + E \left\langle \psi \left| \frac{d\psi}{d\lambda} \right\rangle \right. = \frac{dE}{d\lambda} + E \left\langle \psi \left| \frac{d\psi}{d\lambda} \right\rangle \right. . \tag{5.43}$$

The terms involving the derivative of the wavefunction cancel out, yielding

$$\frac{dE(\lambda)}{d\lambda} = \left\langle \psi(\lambda) \left| \frac{d\hat{H}(\lambda)}{d\lambda} \right| \psi(\lambda) \right\rangle. \tag{5.44}$$

□

In the context of DFT, the Hellmann-Feynman theorem is particularly useful for calculating the forces on nuclei during molecular dynamics or geometry optimization. The force $\vec{F}_I$ on nucleus I is given by the negative gradient of the total energy with respect to the position of that nucleus:

$$\vec{F}_I = -\nabla_{\vec{R}_I} E_{\text{total}}(\{\vec{R}\}) \tag{5.45}$$

Consider the full Hamiltonian of a molecular system, Eq. (5.9): the only two terms that depend on the positions of nuclei $\{\vec{R}_I\}$ are the nuclear-nuclear and nuclear-electron interactions. Using the Hellmann-Feynman theorem, these two contributions can be explicitly calculated:

$$\vec{F}_I = -\int n(\vec{r})\nabla_{\vec{R}_I} V(\vec{r};\{\vec{R}\})\, d\vec{r} - \nabla_{\vec{R}_I} V_{\text{nuc-nuc}}(\{\vec{R}\}), \tag{5.46}$$

where we used the expression of the electron-nucleus interaction in terms of the electron density $n(r)$ (see Eq. (5.26)). This result shows that the force on a nucleus depends only on the electron

[2] In the derivation I drop the explicit dependence on λ for clarity.

density and the derivative of the external potential (*e.g.*, the Coulomb potential from the other nuclei). Importantly, it does not require calculating the derivatives of the wavefunctions with respect to the nuclear positions, which is computationally demanding.

5.5.1 Implications and limitations

The Hellmann-Feynman theorem is widely used in *ab initio* simulations for calculating forces on atoms, particularly in DFT and other quantum chemistry methods. The theorem simplifies force calculations by avoiding the need for wavefunction derivatives. This makes molecular dynamics and geometry optimization in DFT feasible for large systems. Interestingly, the forces calculated using the Hellmann-Feynman theorem are exact as long as the wavefunction is an exact eigenstate of the Hamiltonian. In practice, errors can arise if the wavefunction is not fully converged, but these can often be minimized with proper numerical techniques. Moreover, the Hellmann-Feynman theorem also applies to systems where the parameter λ represents something other than nuclear positions, such as the strength of an external field or the variation of a coupling constant, making it a versatile tool.

However, note that in some cases, additional corrections beyond the Hellmann-Feynman theorem are needed. For example, when using approximate wavefunctions or basis sets, so-called Pulay [1969] arise from using a non-complete basis functions that depend on the nuclear positions, contributing to the total force calculation in methods where the basis set is not fixed (*e.g.*, Gaussian-type orbitals).

This can be demonstrated by using the following derivation (heavily inspired by this answer by Michael F. Herbst). Consider a non-complete basis $\{f\}$, which we use to express our wavefunction:

$$\Psi = \sum_i c_i f_i, \tag{5.47}$$

where c_i are constants chosen so that the resulting Ψ is a good approximation of the ground state, and it is associated to an energy E. Here "good" means that each coefficient c_i obeys the variational principle:

$$0 = \frac{dE}{dc_i} = 2\left\langle \Psi \middle| H \middle| \frac{d\Psi}{dc_i} \right\rangle = 2\left\langle \Psi | H | f_i \right\rangle \tag{5.48}$$

Since we are interested in forces, instead of taking the derivative of the energy with respect to a generic parameter λ, we derive the energy with respect to the nuclear positions:

$$\frac{dE}{d\vec{R}} = \left\langle \Psi \middle| \frac{dH}{d\vec{R}} \middle| \Psi \right\rangle + 2\left\langle \Psi \middle| H \middle| \frac{d\Psi}{d\vec{R}} \right\rangle. \tag{5.49}$$

As we have seen in Proof 5.5, the second ("Pulay") term vanishes if Ψ is an eigenstate. Let's see what happens in this case. Consider as an example the derivative with respect to R_1. Its Pulay term is

$$\left\langle \Psi \middle| H \middle| \left(\sum_i c_i \frac{df_i}{dR_1} \right) \right\rangle \tag{5.50}$$

When does this term vanish? There are two possibilities:

1. The derivatives df_i/dR_1 are all zero, *i.e.* if the basis functions are independent of atomic positions, *e.g.* plane waves.

2. If the derivatives $\frac{df_1}{dR_1}$ and $\frac{df_2}{dR_1}$ are themselves basis functions or can be exactly represented by the basis. In this case,

$$\frac{df_i}{dR_1} = \sum_j k_{ij} f_i \tag{5.51}$$

for some values of k_{ij}. Now the Pulay term can be rewritten by leveraging Eq. (5.48) as

$$\left\langle \Psi \middle| H \middle| \left(\sum_i c_i \frac{df_i}{dR_1} \right) \right\rangle = \left\langle \Psi \middle| H \middle| \left(\sum_{i,j} c_i k_{ij} f_j \right) \right\rangle = \tag{5.52}$$

$$= \sum_{i,j} c_i k_{ij} \langle \Psi | H | f_j \rangle = 0. \tag{5.53}$$

5.6 The Car-Parrinello method

The Car-Parrinello method, introduced by Car and Parrinello [1985], revolutionized the field of *ab initio* molecular dynamics (AIMD) by combining classical molecular dynamics with electronic structure calculations based on DFT. The key innovation of this method is its ability to simultaneously evolve both nuclear and electronic degrees of freedom within a unified framework.

As discussed Section 5.2, in traditional Born-Oppenheimer molecular dynamics, the forces on the nuclei are obtained by solving the electronic structure problem for each configuration of the nuclei, possibly with DFT-like approaches. The nuclei are then moved according to classical equations of motion, and this process is repeated iteratively. Although this approach is accurate, it is computationally expensive because a self-consistent field (SCF) calculation must be performed at every MD time step to obtain the forces.

In contrast, the Car-Parrinello molecular dynamics (CPMD) method evolves both the nuclear positions and the electronic wavefunctions simultaneously. This avoids the need for repeated SCF calculations, making the method more efficient while retaining accuracy. The key idea is to treat the Kohn-Sham orbitals as fictitious dynamical variables governed by a classical-like equation of motion, allowing them to follow the nuclear motion without needing to re-optimize the electronic structure at each step.

5.6.1 The equations of motion

The Car-Parrinello method is based on a Lagrangian formulation that includes both the nuclear and electronic degrees of freedom. The total Lagrangian L for the system is given by:

$$L = \frac{1}{2} \sum_I M_I \dot{\vec{R}}_I^2 + \frac{\mu}{2} \sum_i \langle \dot{\psi}_i | \dot{\psi}_i \rangle - E_{\text{tot}}[\{\psi_i\}, \{\vec{R}_I\}] + \sum_{ij} \lambda_{ij} (\langle \psi_i | \psi_j \rangle - \delta_{ij}), \tag{5.54}$$

where $\vec{R}_I$ are the nuclear positions, M_I are the nuclear masses, $\dot{\vec{R}}_I$ are the nuclear velocities, ψ_i are the Kohn-Sham orbitals, μ is a fictitious mass-like parameter associated with the electronic degrees of freedom, $\dot{\psi}_i$ are the time derivatives ("velocities") of the Kohn-Sham orbitals, and $E_{\text{tot}}[\{\psi_i\}, \{\vec{R}_I\}] = E[\{\psi_i\}, \{\vec{R}_I\}] + E_{\text{nuc,nuc}}(\{\vec{R}_I\})$ is the sum of the DFT (Kohn-Sham) energy functional and of the nuclear-nuclear interaction potential.

The first term in Eq. (5.54) represents the kinetic energy of the nuclei, the second term represents the kinetic energy of the fictitious electronic degrees of freedom, the third term $E_{\text{tot}}[\{\psi_i\}, \{\vec{R}_I\}]$ is the total potential energy, which includes contributions from the nuclear-nuclear, nuclear-electronic, and electron-electron interactions, while the fourth term enforces the orthonormality of the Kohn-Sham orbitals.

From this Lagrangian, one can derive the equations of motion for both the nuclei and the electronic wavefunctions using the Euler-Lagrange formalism:

$$M_I \ddot{\vec{R}}_I = -\frac{\partial E_{\text{tot}}[\{\psi_i\}, \{\vec{R}_I\}]}{\partial \vec{R}_I} + \sum_{i,j} \lambda_{ij} \frac{\partial}{\partial \vec{R}_I} \langle \psi_i | \psi_j \rangle \tag{5.55}$$

$$\mu \ddot{\psi}_i(\vec{r}) = -\frac{\delta E_{\text{tot}}[\{\psi_i\}, \{\vec{R}_I\}]}{\delta \psi_i^*(\vec{r})} + \sum_j \lambda_{ij} \psi_j \tag{5.56}$$

Here, the fictitious mass parameter μ controls the dynamics of the electronic wavefunctions. These equations ensure that the electronic wavefunctions follow the nuclear motion, remaining close to the

> As the kinetic energy T_e of the electronic degrees of freedom will vary with time, so will E_{phys}, the total energy of the relevant Born-Oppenheimer system. For a physically meaningful simulation, it is therefore necessary that T_e only performs bound oscillations around a constant small value. The interpretation of this state of the system is that the total CP dynamical system consists of two adiabatically decoupled subsystems, the cold electronic degrees of freedom and the nuclear degrees of freedom at the relevant physical temperature. The adiabatic separation will never be complete and it will be necessary to choose the mass parameter μ in a way that the two systems stay decoupled over the full range of the simulation. However, the choice of μ also directly affects the efficiency of the simulation, as the maximal time step of the numerical integration is proportional to $\sqrt{\mu}$. The choice of an optimal value of the free parameter μ in a CP simulation needs care and has been discussed at length [in the literature].
>
> Hutter [2011]

instantaneous ground state of the electronic structure, without the need for explicit SCF iterations at each time step. Note that in the functional derivative that appears in the electronic equations of motions, only the terms of the non-interacting (Kohn-Sham) Hamiltonian survives, since the nuclear-nuclear interaction does not depend on the ψ_i orbitals.

The constant of motion (*i.e.* the quantity that is conserved) corresponding to the time-independence of the Car-Parrinello Langrangian is

$$E = \frac{1}{2}\sum_I M_I \dot{\vec{R}}_I^2 + \frac{\mu}{2}\sum_i \langle \dot{\psi}_i | \dot{\psi}_i \rangle + E_{\text{tot}}[\{\psi\}, \{\vec{R}\}] \equiv E_{\text{phys}} + T_e. \tag{5.57}$$

As well put in an excellent review,

Indeed, the success of the Car-Parrinello method relies on a careful choice of the fictitious mass parameter μ. This parameter determines the timescale on which the electronic wavefunctions evolve. To ensure that the electrons remain adiabatically "slaved" to the nuclei, μ must be small enough that the electronic motion remains much faster than the nuclear motion. However, μ cannot be too small, or the time step required for stable integration of the electronic equations of motion would become prohibitively small. Therefore, μ should be chosen such that the electronic degrees of freedom evolve rapidly enough to stay close to the instantaneous ground state, but not so fast that the computational efficiency is compromised (which is easier said than done).

5.6.2 Implications and limitations

The Car-Parrinello method offers several advantages over traditional approaches (such as those based on the Born-Oppenheimer approximation). By evolving the electronic wavefunctions dynamically, CPMD avoids the need for repeated SCF calculations, leading to significant savings in computational cost, particularly for large systems. Moreover, the method retains the accuracy of DFT-based molecular dynamics since the electronic wavefunctions are kept close to the ground state throughout the simulation. Finally, the treatment of both nuclear and electronic degrees of freedom within a single Lagrangian formalism provides a natural and elegant way to simulate systems where electronic structure and nuclear motion are strongly coupled.

However, the introduction of the fictitious kinetic energy for the electronic wavefunctions means that energy is not strictly conserved over long simulation times. Although this can be controlled by adjusting the fictitious mass μ, it remains a source of error that requires careful monitoring. The need to integrate both the nuclear and electronic equations of motion imposes restrictions on the time step size, which can be smaller than in traditional Born-Oppenheimer molecular dynamics, depending on the choice of μ. The Car-Parrinello method assumes that the electronic wavefunctions remain close to the ground state during nuclear motion. However, in systems where non-adiabatic effects (such as electronic excitations) are important, more advanced methods may be required. Extensions of the Car-Parrinello method have been developed to address some of these limitations. For example, extended

Lagrangian methods introduce additional degrees of freedom to improve energy conservation, while time-dependent DFT can be used to model systems where excited-state dynamics play a significant role.

5.7 Running simulations

While I will not provide any example, here I mention some of the most used open-source software packages for performing *ab initio* molecular dynamics simulations:

- CP2K
- SIESTA
- Quantum ESPRESSO

Most of these packages can be used to simulate systems at varying (and even mixed) levels of details.

Chapter 6

Classical molecular dynamics and all-atom simulations

Tip
The main references for this part are Frenkel and Smit [2023], Šulc et al. [2018] (which can be downloaded here), Schlick [2010] and Leach [2001].

Quantum mechanics provides a rigorous framework for describing the behavior of molecules that explicitly includes the effect of the electrons. However, the calculations are very heavy and thus the evolution of a system can be followed for short times only. In addition, the computational complexity of simulating quantum systems often scales super-linearly with the size of the problem[1]. This poses significant challenges when attempting to model large-scale many-body systems accurately and for long times.

One of the fundamental distinctions between quantum and classical approaches lies in the treatment of electron contributions. In quantum calculations, the interactions and motions of electrons are computed explicitly, for instance with DFT methods, as discussed in the previous Chapter. This level of detail enables precise predictions of electronic structure and properties but comes at a high computational cost, particularly as the number of electrons and/or atoms increases.

In contrast, classical force fields adopt a simplified approach where the contributions of electrons are averaged out. Instead of explicitly modeling individual electron behaviors, classical force fields approximate the interactions between atoms using simplified mathematical models based on classical mechanics. Note that there exist also "mixed" methods, where some degrees of freedom are accounted for by using a quantum-mechanical treatment, while others are treated classically, *e.g.* the nuclear degrees of freedom in the CPMD approach.

In the rest of the Chapter (unless stated otherwise), we consider systems composed of N interacting point particles following Newton's equations of motion set by a classical Hamiltonian:

$$H = \sum_{i=1}^{N} \frac{p_i^2}{2m_i} + V(\{\vec{r}_i\}), \tag{6.1}$$

where $\vec{p}_i$ and $\vec{r}_i$ are the momentum and coordinates of particle i, m_i is its mass and $V(\{\vec{r}_i\})$ is the total potential energy.

Python implementation
Some of the algorithms presented in this section have been implemented in this Jupyter notebook.

[1] The complexity ranges from $\mathcal{O}(e^N)$ for brute-force implementations, to $\mathcal{N}^3$ for many DFT codes, but can be linear in some cases (see *e.g.* Bowler and Miyazaki [2012]).

6.1 Molecular dynamics

In a molecular dynamics simulation, the atoms, or particles, follow Newton's equations of motions. As a result, the dynamics happens on a hypersurface defined by $E = \text{const}$, where E is the energy of the system. In practice, the equations of motions are solved iteratively by discretising time: the quantities of interest (position r, velocity v and force F) at time t are used to obtain those at time $t + \Delta t$, where Δt is the integration *time step*. The flow of a simple MD program is:

1. The simulation parameters (*e.g.* temperature, density, time step) are read and initialised.
2. The initial configuration (*i.e.* the initial positions and velocities) is read or generated.
3. The simulation runs, iterating the following steps:
   (a) The forces (and possibly torques) on all particles are computed.
   (b) The positions and velocities are updated according to Newton's equations.
4. The simulation stops when some condition is met (*e.g.* number of steps run).

The following snippet shows a pseudo-code implementation of the foregoing algorithm:

```
CALL initialise_parameters()
CALL initialise_system()

WHILE t is smaller than t_max
   CALL compute_force()
   CALL integrate()
   t += delta_t
   CALL sample_observables()
```

Figure 6.1: Pseudo-code for a simple molecular dynamics code.

> **Important**
> Nowadays there are sophisticated codes that can be used, in principle, to run MD simulations with no knowledge any of the algorithms employed. This is **very** dangerous, as it is very easy to produce wrongs results that appear correct if one does not know what they are doing. Therefore, I urge you to try to understand how (and why) these algorithms work.

6.1.1 Initialisation

Initialising a simulation is a trivial task when simulating simple systems, *e.g.* gases, most liquids and even many solids, but can become extremely difficult for highly heterogeneous systems interacting through complicated, and possibly steep, interactions. Biomacromolecules modelled at the all-atom level are definitely within this class of systems.

In general, the initial configuration should be such that there is no sensible overlap between the system's constituents (atoms, molecules, particles, *etc.*), in order to avoid large initial forces that could lead to numerical instabilities that could lead to crashes or, which is worse, unphysical behaviour, such as two chains crossing each other. For simple systems this can be done in several ways:

- Particle coordinates are chosen randomly one after the other, ensuring that the distance between the new particle and any other particle is larger than sum threhsold. This works great as long as the density is not too high.
- Particles are placed on a lattice, making it possible to go to generate highly dense systems.

For more complicated systems, the generation of the initial configuration is often done in steps. In some cases, and I will show some examples, it is also common to run short-ish simulations where the dynamics is constrained (*e.g.* some degrees of freedom such as bond lengths or angles are kept fixed) and the energy is minimised to remove (part of) the initial stress.

In the case of highly heterogeneous systems, it is very common to use external tools to build part of (or the whole) system. Take for instance the simulation of protein-membrane systems (*e.g.* Figure 2.28), where one has to build a membrane with a given composition and density, embed one or more proteins, add ions and solvate the system.

What about velocities (or, in general, momenta)? Here initialisation is more straightforward, but there are some subtleties that can lead to errors later on if one is not careful. First of all, recall that, in a system in thermal equilibrium,

$$\langle v_{i,\alpha}^2 \rangle = \frac{k_B T}{m}, \tag{6.2}$$

where $v_{i,\alpha}$ is the α-th component of the velocity of particle i. We can then define the instantaneous temperature of a system of N particles[2] as

$$k_B T(t) = \sum_{i,\alpha}^{N} \frac{m v_{i,\alpha}^2}{dN}, \tag{6.3}$$

where d is the dimensionality of the system. We now use this relation in the following initialisation procedure:

1. Randomly extract each velocity component of each particle from a distribution with zero mean. A good choice is a Gaussian, but a uniform distribution will also do (but don't be lazy!).
2. Set the total momentum of the system to zero by computing $\langle v_\alpha \rangle = \frac{1}{N} \sum_i^N v_{i,\alpha}$ and subtracting it from each $v_{i,\alpha}$.
3. Rescale all velocities so that the initial temperature $T(0)$ matches the desired value T, *i.e.* multiply each component by $\sqrt{T/T(0)}$.

If you chose to use a Gaussian distribution for step 1., then you will end up with velocities distributed according to the Maxwell-Boltzmann probability distribution at temperature T, *viz.*

$$P(v) = \left(\frac{\beta}{2\pi m}\right)^{3/2} \exp\left(-\frac{\beta m v^2}{2}\right). \tag{6.4}$$

However, since the initial configuration is most likely *not* an equilibrium configuration, the subsequent equilibration will change the average temperature, so that $\langle T(t) \rangle \neq T$. We will see later on how to couple the system to a Section 6.2.1 to enforce thermal equilibrium at the desired temperature.

6.1.2 Reduced units

In molecular simulations, reduced units are employed to simplify calculations by scaling physical quantities relative to characteristic properties of the system, such as particle size or interaction strength. This approach not only makes the equations dimensionless and more general, but it also prevents the numerical instability that can arise from dealing with values that are extremely small or large, which can occur when using standard physical units (*e.g.* SI units). For example, distances are often scaled by the size of the particles, *e.g.* the particle diameter σ, energies by a characteristic interaction energy, *e.g.* the well depth of the attraction ϵ, and masses by the particle mass, m. Any other unit is then expressed as a combination of these basic quantities; for instance, given σ, ϵ, and m, time is in units of $\sigma\sqrt{m/\epsilon}$, and pressure is in units of ϵ/σ^3. This allows quantities like temperature to be expressed as $T^* = \frac{k_B T}{\epsilon}$, reducing the need to handle numbers with extreme magnitude, which can hinder numerical stability and therefore lead to inaccuracies in the simulation. As an alternative, it is also possible to

[2] here I assume that we are dealing with point-like objects.

use the characteristic constants directly as units of measurements. For instance, a distance may be written as $L = 10\,\sigma$, or an energy as $U = 0.1\,\epsilon$.

6.1.3 The force calculation

This is the most time-consuming part of an MD simulation. It consists of evaluating the force (and, for rigid bodies, the torque) acting on each individual particle. Since, in principle, every particle can interact with every other particle, for a system of N objects the number of interacting contributions will be equal to the number of pairs, $N(N-1)/2$. Therefore, the time required to evaluate all forces scales as $\mathcal{O}(N^2)$. Although, as we will see later, there are methods that can bring this complexity down to $\mathcal{O}(N)$, here I present the simplest case.

Consider, for simplicity, a system made of N like atoms interacting through the Lennard-Jones potential modelling Section 2.3.2. The atom-atom interaction depends only on the inter-atomic distance r, and reads

$$V_{\mathrm{LJ}}(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right), \tag{6.5}$$

where ϵ is the potential well depth and σ is the atom diameter. The force associated to this potential is then

$$\vec{f}(\vec{r}) = -\vec{\nabla} V_{\mathrm{LJ}}. \tag{6.6}$$

Interaction cut-off

The range of the interaction

A "short-range potential" is a definition that applies to all those potentials that can be truncated at a certain distance r_c (called *cut-off distance*), and whose tail contribution (*i.e.* the contribution for $r > r_c$) is finite. This can happen either because $V(r > r_c) = 0$, or if the potential decays to zero quickly enough. Indeed, the tail correction can be estimated by considering the continuous limit, *viz.*

$$U_{\mathrm{tail}} \approx \frac{N\rho}{2}\int_{r_c}^{\infty} V(r)\,d\vec{r}, \tag{6.7}$$

where ρ is the number density of the system, and $d\vec{r} = 4\pi r^2 dr$ in 3D and $d\vec{r} = 2\pi r dr$ in 2D. In order to yield a finite value, the integrand in Eq. (6.7) should decay faster than r^{-1}. If $V(r)$ decays at large distances as $\sim r^{-\alpha}$, the convergence condition becomes $\alpha > d$, where d is the dimensionality.

Given the definition above, the LJ potential is short ranged[3]. As a result, the total potential energy of a particle is dominated by the contributions of those particles that are closer than some cut-off distance r_c. Therefore, in order to save some computing time, it is common to truncate the interaction at r_c, so that the potential reads

$$V_{\mathrm{tr}}(r) = \begin{cases} 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right) & \text{if } r \le r_c \\ 0 & \text{otherwise.} \end{cases} \tag{6.8}$$

In this way, only pairs of particles closer than r_c feel a mutual interaction. However, the potential has a discontinuity, and therefore the force diverges, at $r = r_c$. The discontinuity introduces errors in the estimates of several quantities. For instance, the potential energy will lack the contributions of distance atoms, which can be estimated through Eq. (6.7) and for a 3D LJ potential is

[3]But Coulomb and dipolar interactions are not, and has to be treated differently, as we will see later on.

$$U_{\text{tail}} \approx 2\pi N\rho \int_{r_c}^{\infty} V(r) r^2 dr = \frac{8}{3}\pi N\rho\epsilon\sigma^3 \left[\frac{1}{3}\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3\right]. \tag{6.9}$$

Another important observable affected by the truncation is the pressure. The correction can be estimated by using the virial theorem and Eq. (6.9) as

$$\Delta P_{\text{tail}} = 2\pi\rho^2 \int_{r_c}^{\infty} \vec{r}\cdot\vec{f}(r) r^2 dr = \frac{16}{3}\pi\rho^2\epsilon\sigma^3 \left[\frac{2}{3}\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3\right]. \tag{6.10}$$

Note that this is the contribution that should be added to P if one wishes to estimate the pressure of the true (untruncated) potential, rather than the true pressure of the truncated potential[4]. This is a specific example of a more general lesson: one should always be aware of the consequences of any approximation that is introduced in the model, weighing in advantages and disadvantages and, if possible, finding a way to correct the results *a posteriori*, as in this case.

A common way of removing the divergence in Eq. (**??**) is to truncate and shift the potential:

$$V_{\text{tr,sh}}(r) = \begin{cases} 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6\right) - V(r_c) & \text{if } r \le r_c \\ 0 & \text{otherwise.} \end{cases} \tag{6.11}$$

This is especially important in constant-energy MD simulations (*i.e.* simulations in the NVE ensemble), since the discontinuity of the truncated (but not shifted) potential would greatly deteriorate the energy conservation. In this case the pressure tail correction remains the same, while the energy requires an additional correction on top of Eq. (6.9), which accounts for the average number of particles that are closer than r_c from a given particle, multiplied by $\frac{1}{2}V(r_c)$.

TODO
Add part about smoothing

Minimum image convention and periodic boundary conditions

The number of particles in a modern-day simulation ranges from hundreds to millions, thus being very far from the thermodynamic limit. Therefore, it is not surprising that finite-size effects are always present (at least to some extent). In particular, the smaller the system, the larger boundary effects are: since in 3D the volume scales as N^3 and the surface as N^2, the fraction of particles that are on the surface scales as $N^{-1/3}$, which is a rather slowly decreasing function of N. For instance, if $N = 1000$, in a cubic box of volume $V = L^3$ more than half of the particles are on the surface. One would have to simulate $\approx 10^6$ particles to see this fraction decrease below 10%!

Those pesky boundary effects can be decreased by using periodic boundary conditions (PBCs): we get rid of the surface by considering the volume containing the system, which is often but not always a cubic box, one cell of an infinite periodic lattice made of identical cells. Then, each particle i iteracts with any other particle: not only with those in the original cell, but also with their "images", including its own images, contained in all the other cells. For instance, the total energy of a (pairwise interacting) cubic system of side length L simulated with PBCs would be

$$U_{\text{tot}} = \frac{1}{2}{\sum_{i,j,\vec{n}}}' V(|\vec{r}_{ij} + \vec{n}L|), \tag{6.12}$$

where $\vec{r}_{ij}$ is the distance between particles i and j, $\vec{n}$ is a vector of three integer numbers $\in [-\infty, +\infty]$, and the prime over the sum indicates that the $i = j$ term should be excluded if $\vec{n} = (0, 0, 0)$. How do we handle such an infinite sum in a simulation? Keeping the focus on short-range interactions, it is clear that all terms with $|\vec{r}_{ij}+\vec{n}L| > r_c$ vanish, leaving only a finite number of non-zero interactions.

[4]Check Frenkel and Smit [2023] if you want an expression for the latter.

In practice, Eq. (6.12) is evaluated by making sure that $r_c < L/2$, so that each particle can interact **at most** with a single periodic image of another particle. Then, given any two particles i and j, the vector distance $\vec{r}_{ij} = (x_{ij}, y_{ij}, z_{ij})$ between the closest pair of images is

$$x_{ij} = x_j - x_i - \text{round}\left(\frac{x_j - x_i}{L_x}\right) L_x \tag{6.13}$$

$$y_{ij} = y_j - y_i - \text{round}\left(\frac{y_j - y_i}{L_y}\right) L_y \tag{6.14}$$

$$z_{ij} = z_j - z_i - \text{round}\left(\frac{z_j - z_i}{L_z}\right) L_z, \tag{6.15}$$

where, for the sake of completeness, I'm considering a non-cubic box of side lengths L_x, L_y and L_z, and round$(\cdot)$ is the function that rounds its argument to its closest integer. Figure 6.2 shows a 2D schematic of periodic-boundary conditions and of the minimum-image construction.

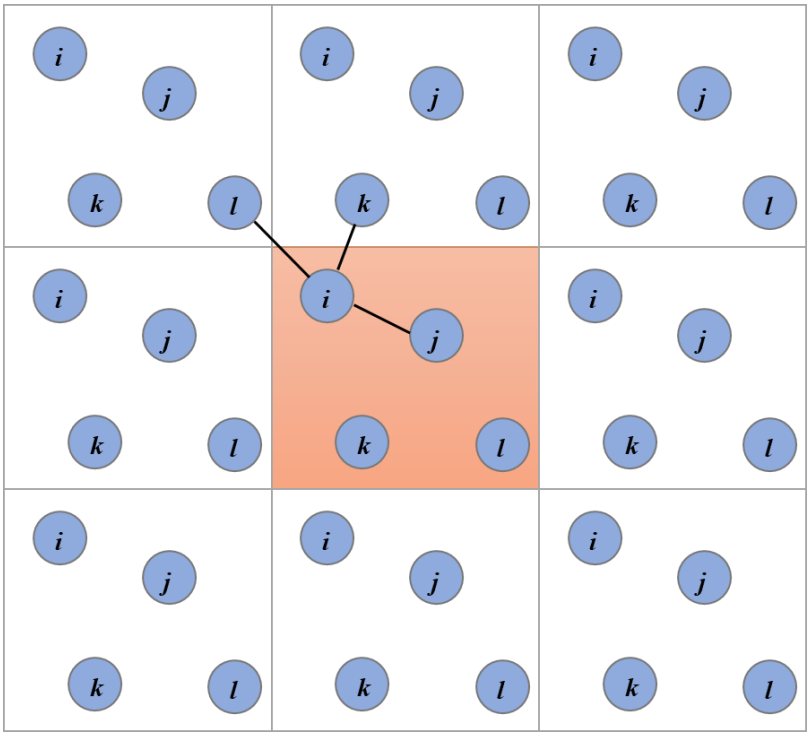

Figure 6.2: A schematic representation of periodic-boundary conditions and the minimum-image construction for a two-dimensional system. The system shown here contains 4 particles (labelled i, j, k, and l), with the black lines connecting particle i with the closest periodic images of the other particles. Credits to Abusaleh AA via Wikimedia Commons.

The total force acting on a particle i can now be computed by summing up the contributions due to each particle $j \neq i$, where the distance is given by Eq. (**??**).

The use of periodic boundary conditions gets rid of surface effects altogether, and more generally can greatly limit finite-size effects, even for small numbers of particles. However there are some important things that should always be kept in mind when using PBCs (which is by far the most common form of boundary conditions employed in molecular simulations):

1. If the system under study exhibit correlations that are of the same order of or exceed the box size, the system will "feel itself" through the PBCs, leading to spurious non-linear effects.

2. Only wavelengths compatible with the box size are allowed. This is true for any observable, and applies to a wide range of phenomena. For instance

   - If one wishes to simulate equilibrium crystals, the shape of the box and the length of its sides should be compatible with the lattice's unit cell.

- Fluctuations with wavelengths longer than the box size are suppressed. This makes it hard to study critical phenomena, which have diverging correlation lengths.

3. Angular momentum is not conserved, since the system is **not** rotationally invariant (even if the Hamiltonian is).

4. When particles cross the box boundary to, for instance, exit the central cell, one of its images will enter it. It is important to keep track of the "original" particle to correctly compute the value of some observables (*e.g.* the mean-squared displacements, see below, or a chain connectivity).

6.1.4 Integrating the equations of motion

> **Important**
> For the sake of simplicity, I will derive equations for 1D systems, but these remain valid for 2D and 3D systems, as the dynamics is decoupled along the three dimensions.

Consider a particle moving along the x-axis under the influence of a force $F(t)$. Newton's second law gives $m\frac{d^2x(t)}{dt^2} = F(t)$, so that the acceleration $a(t)$ is $a(t) = \frac{F(t)}{m}$

Expanding the position $x(t)$ around the time t using a Taylor series we obtain

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 + \frac{1}{6}\frac{da(t)}{dt}\Delta t^3 + \mathcal{O}(\Delta t^4) \tag{6.16}$$

$$x(t-\Delta t) = x(t) - v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 - \frac{1}{6}\frac{da(t)}{dt}\Delta t^3 + \mathcal{O}(\Delta t^4) \tag{6.17}$$

Adding the two Taylor expansions yields

$$x(t+\Delta t) + x(t-\Delta t) = 2x(t) + a(t)\Delta t^2 + \mathcal{O}(\Delta t^4), \tag{6.18}$$

which, if we neglect the higher-order terms $\mathcal{O}((\Delta t)^4)$ becomes

$$x(t+\Delta t) = 2x(t) - x(t-\Delta t) + a(t)\Delta t^2. \tag{6.19}$$

This is the Verlet algorithm, which allows us to calculate the position $x(t+\Delta t)$ at the next time step using the current position $x(t)$, the previous position $x(t-\Delta t)$, and the current acceleration $a(t)$. Although this basic Verlet method does not explicitly involve velocity, if we consider the expansion up to the second order it can be written as

$$v(t) = \frac{x(t+\Delta t) - x(t-\Delta t)}{2\Delta t} + \mathcal{O}(\Delta t^2). \tag{6.20}$$

Note the lower order of accuracy with respect to $x(t)$. We can modify the basic Verlet approach to include an explicit update for the velocity, leading to the Velocity Verlet method. Instead of relying on the positions from the previous and current time steps, the Velocity Verlet algorithm updates the position and velocity in a two-step process.

First, we use the current velocity and acceleration to update the position at time $t+\Delta t$. This is done similarly to the basic Verlet method but with the velocity term explicitly included:

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 \tag{6.21}$$

This equation uses the current position $x(t)$, the current velocity $v(t)$, and the current acceleration $a(t)$ to compute the new position $x(t+\Delta t)$. Next, after updating the position, we need to compute the new acceleration at time $t+\Delta t$ because the force (and hence the acceleration) may have changed due to the updated position. The new acceleration is given by:

$$a(t+\Delta t) = \frac{F(t+\Delta t)}{m} \tag{6.22}$$

With this new acceleration in hand, we can update the velocity. Instead of using just the current acceleration, the Velocity Verlet method uses the average of the current and new accelerations to update the velocity:

$$v(t+\Delta t) = v(t) + \frac{1}{2}\left[a(t) + a(t+\Delta t)\right]\Delta t \tag{6.23}$$

This velocity update equation accounts for the change in acceleration over the time step, providing a more accurate velocity update than simply using the current acceleration. The Velocity Verlet method is the *de-facto* standard for MD codes. The common way to implement it is to split the velocity integration step in two, so that a full MD interation becomes

1. Update the velocity, first step, $v(t+\Delta t/2) = v(t) + \frac{1}{2}a(t)\Delta t$.
2. Update the position, $x(t+\Delta t) = x(t) + v(t+\Delta t/2)\Delta t = x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2$ (*e.g.* Eq. (6.21)).
3. Calculate the force (and therefore the acceleration) using the new position, $x(t+\Delta t) \rightarrow a(t+\Delta t) = F(t+\Delta t)/m$.
4. Update the velocity, second step, $v(t+\Delta t) = v(t+\Delta t/2) + \frac{1}{2}a(t+\Delta t)\Delta t = v(t) + \frac{1}{2}\left[a(t) + a(t+\Delta t)\right]\Delta t$ (*e.g.* Eq. (6.23)).

An MD implementation based on Program 6.1 that implements a Velocity Verlet algorithm is

```
CALL initialise_parameters()
CALL initialise_system()

WHILE t is smaller than t_max
   CALL integrate_first_step()
   CALL compute_force()
   CALL integrate_second_step()

   t += delta_t
   CALL sample_observables()
```

Figure 6.3: Pseudo-code for an MD code implementing the Velocity Verlet algorithm.

6.1.5 Energy conservation

The Velocity Verlet algorithm is *symplectic*, which means that it conserves the energy (*i.e.* the value of the Hamiltonian)[5]. This is not a common property, as most schemes (such as the explicit Euler scheme or the famous Runge-Kutta method) are not symplectic, and therefore generate trajectories that do not live on the $H = \text{const}$ hypersurface. Since this is where the dynamics of systems evolving through Newton's equations move, MD integrators should always be symplectic. If this is the case, the resulting molecular dynamics simulations will conserve the total energy, which is a constant of motion. In turn, this means that MD simulations sample the microcanonical ensemble (as long as the system is ergodic, so that time and ensemble averages are equivalent).

Since we are discretising the equations, there are two caveats associated to the energy conservation:

1. If the time step Δt is too large, then a deviation (drift) from the "correct" value of the energy will be observed. A good rule of thumb to avoid an energy drift, which would invalidate the simulation, is to calculate the highest vibrational frequency associated to the interaction potential employed, $\omega_c \equiv \sqrt{k_c/m}$, where k_c is the highest curvature of the potential, *i.e.* the largest value of $d^2V(r)/dr^2$, and m is the mass of the particle. Then, a sensible value for the integration time step is an order of magnitude less than the characteristic time associated to ω_c, *i.e.* $\Delta t \approx \frac{1}{10}\frac{2\pi}{\omega_c}$.

[5] it is also time-invariant and conserves volumes in phase space.

2. If the value of Δt is appropriate, then the energy will be conserved *on average*, meaning that it will fluctuate around its average value with an amplitude, $\delta U \equiv \sqrt{\langle \Delta U^2 \rangle}$. If the integrator is of the Verlet family (*i.e.* Velocity Verlet), then $\delta U \sim \Delta t^2$. This behaviour can be exploited to ensure that codes and simulations are working correctly, at least from the point of view of the energy conservation. See Figure 6.4 for an example.

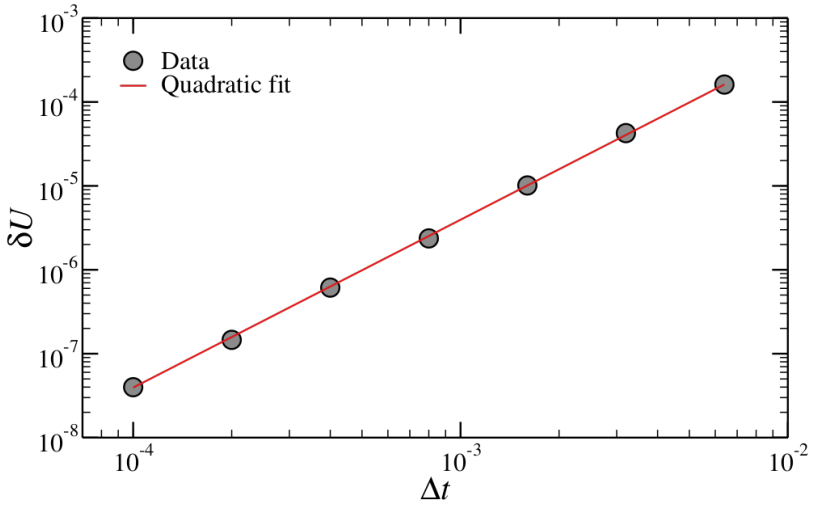

Figure 6.4: The extent of the fluctuations of the total energy, δU, for a Lennard-Jones system simulated at $k_B T/\epsilon = 1.5$ and $\rho\sigma^3 = 0.36$ as a function of the time step Δt. The line is a quadratic fit. **Nota Bene:** this scaling requires that truncation errors are small, which is why I had to use a rather large cut-off, $r_c = 3.5$.

6.1.6 Some observables

Since, in principle, we have access to the whole phase space, in MD simulations we can compute averages of any (classical) observable. Here I present a short list of some useful (and common) quantities.

Pressure

> **Tip**
> The main source for this part is Allen and Tildesley [2017].

In the canonical ensemble, for any observable A and generalised coordinate or momentum h_k, integrating by parts (and assuming reasonable boundary conditions) one finds

$$\left\langle \frac{\partial A}{\partial h_k} \right\rangle = \frac{1}{Q} \int \frac{\partial A}{\partial h_k} \exp(-\beta H) d\{h_k\} = \frac{1}{Q} \int \beta A \frac{\partial H}{\partial h_k} \exp(-\beta H) d\{h_k\} \tag{6.24}$$

$$= \beta \left\langle A \frac{\partial H}{\partial h_k} \right\rangle, \tag{6.25}$$

which translates to the following generalised equipartition relation:

$$\left\langle h_k \frac{\partial H}{\partial h_k} \right\rangle = k_B T. \tag{6.26}$$

If we plug in a generalised momentum and sum over all momenta, Eq. (6.26) yields the usual equipartition principle:

$$\left\langle \sum_{i=1}^{N} \frac{|\vec{p}_i|^2}{m_i} \right\rangle = 3Nk_BT, \tag{6.27}$$

where m_i is the mass of the i-th particle. By contrast, if we choose to use Cartesian coordinates as generalised coordinates in Eq. (6.26) and recall that the derivative of the Hamiltonian with respect to a particle coordinate is minus the total force acting on the particle along that coordinate, we find

$$\left\langle \sum_{i=1}^{N} \vec{r}_i \cdot \vec{F}_i^{\text{tot}} \right\rangle = -3Nk_BT, \tag{6.28}$$

where $\vec{r}_i$ is the position of particle i. Note that here $\vec{F}_i^{\text{tot}}$ is the sum of inter-molecular interactions, $\vec{F}_i^{\text{int}}$, and external forces, $\vec{F}_i^{\text{ext}}$. If the latter consist only of the forces exerted by the container walls (which keeps the system in a volume V under a pressure P), we have

$$\frac{1}{3}\left\langle \sum_{i=1}^{N} \vec{r}_i \cdot \vec{F}_i^{\text{ext}} \right\rangle = -PV. \tag{6.29}$$

Since $\vec{F}_i^{\text{tot}} = \vec{F}_i^{\text{int}} + \vec{F}_i^{\text{ext}}$, we find

$$P = \frac{Nk_BT}{V} + \frac{1}{3V}\left\langle \sum_{i=1}^{N} \vec{F}_i^{\text{int}} \cdot \vec{r}_i \right\rangle \equiv \langle P_{\text{inst}} \rangle, \tag{6.30}$$

where I have defined the instantaneous pressure P_{inst}. The second term in Eq. (6.30) represents the contribution from interparticle forces, where $\vec{F}_i^{\text{int}}$ is the force on particle i due to all other particles. Applying Eq. (6.30) makes it possible to evaluate the pressure in computer simulations.

Radial distribution function

There are many ways to obtain structural information from simulation output. The simplest, and most straightforward, piece of such information is encoded in the pair correlation function, also known as the radial distribution function, $g(r)$.

Let us start by defining a function $n(r)$ such that $n(r)dr$ represents the number of particles found between r and $r + dr$, assuming the origin is placed at a random particle. With this definition, the total number of particles contained within a sphere of radius R centered on a random particle is:

$$N(R) = \int_0^R n(r)dr. \tag{6.31}$$

For an ideal gas, $n(r) = 4\pi r^2\rho$, where ρ is the number density $(N-1)/V$.

The function $g(r)$ is defined as:

$$g(r) \equiv \frac{n(r)}{4\pi r^2\rho}, \tag{6.32}$$

and it indicates how much denser the system is at a distance r compared to an ideal gas. The function $g(r)$ is always positive and, for large values of r, $g(r) \to 1$. The $g(r)$ is a central observable in molecular simulations, and can be used to estimate many other observables (see *e.g.* Hansen and McDonald [2013]).

Figure 6.5 shows a typical shape of $g(r)$, together with a 2D cartoon that gives an idea of how the structural features of the system affects the resulting radial distribution function. Near the origin, $g(r) = 0$ due to the excluded volume. This is always observed in simple liquids and, more generally, in systems with short-range repulsions (such as atom-based systems). For soft (or ultrasoft) potentials, which are typically effective, $g(r)$ may not be zero even at the origin.

Figure 6.5 also conveys a second message: the packing of objects with excluded volume inevitably generates oscillations in the $g(r)$ function, with a periodicity determined by the diameter of the particles themselves.

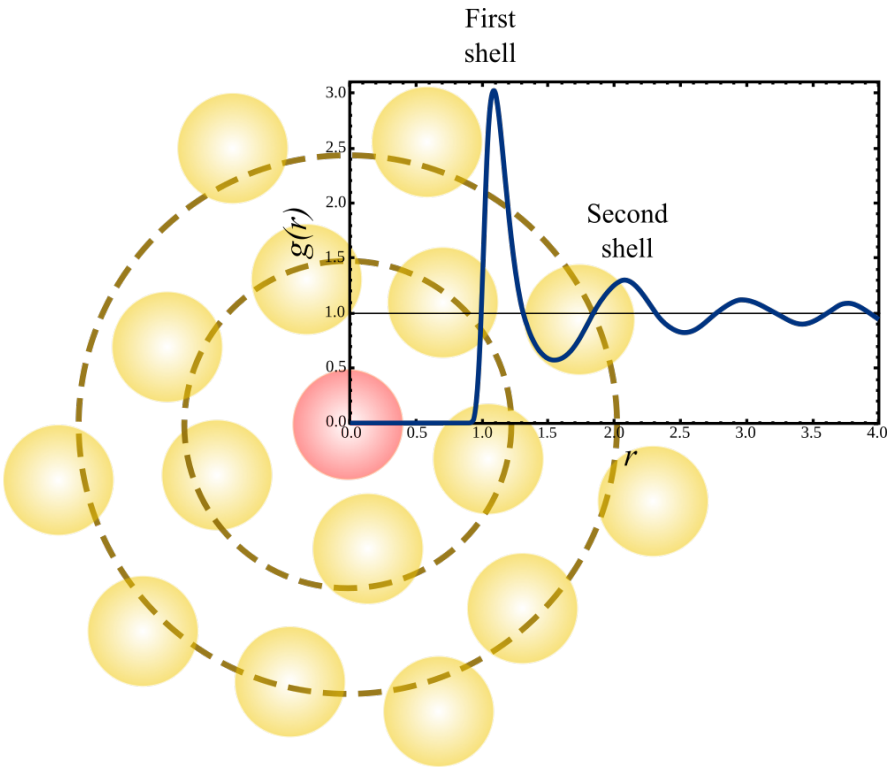

Figure 6.5: The radial distribution function of a Lennard-Jones fluid at $k_BT/\epsilon = 0.71$ and $\rho\sigma^3 = 0.844$. Superimposed on the plot is a cartoon that shows the interpretation of the features of the $g(r)$, highlighting the first and second shells around the central particle (coloured in red). Adapted from Wikipedia.

In a system composed by atoms (or particles) of different types, it is possible to define radial distribution functions between pairs of atom types. For instance, when simulating water it is common to define $g_{OO}(r)$, $g_{HH}(r)$, and $g_{HO}(r)$, which provide information on the oxygen-oxygen, hydrogen-hydrogen and hydrogen-oxygen pair correlations, respectively. See Figure 6.6 for an example.

Mean-squared displacement

The mean squared displacement (MSD) quantifies the extent of particle diffusion within a system over time. It provides insights into the mobility of particles by tracking how far each particle moves from its original position. Formally, for a given particle i, the MSD at time t is calculated as the average of the squared differences between its position $\vec{r}_i(t)$ at time t and its initial position $\vec{r}_i(0)$, represented as $\text{MSD}_i(t) = \langle|\vec{r}_i(t) - \vec{r}_i(0)|^2\rangle$, so that the overall MSD is

$$\text{MSD}(t) = \frac{1}{N}\sum_{i=1}^{N}\langle|\vec{r}_i(t) - \vec{r}_i(0)|^2\rangle \tag{6.33}$$

In MD simulations, the MSD can be computed by first recording the position of each particle at each timestep, then calculating the squared displacement for each particle relative to its starting position, and finally averaging this value over all particles. If the system is ergodic, as it is often the case, we can also average over different time origins to decrease the statistical error. This is done by averaging contributions of the type $|\vec{r}_i(t_0 + t) - \vec{r}_i(t_0)|^2$ for multiple values of t_0.

The MSD provides essential information on diffusive behavior, as its time dependence can help distinguish between different diffusion regimes, such as ballistic ($\text{MSD}(t) \sim t^2$), brownian ($\text{MSD}(t) \sim t$), or subdiffusive (*e.g.* $\text{MSD}(t) \sim t^\alpha$, with α <1), which are characteristic of different types of material and dynamical properties. Figure 6.7 provides an example of a system that, depending on temperature, experience all these regimes.

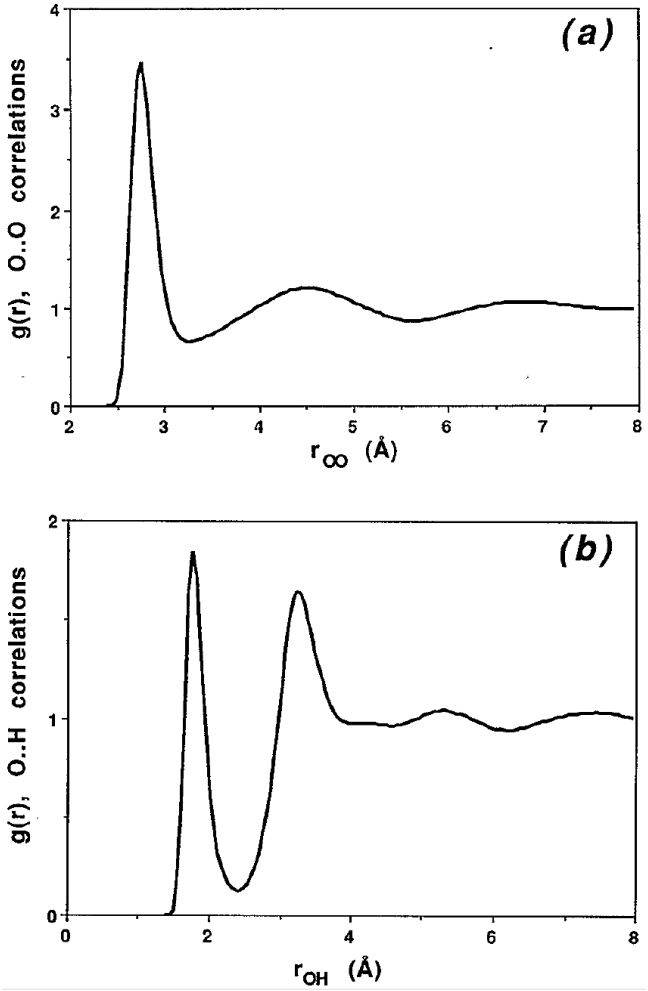

Figure 6.6: The (a) $O-O$ and (b) $O-H$ radial distribution functions of SPC/E liquid water at $-10°$ C. Taken from Svishchev and Kusalik [1993].

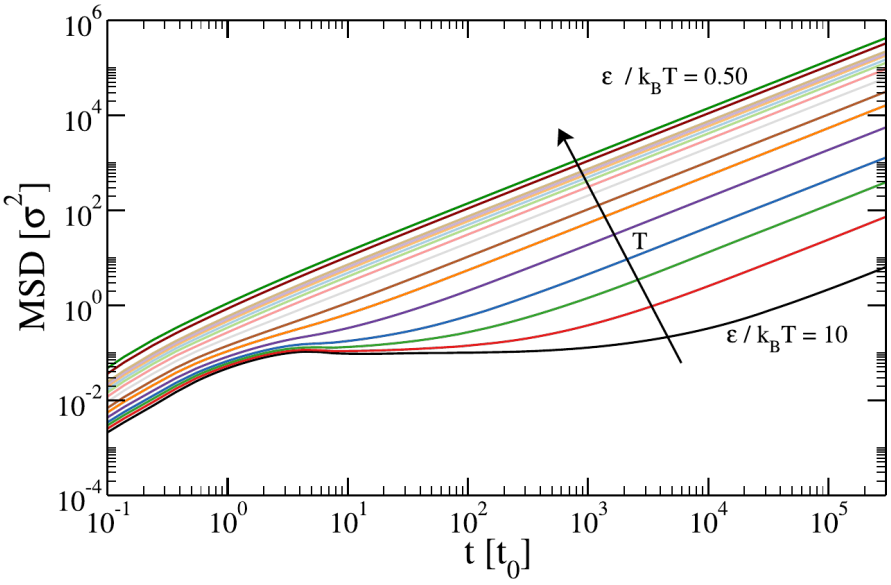

Figure 6.7: The mean-squared displacement of a coarse-grained system (a tetravalent patchy particle model) computed at fixed density and varying ratio between the attraction strength ϵ (kept constant) and the thermal energy, k_BT. Taken from Gomez and Rovigatti [2024].

Root mean-squared deviation

The root mean-squared deviation (RMSD) is a measure commonly used to assess the structural deviation of a molecule or a set of particles from a reference structure, typically the initial or an experimentally determined structure. It quantifies the average distance between corresponding atoms in two structures, giving insight into conformational changes over time. The RMSD is particularly useful in studying the stability and flexibility of molecular structures, such as proteins, over the course of a simulation.

Formally, for a system with N atoms, the RMSD at time t relative to a reference structure is given by:

$$\mathrm{RMSD}(t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} |\vec{r}_i(t) - \vec{r}_i^{\,\mathrm{ref}}|^2} \tag{6.34}$$

where $\vec{r}_i^{\,\mathrm{ref}}$ is the position of the i-th atom in the reference structure. To compute the RMSD in MD simulations Eq. (6.34) is applied after that the structure at each time point is aligned with the reference structure (typically minimizing rotational and translational differences). Formally, this can be written as

$$\mathrm{RMSD}(t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} |\hat{R}\cdot\vec{r}_i(t) + \vec{t} - \vec{r}_i^{\,\mathrm{ref}}|^2}, \tag{6.35}$$

where $\hat{R}$ and $\vec{t}$ are the rotation matrix and translation vector that minimise the resulting RMSD, respectively. The most common use of the RMSD is to plot it as a function of time to analyse the stability of the molecule: lower RMSD values indicate that the structure remains close to the reference, while higher values suggest significant conformational changes.

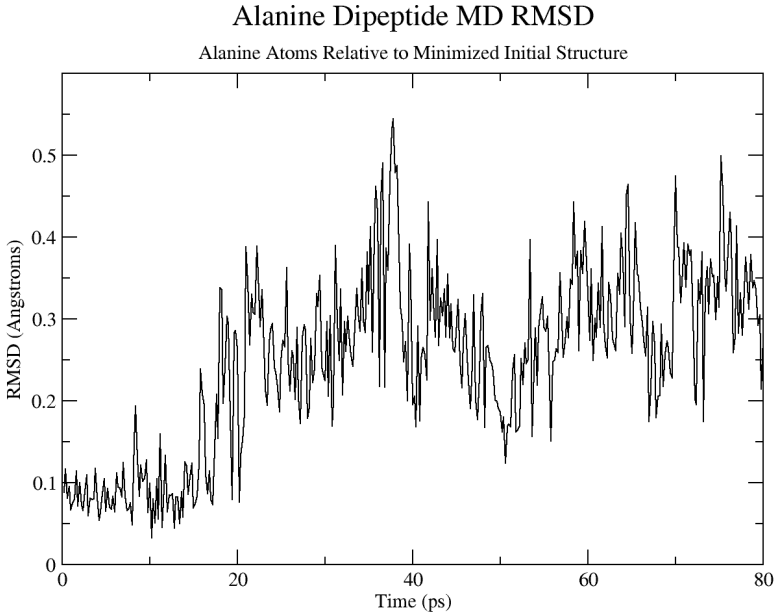

Figure 6.8: The RMSD of a Alanine dipeptide simulated with Amber. Taken from this Amber tutorial.

Figure 6.8 shows the RMSD of a simulation of a small molecule (an Alanine dipeptide). In this example the small value of the RMSD throughout the trajectory shows that there is no significant conformational change in the positions of the atoms relative to the starting structure.

6.1.7 Tricks of the trade

Neighbour lists

If the interaction potential is short-ranged, in the sense that it goes to zero at some distance r_c, it is clear that particles that are further away than r_c will not feel any reciprocal force. If this is the case, calculating distances between all pairs of particles would be wasteful, as only a fraction of pairs will feel a mutual interaction. There are several techniques that can be used to optimise the force calculation step (and therefore the simulation performance) by performing some kind of bookkeeping that makes it possible to evaluate only those contributions due to pairs that are "close enough" to each other. Here I will present the most common ones: cells and Verlet lists.

The idea behind cell lists is to partition the simulation box into smaller boxes, called cells, with side lengths $\geq r_c$[6]. This partitioning is done by using a data structure that, for each cell, stores the list of particles that are inside it. Then, during the force calculation loop, we consider that particle i, which is in cell c, can interact only with particles that are either inside c or in one of its neighbouring cells (8 in 2D and 26 in 3D). The cell data structure can either be built every step, or updated after each integration step. In this latter case, the code checks whether a particle has crossed a cell boundary, and in this case it removes the particle from the old cell and adds it to the new one. This can be done efficiently with linked lists. With this technique, each particle has a number of possibly-interacting neighbours that depends only on particle density and cell size, and therefore is independent on N. As a result, the algorithmic complexity of the simulation is $\mathcal{O}(N)$ rather than $\mathcal{O}(N^2)$.

Differently from cell lists, in Verlet lists for each particle the code stores a list of particles that are within a certain distance $r_v = r_c + r_s$, where r_s is a free parameter called "Verlet skin". Every time the lists are updated, the current position of each particle i, $\vec{r}_{i,0}$ is also stored. During the force calculation step of particle i, only those particles that are in i's Verlet list are considered. For a homogeneous system of density ρ, the average number if neighbours is

$$N_v = \frac{4}{3}\pi r_v^3 \rho, \tag{6.36}$$

which should be compared to the average number of neighbours if cell lists are used, which in three dimensions is

$$N_c = 27 r_c^3 \rho. \tag{6.37}$$

In the limit $r_s \ll r_c$, which is rather common, $r_v \approx r_c$, so that $N_c/N_v = 81/4\pi \approx 6$, which means that if Verlet lists are used, the number of distances to be checked is six times smaller than with cell lists.

Verlet lists do not have to be updated at every step, or we would go back to checking all pairs, but only when when any particle has moved a distance larger than $r_s/2$ from its original position $\vec{r}_{i,0}$. Note that the overall performance will depend on the value of r_s, as small values will result in very frequent updates, while large values will generate large lists, which means useless distance checks on particles that are too far away to interact. Although the average number of neighbours of each particle i is independent on N and therefore the actual force calculation is $\mathcal{O}(N)$, the overall algorithmic complexity is still $\mathcal{O}(N^2)$, since list updating, even if not done at each time step, requires looping over all pairs. To overcome this problem it is common to use cell lists to build Verlet lists, bringing the complexity of the list update step, and therefore of the whole simulation, down to $\mathcal{O}(N)$, while at the same time retaining the smaller average number of neighbours of Verlet lists.

Long-range interactions

[6] In principle, cells don't have to be cubic.

Tip
Most of the text of this part comes from Frenkel and Smit [2023], while some bits have been adapted from Schlick [2010].

In molecular simulations, long-range interactions refer to forces that decay slowly with distance. The definition of "long-range" can be made unambiguous if we consider the general form of a pairwise interaction potential $V(r)$ between two particles separated by a distance r. The energy contribution of these interactions beyond a certain cut-off distance r_c is given by the "tail" correction of (6.7). For a potential that decays as $1/r^\alpha$, the tail correction is, asymptotically,

$$U_{\text{tail}} \sim \int_{r_c}^{\infty} \frac{r^2}{r^\alpha}\, dr = \int_{r_c}^{\infty} \frac{1}{r^{\alpha-2}}\, dr \sim \left.\frac{1}{r_c^{\alpha-3}}\right|_{r_c}^{\infty} \tag{6.38}$$

which converges for any r_c as long as $\alpha > 3$ (*e.g.* the Lennard-Jones potential, for which $\alpha = 6$)[7]. By contrast, if $\alpha < 3$, the integral diverges, indicating that the energy contribution from the interaction tail remains significant when any cut-off is applied: the tail of the interaction potential beyond r_c contributes non-negligibly to the total energy of the system, making a direct cut-off inaccurate.

I'll now quickly introduce the most popular methods used to handle long-range interactions, together with two techniques used to improve its efficiency. First, we start by considering a system consisting of N charged particles in a box of volume $V = L^3$ and periodic boundary conditions. We assume that particles cannot overlap (*i.e.* that there is at least an additional short-range repulsion), and that the system is electrically neutral, *e.g.* that $\sum_i q_i = 0$. The total electrostatic energy of the system (in Gaussian units, which make the notation lighter) is

$$U_{\text{el}} = \frac{1}{2} {\sum_{i,j,\vec{n}}}' \frac{q_i q_j}{|\vec{r}_{ij} + \vec{n}L|}, \tag{6.39}$$

where the prime on the summation indicates that the sum is over all periodic images $\vec{n}$ and over all particle pairs (i, j), except $i = j$ if $\vec{n} = (0, 0, 0)$, *i.e.* particle i interacts with all its periodic images but not with itself. Unfortunately, Eq. (6.39) cannot be used to compute the electrostatic energy in a simulation because it contains a conditionally convergent sum.

To improve the convergence of the expression for the electrostatic potential energy, we follow Ewald [1921] and rewrite the expression for the charge density. In equation (6.39) we have represented the charge density as a sum of δ-functions. The contribution to the electrostatic potential due to these point charges decays as $1/r$. Now consider what happens if we assume that every particle i with charge q_i is surrounded by a diffuse charge distribution of the opposite sign, such that the total charge of this cloud exactly cancels q_i . In that case only the fraction of q_i that is not screened contributed to the electrostatic potential due to particle i. At large distances, this fraction goes to 0 in a way that depends on the functional form of the screening charge distribution, which we will take as Gaussian in the following.

The contribution to the electrostatic potential at a point r due to a set of screened charges can be easily computed by direct summation, because the electrostatic potential due to a screened charge is a rapidly decaying function of r. However, it was not our aim to evaluate the potential due to a set of screened charges but due to point charges. Hence, we must correct for the fact that we have added a screening charge cloud to every particle. This is shown schematically in Figure 6.9. This compensating charge density varies smoothly in space (because the screening charge distribution is a smoothly varying function!). We wish to compute the electrostatic energy at the site of ion i. Of course, we should exclude the electrostatic interaction of the ion with itself. We have three contributions to the electrostatic potential: first of all, the one due to the point charge q_i, secondly, the one due to the Gaussian screening charge cloud with charge $-q_i$, and finally the one due to the compensating charge cloud with charge q_i. In order to exclude Coulomb self-interactions, we should not include any of these three contributions to the electrostatic potential at the position of ion i. However, it turns out that it

[7]Note that if $\alpha = 3$ (*e.g.* a dipole-dipole interaction), the tail correction is $U_{\text{tail}} \sim [\log r]_{r_c}^{\infty}$, which diverges.

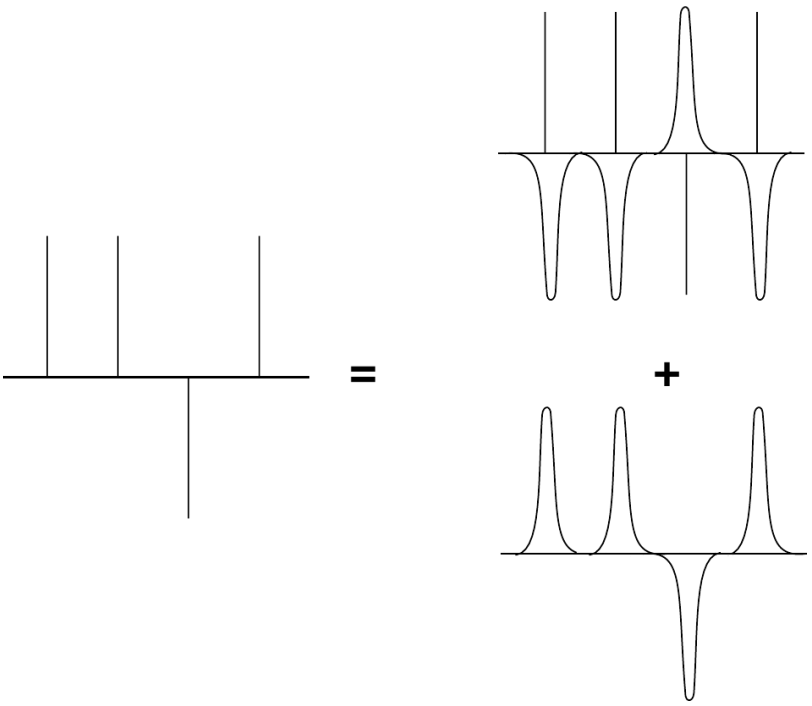

Figure 6.9: The elecrostatic effect due to point charges can be seen as a sum of screend charges, minus the smoothly varying screening background. Taken from Frenkel and Smit [2023].

is convenient to retain the contribution due to the compensating charge distribution and correct for the resulting spurious interaction afterwards. The reason we retain the compensating charge cloud for ion i is that, if we do so, the compensating charge distribution is not only a smoothly varying function, but it is also periodic. Such a function can be represented by a (rapidly converging) Fourier series, and this turns out to be essential for the numerical implementation. Of course, in the end we should correct for the inclusion of a spurious "self" interaction between ion and the compensating charge cloud.

Considering screening Gaussians with width $\sqrt{2/\alpha}$, the splitting can be written as follows:

$$\frac{1}{|\vec{r}_i - \vec{r}_j|} = \frac{\mathrm{erfc}(\alpha|\vec{r}_i - \vec{r}_j|)}{|\vec{r}_i - \vec{r}_j|} + \frac{\mathrm{erf}(\alpha|\vec{r}_i - \vec{r}_j|)}{|\vec{r}_i - \vec{r}_j|}, \tag{6.40}$$

where α is a parameter that controls the width of the Gaussian distribution used to split the potential, and $\mathrm{erfc}(x)$ and $\mathrm{erf}(x)$ are the complementary error function and error function, respectively, and they come out from integrating the Gaussian screening function. In Eq. (6.40), the first term decays quickly as $|\vec{r}_i - \vec{r}_j|$ increases, so it is computed only for nearby particles within a cut-off r_c. By contrast, the second term decays slowly in real space, but can be computed efficiently in Fourier space, using a sum over reciprocal lattice vectors $\vec{k}$.

The total electrostatic energy using Ewald summation is the sum of the real-space term, the reciprocal-space term, and the self-interaction correction:

$$E_{\mathrm{tot}} = E_{\mathrm{real}} + E_{\mathrm{rec}} + E_{\mathrm{self}}. \tag{6.41}$$

- The real-space contribution to the total energy, E_{real} is given by:

$$E_{\mathrm{real}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} \frac{q_i q_j \,\mathrm{erfc}(\alpha|\vec{r}_i - \vec{r}_j|)}{|\vec{r}_i - \vec{r}_j|}, \tag{6.42}$$

  where the sum is truncated at a cut-off distance r_c.
- The reciprocal-space contribution E_{rec} is computed using the Fourier transform of the charges and involves a sum over the reciprocal lattice vectors $\vec{k}$, which can be safely truncated at some cut-off wave vector k_c, since the exponential term ensures that the sum converges rapidly:

$$E_{\text{rec}} = \frac{1}{2V} \sum_{\vec{k} \neq 0} \frac{4\pi}{k^2} \exp\left(-\frac{k^2}{4\alpha^2}\right) \left| \sum_{j=1}^{N} q_j \exp(i\vec{k} \cdot \vec{r}_j) \right|^2 . \tag{6.43}$$

- The self-interaction term, E_{self}, is:

$$E_{\text{self}} = -\frac{\alpha}{\sqrt{\pi}} \sum_{i=1}^{N} q_i^2. \tag{6.44}$$

Note that there are many subtleties that are linked to the boundary conditions that are applied to the system (even though the latter is infinite!). Have a look at Frenkel and Smit [2023] and Schlick [2010] (and references therein) for additional details.

For fixed values of k_c and r_c, the algorithmic time scales as $\mathcal{O}(N^2)$. However, this behaviour can be improved by realising that there are values of the cut-offs that minimise the error due to the truncations. Using these values, that depend on N, the complexity can be brought down to $\mathcal{O}(N^{3/2})$.

A further scaling improvement can be obtained by using the so-called Particle Mesh Ewald method (Darden et al. [1993]). In the PME method, the reciprocal space contribution to electrostatic interactions is computed through the following series of steps that involve transforming particle charges into a charge density on a grid, applying Fourier transforms, and then transforming back to obtain the forces and energies:

1. Assign each particle's charge to a regular 3D grid that spans the simulation box. This is achieved through an interpolation scheme, where each particle's charge is "spread" over multiple neighboring grid points. The most common method is B-spline interpolation (see Essmann et al. [1995] for details).

2. Once the charges are mapped onto the grid, the Fourier transform of the grid-based charge density is computed. This is done by using the Fast Fourier Transform (FFT), an efficient algorithm for performing discrete Fourier transforms, with a computational complexity of $\mathcal{O}(N \log N)$.

3. In Fourier space, the electrostatic potential for each grid point at wave vector is computed using the Ewald screening function (*i.e.* Eq. (6.43)).

4. Once the reciprocal-space potential has been calculated, an inverse Fourier transform is applied to convert the potential back to real space on the grid, yielding the electrostatic potential on each grid point in the simulation box.

5. The final step is to interpolate the grid-based potential back to the positions of the actual particles, which allows the forces and energies to be computed for each particle based on the smoothed potential field. This is essentially the reverse of the initial charge assignment process, with each particle's force interpolated from the neighboring grid points.

If the cut-off of the real-space contribution is chosen so that E_{real} can be computed in $\mathcal{O}(N)$, the overall complexity of the PME method is $\mathcal{O}(N \log N)$, which is **much** better than $\mathcal{O}(N^{3/2})$. The method has some constant overhead that makes it slower than the original Ewald sums only for small systems, which is why PME is the *de-facto* standard of biomolecular simulations.

Note that it is possible to achieve linear scaling ($\mathcal{O}(N)$) using sophisticated techniques such as the Fast Multipole Method (see *e.g.* Greengard and Rokhlin [1987], but those are rather difficult to implement, and not widely used yet.

6.2 Other ensembles

We know from statistical mechanics that, in the thermodynamic limit, all ensembles are equivalent. However, in simulations it can be convenient to use different ensembles, depending on the phenomena one wishes to study. I will now briefly present some methods that can be used to fix the temperature (rather than energy) and the pressure (rather than volume) in MD simulations.

6.2.1 Thermostats

Before introducing some of the schemes that are used to perform Molecular Dynamics simulations at constant temperature, it is important to understand what "constant temperature" even means. From a statistical mechanical point of view, there is no ambiguity: we can impose a temperature on a system by bringing it into thermal contact with a large heat bath. Under those conditions, the distribution of the velocities is given by the Maxwell-Boltzmann distribution (Eq. (6.4)), whose second moment is connected to the temperature *via* Eq. (6.2). Given in this form, it is clear that the temperature also fluctuates, with the fluctuations being linked to the second and fourth moment of the Maxwell-Boltzmann distribution (*e.g.* $\propto \langle v^4 \rangle - \langle v^2 \rangle^2$). Therefore, any algorithm devised to fix the temperature of a MD simulation should reproduce not only the average T, but also its fluctuations. Here I will present three such algorithms.

No more energy conservation!
As soon as a thermostat is coupled to the system, the energy will not be conserved any more. Therefore, we have one fewer way of testing the software, or the simulation parameters (*e.g.* the time step) we have chosen to use. My advice is to always turn off the thermostat and run a short simulation to see whether everything looks in order or not.

Andersen

The Andersen thermostat is a widely used method in molecular dynamics simulations for controlling temperature, introduced by Andersen [1980]. Its fundamental idea is to maintain a system's temperature by coupling the particles to an external heat bath through random collisions. In this approach, particles in the simulation periodically undergo stochastic collisions with a fictitious heat bath, resulting in velocity reassignment according to a Maxwell-Boltzmann distribution that corresponds to the desired temperature. This random reassignment of velocities, which can be considered as a Monte Carlo move that transports the system from one constant-energy shell to another, mimics the effect of a thermal reservoir, ensuring that the system reaches and maintains thermal equilibrium.

The thermostat has a parameter ν that is the frequency of stochastic collisions, which represents the strength of the coupling to the heat bath: by adjusting this collision frequency, the user can control how frequently the system interacts with the heat bath, thus influencing the rate at which the system equilibrates. In practice, at each time step each particle has a probability $\nu \Delta t$ of undergoing a collision, *i.e.* of being reassigned a velocity from a Maxwell-Boltzmann distribution corresponding to the target temperature. This scheme reproduced the canonical ensemble, since temperature fluctuations align with those expected from statistical mechanics.

One drawback of the Andersen thermostat is connected to its stochastic nature: the random collisions "disturb" the dynamics in a way that is not realistic. In particular, they enhance the decorrelation of the particle velocities, thereby affecting quantities such as the diffusion constant. This effect depends on the value of ν: the larger the collision frequency, the faster the decorrelation, the smaller the diffusion constant. Therefore, the Andersen thermostat should be used when we are interested in static properties only, and dynamics observables are not important.

Nose-Hoover

The Nosé-Hoover thermostat is a more sophisticated method for controlling temperature in molecular dynamics simulations, designed to generate the canonical ensemble while preserving the continuous evolution of the system's dynamics. Unlike the Andersen thermostat, which introduces stochastic velocity reassignment, the Nosé-Hoover thermostat operates deterministically by coupling the system to a fictitious heat bath via an extended Lagrangian formalism. This allows the system to exchange energy with the thermostat in a smooth and continuous manner, maintaining temperature without introducing artificial randomness.

The extended Lagrangian formalism starts by introducing a scaling factor, s, that modifies the velocities of particles in the system (Nosé [1984]). The extended Lagrangian for the Nosé-Hoover thermostat is expressed as:

$$L = \sum_{i=1}^{N} \frac{m_i}{2} \left(\frac{\dot{\vec{r}}_i}{s} \right)^2 - V(\{\vec{r}_i\}) - g k_B T \log s, \tag{6.45}$$

where m_i and $\dot{\vec{r}}_i = \vec{v}_i$ are the mass and velocity particle i, $V(\{r\})$ is the potential energy of the system, and the term $g k_B T \ln(s)$, where g is the number of degrees of freedom, introduces the necessary coupling between the system and the heat bath. The auxiliary variable s is responsible for controlling the temperature by scaling the velocities of the particles so that the system's kinetic energy corresponds to the target temperature.

The dynamics of the system are then derived from this Lagrangian. The equations of motion for the positions and velocities of the particles are modified by the thermostat, resulting in:

$$\ddot{\vec{r}}_i = \frac{\vec{F}_i}{m_i} - \zeta \dot{\vec{r}}_i \tag{6.46}$$

where $\vec{F}_i$ is the force acting on particle i, and the term $\zeta \equiv \dot{s}/s$ is a friction-like coefficient that emerges from the thermostat variable. This friction term adjusts the particle velocities in response to deviations from the target temperature, ensuring that the system remains at the correct thermal equilibrium, and evolves according to:

$$\dot{\zeta} = \frac{1}{Q} \left(\sum_{i=1}^{N} \frac{\vec{p}_i^2}{m_i} - g k_B T \right), \tag{6.47}$$

where Q is a parameter that controls the strength of the coupling between the system and the thermostat, $\vec{p}_i = m\dot{\vec{r}}_i$ is the momentum conjugate to $\vec{r}_i$, so that $\sum_{i=1}^{N} \frac{\vec{p}_i^2}{m_i}$ is the total kinetic energy of the system, and $g k_B T$ represents the target thermal energy. If the system's kinetic energy exceeds the desired value, the variable ζ increases, effectively damping the particle velocities to bring the temperature back in line. Conversely, if the kinetic energy is too low, ζ decreases, allowing the velocities to rise and the temperature to stabilize at the target value. The "inertia" of this process is controlled by the value of Q, which therefore plays the role of a fictitious mass. This deterministic feedback mechanism distinguishes the Nosé-Hoover thermostat from stochastic approaches. By continuously adjusting the velocities of all particles, this method preserves the natural evolution of the system's dynamics while still achieving temperature control. The benefit of using this extended Lagrangian framework is that it allows for smooth, continuous temperature regulation without disrupting important dynamical properties, such as diffusion coefficients or time-dependent correlations.

Note that the above equations of motion conserve the following quantity

$$U_{\mathrm{NH}} = \sum_{i=1}^{N} \frac{\vec{p}_i^2}{m_i} + V(\{r\}) + \frac{\zeta^2 Q}{2} + g k_B T \log s, \tag{6.48}$$

which can be used as a check when implementing the thermostat, or when testing the simulation parameters (*e.g.* the time step). Hoover [1986] demonstrated that the above equations of motion are unique, in the sense that are different equations of the same form (*i.e.* containing an additional friction-like parameter) cannot lead to a canonical distribution. However, the simulation samples from the canonical distribution *only* if there is a single non-zero constant of motion, the energy. If there are more conservation laws, which can happen, for instance, if there are no external forces and the total momentum of the system is different from zero, than the Nosé-Hoover method does not work any more.

Figure 6.10 shows the failure of this thermostat for a cases that is deceptively simple: that of a harmonic oscillator. It is obvious that the dynamics simulated in the microcanical ensemble or with the Nosé-Hoover thermostat becomes non-ergodic. Although in practice one can always set the initial velocity of the centre of mass of the system to zero to remain with a single non-trivial conserved

Figure 6.10: The trajectory of the harmonic oscillator for a single initial condition simulated (from left to right) without a thermostat, with the Andersen thermostat, with the Nosé-Hoover thermostat. Adapted from Frenkel and Smit [2023].

quantity and therefore avoid this issue, it would be better to have an algorithm that works well in the general case[8].

As demonstrated in Martyna et al. [1992], this issue can be overcome by coupling the Nosé-Hoover thermostat to another thermostat or, if necessary, to a whole chain of thermostats, which take into account additional conservation laws. Here I provide the equations of motion for a system coupled to an M-link thermostat chain, where each link i is an additional degree of freedom associated to a "mass" Q_i:

$$\ddot{\vec{r}}_i = \frac{\vec{F}_i}{m_i} - \zeta \dot{\vec{r}}_i \tag{6.49}$$

$$\dot{\zeta}_k = \frac{p_{\zeta_k}}{Q_k} \tag{6.50}$$

$$\dot{p}_{\zeta_1} = \left(\sum_{i=1}^{N} \frac{\vec{p}_i^2}{m_i} - g k_B T \right) - \frac{p_{\zeta_2}}{Q_2} p_{\zeta_1} \tag{6.51}$$

$$\dot{p}_{\zeta_k} = \left[\frac{p_{\zeta_{k-1}}^2}{Q_{k-1}} - k_B T \right] - \frac{p_{\zeta_{k+1}}}{Q_{k+1}} p_{\zeta_k} \tag{6.52}$$

$$\dot{p}_{\zeta_M} = \left[\frac{p_{\zeta_{M-1}}^2}{Q_{M-1}} - k_B T \right]. \tag{6.53}$$

As discussed in Frenkel and Smit [2023] (where all these arguments are presented more in depth) of the M additional degrees of freedom only two (the first link and the thermostat "centre", $\zeta_c \equiv \sum_{k=2}^{M} \zeta_k$) are independently coupled to the dynamics, which is what is needed when there are two conservation laws.

Important
The chained Nosé-Hoover thermostat is the most common thermostat used in all-atom simulations. However, the default parameters change from package to package (at the time of writing $M = 3$ for LAMMPS and 10 for GROMACS, for instance). The choice of Q (or of $\{Q_i\}$ in the case of chains) is important and should be made with care. If Q is very large, the energy exchange between the system and the reservoir is very slow, as in the $Q \to \infty$ limit we recover the Hamiltonian dynamics. By contrast, in the small-Q limit the high coupling can be give raise to unphysical energy (and therefore temperature) fluctuations.

Bussi-Donadio-Parrinello

We know from the equipartition theorem, Eq. (6.2), that the instantaneous temperature $T(t)$ is related to the kinetic energy of the system, $K(t)$, through:

$$T(t) = \frac{2K(t)}{3Nk_B}. \tag{6.54}$$

Since on a computer we always have a discrete dynamics, in the following I will slightly abuse the notation by using $A(k)$ to mean $A(k\Delta t)$, where k is an index that keeps track of the time step and A is a time-dependent function.

[8] provided that the dynamics is of interest; otherwise, you can rely on the good old Andersen thermostat.

The simplest way of fixing the temperature is to scale the velocities to achieve the target temperature instantaneously: the new velocities $\vec{v}_i(k+1)$ are obtained from the old velocities $\vec{v}_i(k)$ by multiplying them by a scaling factor α, *e.g.* $\vec{v}_i(k+1) = \alpha\vec{v}_i(k)$, where α is given by

$$\alpha = \sqrt{\frac{T}{T(k)}} = \sqrt{\frac{K}{K(k)}} \tag{6.55}$$

where T and K are the target temperature and kinetic energy. This method has two drawbacks:

1. It does not yield the correct fluctuations for the kinetic energy.
2. The rescaling affects all particles at the same time in a abrupt way, greatly disturbing the dynamics, as was the case with the Andersen thermostat.

The second issue can be mitigated by weakly coupling the system to a heat bath with a characteristic relaxation time τ. With this method, called the Berendsen et al. [1984], the velocities are gradually adjusted to bring the temperature toward the target value. The velocity scaling factor in the Berendsen thermostat is given by:

$$\alpha = \sqrt{1 + \frac{\Delta t}{\tau}\left(\frac{K}{K(k)} - 1\right)}. \tag{6.56}$$

This factor ensures that the system's temperature approaches T over time, with the relaxation time τ controlling the speed of this adjustment. The Berendsen thermostat achieves smooth temperature control, but it still suppresses the natural energy fluctuations required for proper sampling of the canonical ensemble.

Bussi et al. [2007] introduced a thermostat that is based on the idea of velocity rescaling, but it samples the canonical ensemble. Note that in the continous limit, *i.e.* for $\Delta t \to 0$, Eq. (6.56) becomes

$$dK = (K - K(t))\frac{dt}{\tau}. \tag{6.57}$$

This equation makes it obvious that $K(t) \to K$ exponentially, in a deterministic way (*i.e.* without fluctuations). A way of introducing fluctuations is to add a Weiner noise dW, weighted by a factor that ensures that the fluctuations of the kinetic energy are canonical[9]:

$$dK = (K - K(t))\frac{dt}{\tau} + 2\sqrt{\frac{K(t)K}{N_f}}\frac{dW}{\sqrt{\tau}}, \tag{6.58}$$

where N_f is the number of degrees of freedom in the system. Since the total momentum is conserved, $N_f = 3N - 1$ in a 3D simulation. With some non-trivial math Bussi et al. [2007] demonstrated that Eq. (6.58) yields the following scaling factor:

$$\alpha^2 = e^{-\Delta t/\tau} + \frac{K}{N_f K(k)}\left(1 - e^{-\Delta t/\tau}\right)\left(R_1^2 + \sum_{i=2}^{N_f} R_i^2\right) \tag{6.59}$$

$$+ 2R_1 e^{-\Delta t/\tau}\sqrt{\frac{K}{N_f K(k)}\left(1 - e^{-\Delta t/\tau}\right)}, \tag{6.60}$$

where the $\{R_i\}$ are independent random numbers extracted from a Gaussian distribution with zero mean and unit variance.

The interpretation of (6.58) is that, in order to sample the canonical ensemble by rescaling the velocities, the target kinetic energy (*i.e.* the target temperature) should be chosen randomly from the associated canonical probability distribution function[10]:

[9] It turns out that there is some freedom in choosing this factor, but only if we don't want to keep the Berendsen part untouched.

[10] This is a chi-squared PDF, which appears when dealing with sums of the squares of independent Gaussian random variables. Since the kinetic energy is the sum of squared velocities, each of which is distributed normally, its PDF is a chi-square.

$$P(K(t)) \propto K(t)^{N_f/2-1} e^{-\beta K(t)}. \tag{6.61}$$

One way of doing this would be to randomly extract a value $\bar{K}$ from Eq. (6.61) each time a rescale should be performed, and use it as the target kinetic energy in Eq. (6.55). However, this would generate abrupt changes in the dynamics, as for the simple velocity rescaling and Andersen thermostats. By contrast, the stochastic process defined in Eq. (6.58) is smoother, since the rescaling procedure happens on a timescale given by τ, in the same vein as with the Berendsen thermostat.

As for the Nosé-Hoover thermostat, this thermostat, which is sometimes called the stochastic velocity rescaling thermostat, has an associated conserved quantity that can be used to check the choice of the time step and of the other simulation parameters:

$$U_{\mathrm{BDP}} = \sum_{i=1}^{N} \frac{\vec{p}_i^2}{m_i} + V(\{r\}) - \int_0^t (K - K(t')) \frac{dt'}{\tau} - 2 \int_0^t \sqrt{\frac{K(t')K}{N_f}} . \frac{dW(t')}{\sqrt{\tau}} \tag{6.62}$$

Other thermostats

There are many other thermostats, which I will not introduce here because of time constraints. Here I list a few, just for reference:

- Langevin thermostats for systems where the dynamics is overdamped. Examples are particles dispersed in a viscous fluid.
- Brownian thermostats for systems where inertia is negligible. Examples are concentrated suspensions of colloids or bacteria.
- Stochastic rotational dynamics to model hydrodynamic interaction. Examples are diluted solutions of colloids or bacteria where long-range hydrodynamic effects are important and of interest (*e.g.* sedimentation or flocking).

> **Important**
> I reiterate that in equilibrium simulations the static properties of a system are independent on the thermostat used (provided that the thermostat reproduces the correct temperature fluctuations). However, if the dynamics is of interest (which is always the case in out-of-equilibrium systems, but it is often the case also for equilibrium systems), then choosing the right thermostat is Ruiz-Franco et al. [2018].

6.2.2 Barostats

Experiments are usually performed at constant pressure (*e.g.* ambient pressure). Moreover, while statistical mechanics results ensure that ensembles are equivalent in equilibrium, there are many situations in which it is more convenient to simulate in a specific ensemble. For instance, equilibrating a crystal structure usually requires that the edge lengths of the box can relax and fluctuate, or the coexisting region between a gas and a liquid is extended in (T, ρ), but it is just a line in the (T, P) projection. In the same as controlling the temperature requires coupling the system to a thermostat, fixing P means coupling the system to a barostat. When simulating at constant pressure it is very common, but by no means the only possibility, to also fix the temperature, thereby simulating in the so-called isothermal-isobaric ensemble.

As for thermostats, there exist many barostats, each having its own unique advantages and disadvantages. Here I briefly present three barostats, two based on the extended-Lagrangian formalism and a stochastic one.

Nosè-Hoover barostat

The Nosè-Hoover formalism can be extended to fix the pressure. In this case the Equations of motion, as presented in Martyna et al. [1994], which improves the original scheme by Hoover [1986] that only yields approximated constant-P distributions, are:

$$\dot{\vec{r}}_i = \frac{\vec{p}_i}{m_i} + \frac{p_\epsilon}{W}\vec{r}_i \tag{6.63}$$

$$\dot{\vec{p}}_i = \vec{F}_i - \left(1 + \frac{d}{dN}\right)\frac{p_\epsilon}{W}\vec{p}_i - \frac{p_\zeta}{Q}\vec{p}_i \tag{6.64}$$

$$\dot{V} = \frac{dVp_\epsilon}{W} \tag{6.65}$$

$$\dot{p_\epsilon} = dV(P(t) - P) + \frac{1}{N}\sum_{i=1}^{N}\frac{\vec{p}_i^2}{m_i} - \frac{p_\zeta}{Q}\vec{p}_\epsilon, \tag{6.66}$$

where d is the space dimensionality, $\epsilon = \log(V/V(0)$, where $V(0)$ is the volume at $t = 0$, W is the inertia parameter associated to ϵ, p_ϵ is the conjugate momenum of ϵ (here I'm using the same notation as Frenkel and Smit [2023]), and P and $P(t)$ are the target and instantaneous pressures, respectively. In particular, the instantaneous pressure is

$$P(t) = \frac{1}{dV}\left[\sum_{i=1}^{N}\left(\frac{\vec{p}_i^2}{m_i} + \vec{r}_i \cdot \vec{F}_i\right) - dV\frac{\partial U(V)}{\partial V}\right], \tag{6.67}$$

which is the virial pressure of a system with a non-constant volume. Note that the second term in (6.67) is non-zero if the energy depends explicitly on volume, which is always the case for interaction potentials that are long-range, or for which long-range corrections due to cut-offs have to be evaluated.

Note that equations (**??**) also contain a coupling to a Nosé-Hoover thermostat, which can be generalised to a chain thermostat if $Q \to Q_1$, $p_\zeta \to p_{\zeta_1}$ the equations of motions of the other $M - 1$ links are added.

Parrinello-Rahman

The Parrinello-Rahman barostat, introduced in Parrinello and Rahman [1980], uses an extended Hamiltonian to account for both particle motions and cell shape fluctuations, enabling the simulation of anisotropic pressure conditions, which is very common, for instance, when dealing with crystalline structures. This Hamiltonian includes original Hamiltonian of the system, which accounts for the kinetic and potential energy of the particles, plus additional terms that describe the kinetic and potential contributions of the simulation box itself.

The additional terms can be written in terms of the matrix

$$\hat{H} = \begin{bmatrix}\vec{a} & \vec{b} & \vec{c}\end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix}, \tag{6.68}$$

where $\vec{a}$, $\vec{b}$ and $\vec{c}$ are the basis vectors of a generic (possibly anisotropic) simulation box, and coincide with its edges. The position of particle i in the box is just

$$\vec{r}_i = \hat{H}\vec{s}_i = x_i\vec{a} + y_i\vec{b} + z_i\vec{c}, \tag{6.69}$$

where $0 < x_i, y_i, z_i < 1$ are the scaled (fractional) coordinates of particle i. With this formalism, the distance between particles i and j is

$$r_{ij}^2 = \vec{s}_{ij}\hat{H}^T\hat{H}\vec{s}_{ij} = \vec{s}_{ij}\hat{G}\vec{s}_{ij}, \tag{6.70}$$

where $G \equiv \hat{H}^T\hat{H}$.

We can now write down the expression for the extended Hamiltonian:

$$H = \sum_{i=1}^{N}\frac{1}{2}m_i\dot{\vec{s}}_i^T\hat{G}\dot{\vec{s}}_i + V(\{\vec{r}_i\}) + \frac{1}{2}W\mathrm{Tr}(\dot{\hat{H}}^T \cdot \dot{\hat{H}}) + P(t)\det(\hat{H}), \tag{6.71}$$

where $\det(\hat{H}) = V$ is the box volume, $P(t)$ is the istantaneous pressure, and the parameter W is the strength of the barostat coupling, and can be interpreted as the mass of the fictitious piston exerting the external pressure on the system.

By deriving Eq. (6.71), the following equations of motion are obtained:

$$m_i \ddot{\vec{s}}_i = \hat{H}^{-1} \vec{f}_i - m_i \hat{G}^{-1} \dot{\hat{G}} \dot{\vec{s}}_i \tag{6.72}$$

$$W \ddot{\hat{H}} = (P(t) - P) V (\hat{H}^{-1})^T, \tag{6.73}$$

where $\vec{f}_i = -\vec{\nabla}_{\vec{r}_i} V(\{\vec{r}_j\})$ is the force acting on particle i.

Stochastic cell rescaling

> **Warning**
> TODO

6.3 Classical force fields

> **Tip**
> The main references for this part are Schlick [2010] and Leach [2001].

Now that we know the algorithms used to run MD codes, we shift our attention to the interaction potentials acting between the components that make up the system we want to simulate. We start from the so-called classical (or empirical) force fields, which describe the interactions between atoms using simplified mathematical models based on quantum mechanics calculations, empirical observations, and physical principles from classical mechanics. Unlike quantum methods (introduced in Molecular quantum mechanics), which rigorously account for the wave-like nature of particles and their interactions through principles like superposition and entanglement, classical force fields offer a pragmatic approach that balances computational feasibility with accuracy.

The potential energy functions used in classical force fields typically consist of terms that describe various types of interactions, including covalent bonds, van der Waals (dispersion) forces and electrostatic interactions. Despite their simplicity, classical force fields can exhibit remarkable predictive power and have been successfully applied across diverse scientific disciplines. Common force fields used in biomolecular simulations are AMBER and CHARMM.

In a classical force field (FF from now on), the main assumption is that of *additivity*, which makes it possible to write down the total energy of a system as a sum of several contributions. These are classified into terms that account for atoms that are linked by covalent bonds (*bonded* or *local* terms), and terms that account for noncovalent interactions (*non-bonded* or *non-local* terms), so that the total energy of the system is written as

$$E_{\text{tot}}(\{\vec{r}\}) = E_{\text{bonded}}(\{\vec{r}\}) + E_{\text{non-bonded}}(\{\vec{r}\}), \tag{6.74}$$

where I made explicit the fact that the energy (and its contributions) are functions of the set of positions of the atoms, $\{\vec{r}\}$.

> **Note**
> The separation of contributions into bonded and non-bonded terms is not only a conceptual one, but it has also practical implications. For instance, bonded interactions act on pairs or groups of atoms that do not change during the course of the simulation, and therefore the data structures that keep track of these interacting atoms do not need to be updated. Moreover, non-bonded interactions are usually much less steep than bonded ones, which makes it possible to implement multiple-timestep protocols to speed-up simulations (see *e.g.* Chapter 14 of Schlick [2010]).

The functional forms of the functions (implicitly) defined in Eq. (6.74) depend on the force field employed, and are parametrised so as to reproduce the experimental structure and dynamics of target molecular systems, but also leveraging *ab initio* calculations, which are often used to complement the experimental results.

Vibrational Mode	Frequency [cm^{-1}]
H–O stretch	3600–3700
H–N stretch	3400–3500
H–C stretch	2900–3000
H–Br stretch	2650
C≡C, C≡N stretch	2200
C=C, C=O stretch	1700–1800
C–N stretch	1250
C–C stretch	1000
C–S stretch	700
S–S stretch	500

Vibrational Mode	Frequency [cm^{-1}]
H–O–H, H–N–H bend	1600
H–C–H bend	1500
H–C–H scissor	1400
H–C–H rock	1250
H–C–H wag	1200
H–S–H bend	1200
O–C=O bend	600
C–C=O bend	500
S–S–C bend	300
C=C torsion	1000
C–O torsion	300–600
C–C torsion	300
C–S torsion	200

Figure 6.11: Characteristic (left) stretching and (right) bending and torsional vibrational frequencies extracted from experimental spectra. Taken from Schlick [2010].

The traditional sources of parameters are provided by vibrational spectra (see Figure 6.11 for examples), which have been routinely used to derive the force constants of many of the functional forms discussed below. The idea is that the frequencies extracted from experimental spectra can be compared with the characteristic frequencies of the normal modes as evaluated in simulations in order to find the values of the force constants that minimise the differences.

6.3.1 Bonded interactions

In the context of molecular systems, *bonded interactions* refer to the forces that arise between atoms due to the formation of chemical bonds. These interactions occur when atoms are directly connected to each other through covalent bonds, which involve the sharing of electrons.

> **Caution**
> Note that the term *bonded* is often used for other mechanism through which atoms or molecules pair beyond covalend bonding (*e.g.* hydrogen bonding). It should be clear from the context what its implied meaning is, but be cautious!

Bonded interactions typically include several contributions due to bond stretching, angle bending, dihedral and improper torsions, so that the total bonded energy can be written as

$$E_{\text{bonded}} = \sum_{b\in\text{bonds}} E_{\text{stretch}}(b) + \sum_{\theta\in\text{angles}} E_{\text{bend}}(\theta) + \sum_{\varphi\in\text{dihedrals}} E_{\text{tors}}(\varphi), \tag{6.75}$$

where the three sums run over all the covalent bonds, all bond angles[11], and proper and improper dihedral angles[12], respectively.

Note that in writing Eq. (6.75) we have made an additional assumption on the interactions, which is commonly (but not always) verified in many FFs. Namely, there are no *cross terms* that couple different internal coordinates (such as bond lengths and angles). We will take a cursory look at these terms Section 6.3.1.

[11] A bond angle is the angle between two bonds that include a common atom.
[12] Angles defined by groups of four atoms that are connected by covalent bonds.

Bond stretching

This interaction occurs when atoms connected by a covalent bond move closer together or farther apart, resulting in the stretching or compression of the bond, and can be considered as "excitation" terms that accounts for small deviations from reference values, which are usually taken from experimental measurements.

The simplest type of interaction that accounts for bond deformations is the harmonic potential, which is based on Hooke's law and takes a quadratic form:

$$E_{\text{harmonic}}(r) = \frac{1}{2}k(r - r_{\text{ref}}), \tag{6.76}$$

where k is the force constant (which can be estimated by the, possibly reduced, mass and frequency of a bond vibration through $k = m\omega^2$), and r_{ref} is the reference value. Eq. (6.76) works only for rather small deformations (≈ 0.1Å).

Going beyond small deformations requires more complicated functional forms. An example is the Morse potential:

$$E_{\text{Morse}}(r) = D[1 - \exp(-S_m(r - r_{\text{ref}})]^2, \tag{6.77}$$

where the constants S_m and D controls the width and the depth of the potential well. This potential goes to infinity for $r \to 0$, while it tends to D as $r \to \infty$, modelling the bond dissociation. Evaluating exponential functions is a rather slow business on a computer, which is why the Morse potential is often expanded in series, and only the first terms are retained. The expansion up to the fourth order reads

$$E_{\text{Morse}}(r) \approx DS_m^2(r - r_{\text{ref}})^2 - DS_m^3(r - r_{\text{ref}})^3 + \frac{7}{12}DS_m^4(r - r_{\text{ref}})^4. \tag{6.78}$$

I note in passing that the previous equation implies the force constant of Eq. (6.76) can be related to the Morse parameters by $k = DS_m^2$.

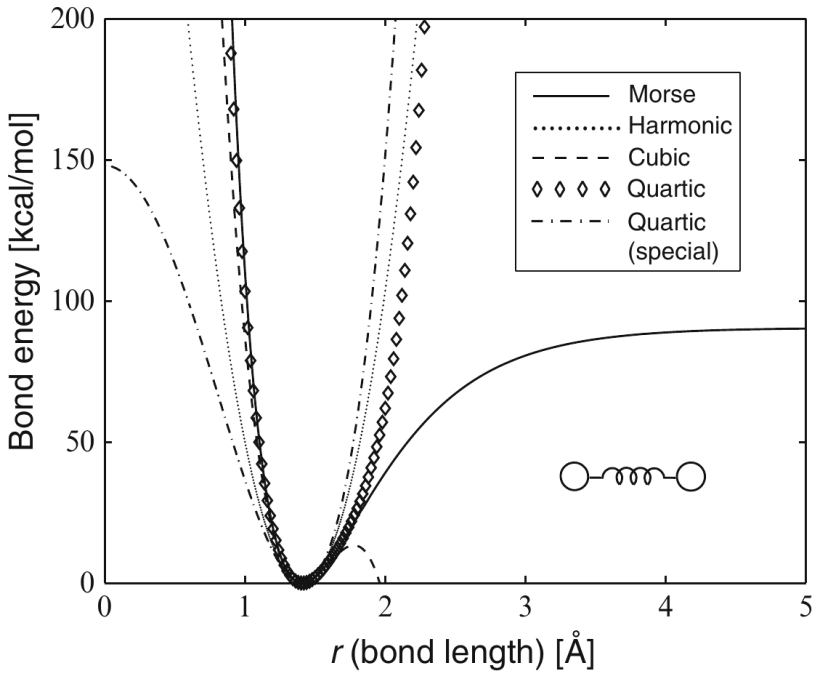

Figure 6.12: Morse, harmonic, cubic and two quartic bond potentials for H-Br. Taken from Schlick [2010].

A comparison between the Morse potential and its expansions at the second, third and fourth order is shown in Figure 6.12. The change of curvature of the cubic potential yields unphysical behaviour for large deformations, but even the other approximated forms cannot model bond dissociation. Finally,

some force fields use the following different (special) quartic form to avoid having to compute odd powers of r (which require the computation of a square root, which is an expensive function to compute):

$$E_{\text{quartic}} = S_q(r^2 - r_{\text{ref}}^2)^2, \tag{6.79}$$

where the constant can be linked to the Morse parameters by matching the two second-order expansions, yielding $S_q = DS_m^2/4r_{\text{ref}}^2$. This function is also shown in Figure 6.12.

Angle bending

When three atoms are connected by two consecutive covalent bonds, the angle between these bonds can change, causing the atoms to deviate from a linear configuration. Angle bending interactions describe the energy associated with such deformations and are often modeled using harmonic potentials, where the energy increases quadratically with the deviation of the angle from its equilibrium value. At first approximation, the reference (equilibrium) value is given by the type of orbital hybridisation due to the bonds (*e.g.* 180° for *sp*, 120° for $sp\hat{2}$ and 109.47° for $sp\hat{3}$). However, the orbitals are often deformed, and real values differ from ideal ones. For instance, in propane the $C-C-C$ and $H-C-H$ bond angles are $\approx 112.5°$, and $\approx 107.5°$, respectively.

The most common potentials used have a harmonic form involving either angles or cosines:

$$E^{\theta}_{\text{harmonic}}(\theta) = K_h(\theta - \theta_{\text{ref}})^2 \tag{6.80}$$

$$E^{\theta}_{\text{trig}}(\theta) = K_t(\cos\theta - \cos\theta_{\text{ref}})^2. \tag{6.81}$$

If the second equation is expanded in series around θ_{ref} and compared to the first equation, it is possible to obtain the following relation, which connects K_h and K_t so that the small-angle fluctuations of the two forms are similar:

$$K_t = K_h \sin^2\theta_{\text{ref}}. \tag{6.82}$$

Since it does not require the computation of inverse trigonometric functions, the trigonometric form is more efficient from the computational point of view. Figure 6.13 shows the two forms defined in Eq. (**??**), where the value K_t has been chosen according to Eq. (6.82).

Dihedral rotation

Dihedral or torsional interactions arise when four atoms are connected by three consecutive covalent bonds, forming a torsion angle or dihedral angle. Rotating one group of atoms around the axis defined by the central bond changes the conformation of the molecule, leading to different energy minima corresponding to different dihedral angles.

As noted Section 2.3, in proteins the most important dihedral (torsional) angles are those associated to rotations around the $N-C^{\alpha}$ and $C^{\alpha}-C$ bonds, ϕ and ψ, which feature free-energy barriers of the order of the thermal energy. However, rotations around the peptide bond (dihedral ω) and around sp^3-sp^3 bonds such as those found in aliphatic said chains (dihedral χ) can also play a role in dictating the flexibility of proteins.

Dihedral interactions are typically described using periodic potentials that capture the periodicity of the energy as a function of the dihedral angle. A generic torsional interaction associated to a dihedral angle φ takes the form

$$E_t(\varphi) = \sum_n \frac{V_n}{2}[1 + \cos(n\varphi - \varphi_0)], \tag{6.83}$$

where n is an integer that determines the periodicity of the barrier of height V_n, and φ_0 is a reference angle, which is often set to 0 or π.

The most common potentials of the form of (6.83)) are those comprising twofold and threefold terms, which are enough to reproduce common energy differences (*e.g.* cis/trans and trans/gauce). Two examples from the CHARMM force field are presented in Figure 6.14:

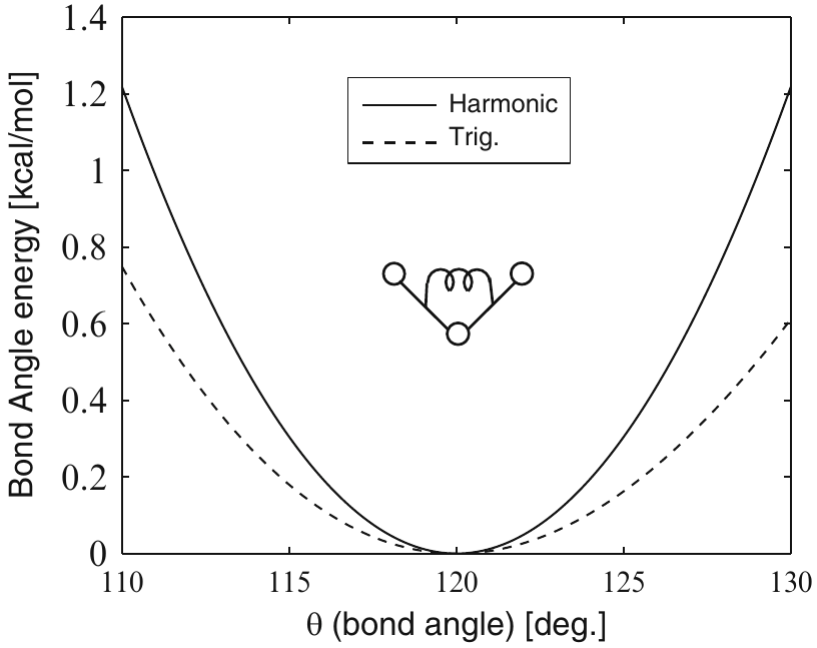

Figure 6.13: Harmonic bond-angle potentials of the forms given in Eq. (**??**) for an aromatic $C-C-C$ bond angle ($CA-CA-CA$ atomic sequence in CHARMM) with parameters $K_h = 40$ kcal/(mol rad^2) and $\theta_{\rm ref} = 2.1$ rad (120°). The K_t force constant is evaluated by using Eq. (6.82). Taken from Schlick [2010].

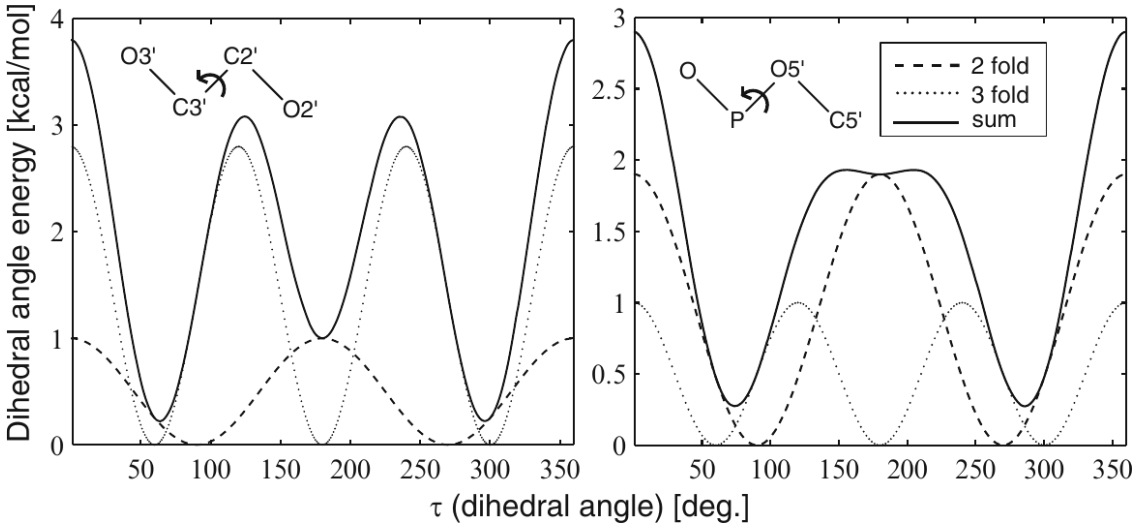

Figure 6.14: Twofold and threefold torsion-angle potentials and their sums for an $O-C-C-O$ rotational sequence in nucleic-acid riboses ($V_2 = 1.0$ and $V_3 = 2.8$ kcal/mol) and a rotation about the phosphodiester ($P-O$) bond in nucleic acids ($V_2 = 1.9$ and $V_3 = 1.0$ kcal/mol). Taken from Schlick [2010].

- The rotational interaction in the $O{-}C{-}C{-}O$ sequence in nucleic acids (*e.g.*, $O3'{-}C3'{-}C2'{-}O2'$ in ribose) shows a minimum at the trans state.
- The rotation about the phosphodiester bond $(P - O)$ in nucleic acids shows a very shallow minimum at the trans state.

For smaller molecules is often necessary to include higher-order terms ($n = 1, 2, 3, 4$ and 6) to reproduce the experimental torsional frequencies accurately.

Improper rotation

Improper torsion interactions occur when four atoms are bonded in a way that one atom is not directly in line with the other three. This configuration is often necessary to maintain the correct geometry or chirality of a molecule. Improper torsions are used to model the energy associated with deviations from the ideal geometry. The potential energy associated with improper torsions is typically described by a harmonic potential or a cosine function, depending on the force field parameters and the desired behavior of the molecule. A common form is

$$E_{\text{imp}}(\chi) = \frac{V_{\text{imp}}}{2}\chi^2, \tag{6.84}$$

where χ is the improper Wilson angle, which has the following definition: given four atoms i, j, k and l, where j is bonded to the other three, χ is the angle between bond $j - l$ and the plane defined by i, j and k, as shown in Figure 6.15.

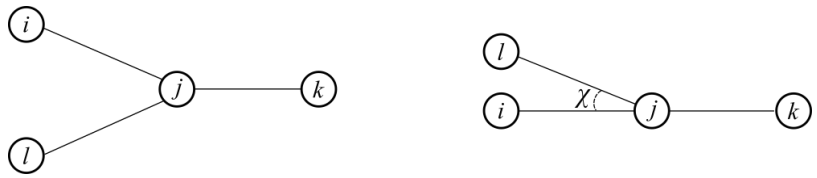

Figure 6.15: The definition of the Wilson angle χ, as seen (left) from the top and (right) from the side.

Cross terms

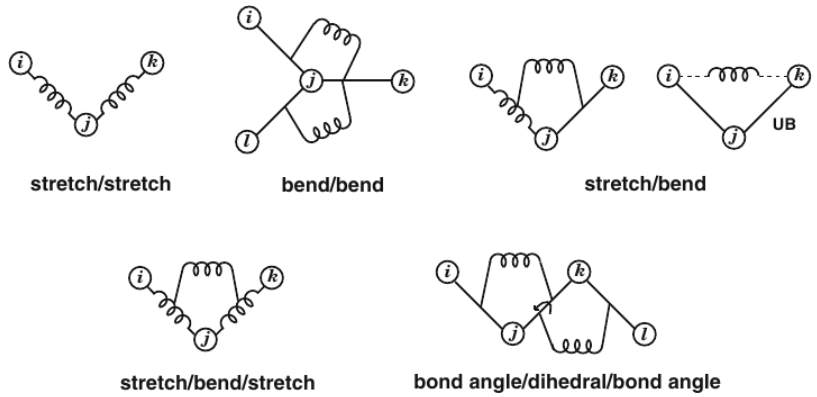

Figure 6.16: Schematic illustrations for various cross terms involving bond stretching, angle bending, and torsional rotations. Here UB stands for Urey-Bradley, which is a way of modelling the stretch/bend coupling by taking into account only the distance between the "far" atoms in the triplet. Taken from Schlick [2010].

It is sometimes necessary to include terms that couple different degrees of freedom in order to improve the accuracy, especially in force fields used to model small molecules. These *cross terms* are usually simple (quadratic) functions of the quantities in play (*e.g.* atom-atom distances or bond angles) and are fitted to the experimental vibrational spectra. Figure 6.16 pictorially shows some of these contributions.

6.3.2 Non-bonded interactions

Non-bonded interactions refer to the energy contributions that arise between atoms or molecules that are not directly connected by chemical bonds, and comprise several terms. Here I present the most common ones. Note that non-electrostatic non-bonded interactions are usually short-ranged and therefore are cut-off at some distance to improve performance, as discussed in Section 6.1.3.

Van der Waals Interactions

Van der Waals interactions are weak forces that arise due to fluctuations in the electron density of atoms or molecules. These interactions include both attractive forces, arising from dipole-dipole interactions and induced dipole-induced dipole interactions (van der Waals dispersion forces, see Israelachvili [2011] for a derivation of these terms), and repulsive forces, resulting from the overlap of electron clouds at close distances. Van der Waals interactions are described by empirical potential energy functions. The most common forms are the Lennard-Jones and Morse potentials, which we have already encountered when discussing Section 2.3.2 and Section 6.3.1, respectively. Here I report their functional forms, both of which account for attractive and repulsive components, for your convenience:

$$V_{\mathrm{LJ}}(r) = 4\epsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right) \tag{6.85}$$

$$V_{\mathrm{Morse}}(r) = \epsilon [1 - \exp(-a(r - r_0)]^2, \tag{6.86}$$

where ϵ sets the depth of the attractive well, σ is the LJ diameter, r_0 is the position of the minimum of the Morse potential, and a is linked to the curvature close to such a minimum.

Electrostatic Interactions

Electrostatic interactions arise from the attraction or repulsion between charged particles, such as ions or polar molecules. These interactions are governed by Coulomb's law, which states that the force between two charged particles is proportional to the product of their charges and inversely proportional to the square of the distance between them. In molecular systems, electrostatic interactions include interactions between charged atoms or molecules (ion-ion interactions), as well as interactions between charged and neutral atoms or molecules (ion-dipole interactions or dipole-dipole interactions). Since electrostatic interactions are long-ranged, they have to be handled with care, as discussed in Section 6.1.7.

Hydrogen Bonding

While sometimes hydrogen bonding interactions are modeled using empirical *ad-hoc* potentials that account for the directionality and strength of hydrogen bonds, in modern force fields they arise spontaneously as a result of the combination of the Van der Waals and electrostatic interactions acting between the atoms.

6.3.3 More on force fields

Modern force fields contain tens, hundreds or even more parameters, depending on what they have been designed to model. The value of every single parameter has been optimised to reproduce some target quantity, either generated with higher-fidelity numerical methods, or measured experimentally, or a mixture of both. However, no force field is perfect, as each FF has its own advantaged and disadvantages.

The force field should be chosen carefully, based on the problem at hand. For proteins and nucleic acids, common FFs are AMBER and CHARMM, but there are other possibilities. Note that it is often necessary to simulate molecules or functional groups that are not supported by the specific FF we wish to use. In this case, it is possible to use multiple force fields, provided the two FFs are compatible,

i.e. someone has parametrised the "cross interactions" between the atoms or molecules modelled with distinct force fields. As a golden rule, you should **never** mix parameters from different force fields.

In general, you should always follow the advice and good-practices given in the documentation of the FF(s) you are using, unless you **really** know what you are doing. For instance, the CHARMM force field recommends to use the TIP3P model for water rather than more sophisticated and realistic ones, since it has been parametrised using that particular model. Using another model can, in principle, be made to work, but would also make your results somehow questionable, which is **not** what you want from a scientific point of view.

6.3.4 GROMACS

GROMACS, an acronym for Groningen Machine for Chemical Simulations, is an open-source software package designed primarily for molecular dynamics simulations of biomolecular systems, providing insights into the structural dynamics of proteins, lipids, nucleic acids, and other complex molecular assemblies. GROMACS can run simulations efficiently on a wide range of hardware platforms, from single processors to large parallel computing clusters, and implements advanced algorithms and techniques, such as domain decomposition, particle-mesh Ewald summation for long-range electrostatics, and multiple time-stepping schemes.

Interestingly (and differently from other simulation engines) GROMACS also offers a comprehensive suite of analysis tools, making it possible to perform tasks such as trajectory analysis, free energy calculations, and visualization of simulation results.

Warning
The text that follows is just a collection of notes. It will be updated in the future.

- Use `mdp` files to create a `tpr` (which is a binary, non-human-readable file): at this step we need a topology file (`topol.top`)[13]
- Index files (`.ndx`) are used to assign atoms to categories that can be used to easily find them. Use `gmx make_ndx -f file.tpr -o output` to make a new index file.
- The `-deffnm` flag tells Gromacs to use the input file as a template for the filenames of output quantities
- The default trajectory file (`.trr`) contains all the info (coordinates, velocities, *etc.*). By contrast, `xtc` files are lighter since they store only the atom coordinates.

Berendsen thermostat
On newer versions, Gromacs will spit out a warning if you use the Berendsen thermostat (`tcoupl = berendsen`). However, by default `gmx grompp` considers a single warning as a fatal error. Use the `-maxwarn` flag to raise the number of acceptable warnings (*e.g.* `-maxwarn`).

[13] in Gromacs lingo, a topology file contains the details of the force field

Chapter 7

Coarse-grained models

7.1 Accuracy

> **Tip**
> This part has been adapted (taken almost verbatim in some parts) from Jin et al. [2022]. However, in that review accuracy is discussed only in the context of bottom-up CG models. Here I have tried to provide a more inclusive discussion.

How accurate a particular CG model is? Answering this question is hard not because it requires costly calculations, but because there are no metrics that makes it possible to provide an unambiguous answer. What do we mean with "accurate"? Following Jin et al. [2022], I define three separate yet related measures of CG model fidelity: consistency, representability, and transferability. The use and importance of each of these measures are dependent upon the scientific question of interest. It is therefore worthwhile to discuss each of these metrics with the understanding that all three contribute to the overall accuracy of a CG model. Although the original review discussing this topic focusses on bottom-up coarse-graining, I think that some of these metrics can, at least to some extent, extended to top-down models. Indeed, while it is true that top-down models are, by definition, not consistent with statistical mechanics, in the sense that they do not provide a direct connection between the FG and CG worlds but are instead primarily models in the larger sense of the word, they can nevertheless reproduce, sometimes quantitatively, some "macroscopic" properties of the coarse-grained system (*e.g.* thermodynamics).

7.2 Bottom-up

Bottom-up coarse-graining is a robust strategy in molecular modeling that allows detailed atomistic information to be condensed into simplified representations. This approach systematically derives effective potentials and interaction parameters from molecular-level data, reducing complex systems to manageable models while retaining critical physical characteristics. By representing groups of atoms or molecules as single units, bottom-up coarse-graining enables the study of larger systems or longer timescales than would be feasible with fully atomistic simulations, making it especially valuable for exploring mesoscale phenomena.

In practice, bottom-up coarse-graining methods often use information from all-atom simulations to inform the development of coarse-grained potentials. One widely used approach is the iterative Boltzmann inversion, which uses distributions from high-resolution simulations to create effective potentials for coarse-grained models. Other techniques, like the force-matching or multiscale coarse-graining methods, involve more direct optimization to match forces and structural correlations between atomistic and coarse-grained representations. The goal is to achieve a model that can replicate the thermodynamic and structural properties of the original system accurately.

Bottom-up coarse-graining procedures often retain the molecular specificity and accuracy needed for a range of applications, from materials science to biomolecular simulations. However, this accuracy

comes at a computational cost, as it requires initial high-resolution data and complex optimization procedures. Nonetheless, by sacrificing only unnecessary degrees of freedom, bottom-up coarse-graining enables researchers to strike a balance between simulation speed and model fidelity, making it an invaluable approach for studying mesoscale phenomena while staying rooted in molecular detail.

A key challenge in bottom-up coarse-graining is that while the coarse-grained model allows for efficient simulation of broader-scale dynamics, it inherently sacrifices fine-grained structural detail. This is where backmapping becomes essential. Backmapping is the process of reconstructing detailed atomic configurations from a coarse-grained representation, effectively transforming a simplified model back into a high-resolution form. This procedure is crucial for bridging scales, as it enables researchers to interpret coarse-grained simulations in terms of atomistic detail when necessary, such as when making predictions about properties that depend on atomic-scale structures.

To achieve accurate backmapping, algorithms often combine statistical inference with additional energetic minimization to produce atomistic configurations that are consistent with the original coarse-grained trajectory. By leveraging data from the coarse-grained simulation as initial constraints, these methods reconstruct missing details in a way that reflects the system's underlying physical behavior. Backmapping is essential for applications where final atomic-scale configurations are needed, such as in the prediction of molecular interactions or the preparation of input structures for further atomistic simulations. Through bottom-up coarse-graining and efficient backmapping, researchers can navigate seamlessly between scales, balancing computational efficiency with molecular accuracy.

7.3 Top-down

7.3.1 oxDNA/oxRNA

7.3.2 Vertex models

7.3.3 A hybrid approach: the Martini force field

Chapter 8

Enhanced sampling

> **Note**
> In this chapter I will use *reaction* as a general term for a microscopic process that transforms the system of interest (be it a macromolecule, an extended many-body system, a collection of atoms, *etc.*) between two well-defined states, A and B.

In biophysics, *reactions* (in the sense given above) are extremely important, and are often the focus of any enhanced sampling technique discussed in the context of biomolecules. However, these same techniques have been very often introduced in, or can be easily adapted to, other contexts[1].

8.1 Collective variables and reaction coordinates

The phase space of a many-body system is a highly-dimensional manifold where each point corresponds to a microstate. As the system evolves in time, the collection of points in phase space it visits form a trajectory that fully describes its evolution. In many cases it is useful to project the trajectory on a space with a lower dimensionality: the resulting quantities can be used to obtain *coarse-grained* models, but also to describe the essential features of a system in a way that is easier to analyse and visualise.

In the context of molecular simulations, any quantity that is used to describe a system in a coarse-grained fashion is a *collective variable* (CV), and can be written as a function of (all or a subset of) the microscopic degrees of freedom $\{\vec{r}\}$. In condensed-matter physics, collective variables are often called *order parameters*. By contrast, in the biophysics and biochemistry worlds the most used term is *reaction coordinate* (RC), defined as a parameter or set of parameters that describe the progress of the reaction of interest, serving as a way to quantify and track the changes occurring within a system as it evolves through the reaction pathway. Since we will be focussing on reactions, in the following I will tend to refer to the coarse-grained variable(s) of interest as RC or reaction coordinate(s), but most of what I will say translates naturally to any other CV.

The primary purpose of a reaction coordinate is to provide a simplified description of the system's thermodynamics, making it possible to monitor and analyze the progress of a reaction in terms of a single or a few variables: by using a reaction coordinate we are reducing the complexity of a many-body system with many degrees of freedom to obtain a simplified description that can be used to investigate the reaction itself, effectively applying a *dimensionality reduction* procedure. This simplification is essential for understanding the microscopic underpinnings of the reaction of interest. Defining a reaction coordinate makes it possible to draw a diagram such as the one shown in Figure 8.1, which are often called free-energy profiles or landscapes, where the variation of the free energy along a particular reaction coordinate or collective variable is plotted.

The choice of the RC depends on the specific process being studied and it is not, in general, unique. It can be a simple geometric parameter such as bond length, bond angle, or dihedral angle, or

[1] To provide an example that you can already appreciate, the bottom-up effective interactions discussed in the coarse graining chapter can be computed with any of the techniques discussed in this chapter.

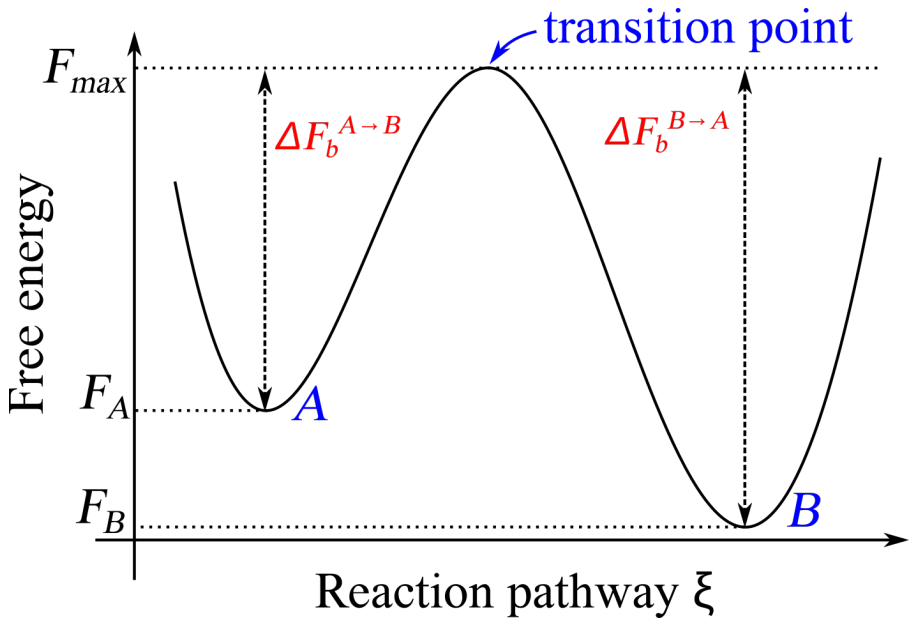

Figure 8.1: An example of the free energy landscape of a system displaying two basins separated by a transition state.

a more complex collective variable that captures the overall structural changes in the system, such as the distance between two key functional groups, the position or coordination number of a particular atom, or the solvent-accessible surface area of a biomolecule.

> **Note**
> In general, a reaction coordinate can be multidimensional, and sometimes it is useful, or even necessary, to define such complex variables. However, for the sake of simplicity here I will use unidimensional reaction coordinates, which will be designated by the symbol $\xi = \xi(\{\vec{r}\})$.

Once we have chosen a RC to characterise the reaction of interest, which is in general not an easy task, and an active area of research in itself, we can use it to describe the reaction. We can formally see how if we calculate the *partially-integrated partition function* (see *e.g.* Hénin et al. [2022]) by integrating the Boltzmann factor over all the degrees of freedom at constant ξ, *viz.*:

$$Q(\xi) = \int_V e^{-\beta H(\{\vec{r}\})} \delta(\xi - \xi(\{\vec{r}\})) d\{\vec{r}\}, \tag{8.1}$$

where $\delta(\cdot)$ is the Dirac delta distribution function. Note that the same procedure has been already carried out in the context of coarse graining, where the partial integration on the phase space made it possible to obtain a simplified description of the system, showing that the process of dimensionality reduction we apply is strictly the same in the two cases.

In turn, $Q(\xi)$ can be used to obtain a free-energy profile (sometimes called free-energy surface is ξ is multidimensional) such as the one presented in Figure 8.1, which is defined as

$$F(\xi) = -k_B T \ln(Q(\xi)). \tag{8.2}$$

Now consider an observable that can be written as a function of ξ. Its ensemble average can be formally written as

$$\langle O(\xi(\{\vec{r}\})\rangle = \frac{\int_V e^{-\beta H(\{\vec{r}\})} O(\xi(\{\vec{r}\})) d\{\vec{r}\}}{\int_V e^{-\beta H(\{\vec{r}\})} d\{\vec{r}\}}, \tag{8.3}$$

which can be simplified by using Eq. (8.1) to

$$\langle O(\xi)\rangle = \frac{\int_{\xi_{\min}}^{\xi_{\max}} O(\xi)Q(\xi)d\xi}{\int_{\xi_{\min}}^{\xi_{\max}} Q(\xi)d\xi} = \int_{\xi_{\min}}^{\xi_{\max}} O(\xi)P(\xi)d\xi, \tag{8.4}$$

where $\xi_{\min}$ and $\xi_{\max}$ correspond to the minimum and maximum values of ξ, and we have defined the *marginal probability density*

$$P(\xi) = \frac{Q(\xi)}{\int_{\xi_{\min}}^{\xi_{\max}} Q(\xi)d\xi} = \frac{Q(\xi)}{Q}, \tag{8.5}$$

where we have used the partition function $Q = \int_{\xi_{\min}}^{\xi_{\max}} Q(\xi)d\xi$.

A simple example
Consider a system formed by two DNA strands made of N_1 and N_2 units (atoms or coarse-grained beads), respectively, that can hybridise: a possible (although not exactly ideal) reaction coordinate could be the distance between the centres of mass of the two strands, *viz.*:

$$\xi = \left|\vec{R}_1^{\mathrm{cm}} - \vec{R}_2^{\mathrm{cm}}\right| = \left|\frac{1}{N_1}\sum_{i\in N_1}\vec{r}_i - \frac{1}{N_2}\sum_{i\in N_2}\vec{r}_i\right|. \tag{8.6}$$

8.2 Reactions and rare events

Consider a system that can switch, possibly reversibly, between two macrostates, A and B. Here the term macrostate is used loosely to indicate ensembles of microstates where the system resides for times that are much larger than the microscopic characteristic time; in thermodynamic parlance, A and B, which are sometimes called *basins*, should be either metastable or equilibrium states, and therefore separated by a free-energy barrier ΔF_b larger than the thermal energy.

Some examples
Examples relevant to computational biophysics are processes involving protein folding and unfolding, nucleic acid hybridisation, or switching between different conformations of the same (macro)molecule.

In this context the free-energy barrier[2] between A from B, $\Delta F_b^{A\to B} = F_{\max} - F_A$, is defined as the difference between the free energy of A, F_A and that of the transition state, $F_{\max}$, which is the highest free-energy point along the reaction pathway connecting A to B[3]. Note that $\Delta F_b^{A\to B}$ controls not only the probability of jumping from A to B, but also the the rate of the reaction, which is proportional to $e^{-\beta\Delta f_b}$ (see *e.g.* Eyring [1935] and Evans and Polanyi [1935]). See Figure 8.1 for a graphical definition of these quantities.

It is often the case that what interests us is the *reaction* itself rather than the A and B states, which are often known (and possibly have been characterised) beforehand. In this case, simulations starting from one of the two states, say A, would remain in A for sometime, then quickly jump to state B, where it would again reside for some time before switching basin once again, and so on. Moreover, if the free-energy barrier between the two basins is large ($\delta F_b 10 k_B T$), the number of transitions from A to B and back will be very small. Therefore, using unbiased simulations[4] to sample the transition itself, for instance to evaluate the free-energy landscape as in Figure 8.1, requires a large computational effort which is mostly wasted in sampling uninteresting parts of the phase space.

[2] Sometimes also called *activation (free) energy.*
[3] Note that, per this definition, $\Delta F_b^{A\to B} \neq \Delta F_b^{B\to A} = F_{\max} - F_B$.
[4] In unbiased simulations the system evolves according to the fundamental laws of physics (Newton's laws of motion in this context), without any artificial bias or constraints imposed.

In this part we will understand how the sampling of the transition from A to B can be enhanced by using advanced computational techniques (collectively known as rare event sampling techniques).

8.3 Umbrella Sampling

The first technique I will present is the venerable *umbrella sampling* (US), which was introduced in the Seventies by Torrie and Valleau [1977].

The basic idea behind umbrella sampling is to bias the system along a chosen reaction coordinate by adding a so-called *biasing potential*[5] that confines the system to different regions (also known as *windows*) along that coordinate. By running multiple simulations with biasing potentials centred on different points along the reaction coordinate, the entire range of interest can be sampled. After sufficient sampling is done in each window, the bias introduced by the additional potentials can be removed to obtain the unbiased free energy profile (or any other observable of interest) along the reaction coordinate.

A typical umbrella sampling simulation thus comprises several steps, which I will discuss separately.

8.3.1 1. Choosing the reaction coordinate

This is arguably the most important step, since choosing a sub-optimal RC can sometimes massively increase the required simulation time. Fortunately, most of the times the choice is either obvious (*e.g.* the concentration of the product in a chemical reaction), or dictated by the observable(s) of interest (see below for an Section 8.3.6).

8.3.2 2. Selecting a biasing potential

The role of the biasing potential $V^{\text{bias}}(\xi)$ is to confine a system within a (usually rather narrow) region of the reaction coordinate. As such it must be a function of the reaction coordinate(s) only, without any explicit dependence on any of the microscopic $\{\vec{r}\}$. The most common choice is a harmonic potential, whose shape gives the method its name and usually takes the form

$$V^{\text{bias}}(\xi) = \frac{1}{2}K(\xi - \bar{\xi})^2, \tag{8.7}$$

where $\bar{\xi}$ is the position of the minimum of the potential and K is the strength of the resulting spring force. Other choices are possible (see *e.g.* Rovigatti et al. [2015] or Virnau and Müller [2004] for examples of biases that are not differentiable and therefore can only be used in Monte Carlo simulations).

8.3.3 3. Partitioning the reaction coordinate into windows

Next, we need to split the range of interest, $[\xi_{\min}, \xi_{\max}]$, into windows. The most common strategy is to divide the reaction coordinate into equispaced windows centred on $\xi_1, \xi_2, \xi_3, \ldots$, with $i \in [1, N]$, where N is the total number of windows (and hence of independent simulations). The distance between two neighbouring windows, $\Delta\xi_i = \xi_{i+1} - \xi_i$, which is often taken as a constant, should be chosen carefully: on one hand it should be as large as possible to make N as small as possible; on the other hand, $\Delta\xi$ should be chosen so that there is some overlap between adjacent windows to prevent discontinuities in the free energy profile. This is to ensure that the neighboring windows provide sufficient sampling for accurate reweighting. An often good-enough first estimate can be made by assuming that $P(R_i) \approx P(R_{i+1})$, and then by choosing a $\Delta\xi$-value for which $V^{\text{bias}}(\Delta\xi_2)$ is of the order ot k_BT, *i.e.* that the value of the biasing potential calculated in the midpoint separating two neighbouring windows is of the order of the thermal energy.

In practice, the number, size and spacing of the windows depends on the curvature of the free energy profile along the reaction coordinate, which is not known beforehand. Smaller windows may

[5] The bias often takes the form of a harmonic potential whose shape, resembling an umbrella, gives the method its name.

be needed in regions with steep gradients or large energy barriers, while larger windows may suffice in more gradually changing regions. Fortunately, given the independent nature of the simulations that run in each window, the partitioning can be improved upon *a posteriori*: if one realises that the explored range is not sufficient, it can be extended by adding simulations with biasing potentials centred beyond $\xi_{\min}$ and/or $\xi_{\max}$. Sampling can also be improved by adding simulations in regions of the RC where the $P(R)$ is steeper.

At the end of this procedure, each window will be assigned a biasing potential $V_i^{\text{bias}}(\xi)$.

Adaptive sampling
There are more advanced methods, where the size and placement of windows are adjusted dynamically based on the evolving free energy landscape observed during the simulation (see *e.g.* Bartels and Karplus [1997] or Bartels and Karplus [1998]). This can help to focus computational resources on regions of interest and improve sampling efficiency.

8.3.4 4. Sampling

Molecular dynamics or Monte Carlo simulations are performed within each window, allowing the system to equilibrate and sample configurations consistent with the biasing potential. In general ensuring that a given window has converged is not necessarily straightforward, but can be done by techniques such as block averaging.

8.3.5 5. Reweighting and combining the data

In the final step we gather the data from each window and combine it together to calculate the unbiased quantities of interest. I will first show how to unbias the data from each window, and then how to join all the results together.

In analogy with Eq. (8.5) we can defined a *biased* marginal probability density for the i-th windows, $P_i^b(\xi)$, as

$$P_i^b(\xi) = \frac{Q_i^b(\xi)}{\int_{\xi_{\min}}^{\xi_{\max}} Q_i^b(\xi) d\xi} = \frac{\int_V e^{-\beta(H(\{\vec{r}\})+V_i^{\text{bias}}(\xi))}\delta(\xi - \xi(\{\vec{r}\}))d\{\vec{r}\}}{\int_V e^{-\beta(H(\{\vec{r}\})+V_i^{\text{bias}}(\xi(\{\vec{r}\})))}d\{\vec{r}\}}, \tag{8.8}$$

where the biased partially-integrated partition function $Q_i^b(\xi)$ has also been defined. We note that the biasing factor $e^{-\beta V_i^{\text{bias}}(\xi)}$ depends only on ξ and therefore, since integration is performed on all degrees of freedom but ξ, can be moved outside of the integral. If we do so and then multiply and divide by Q we obtain

$$\begin{aligned} P_i^b(\xi) &= e^{-\beta V_i^{\text{bias}}(\xi)} \frac{\int_V e^{-\beta H(\{\vec{r}\})}\delta(\xi - \xi(\{\vec{r}\}))d\{\vec{r}\}}{\int_V e^{-\beta(H(\{\vec{r}\})+V_i^{\text{bias}}(\xi(\{\vec{r}\})))}d\{\vec{r}\}} \frac{Q_i}{Q_i} = \\ &= e^{-\beta V_i^{\text{bias}}(\xi)} \frac{\int_V e^{-\beta H(\{\vec{r}\})}\delta(\xi - \xi(\{\vec{r}\}))d\{\vec{r}\}}{Q} \frac{Q}{\int_V e^{-\beta(H(\{\vec{r}\})+V_i^{\text{bias}}(\xi(\{\vec{r}\})))}d\{\vec{r}\}} = \\ &= e^{-\beta V_i^{\text{bias}}(\xi)} P_i(\xi) \left\langle \frac{1}{e^{-\beta V_i^{\text{bias}}(\xi)}} \right\rangle, \end{aligned} \tag{8.9}$$

where $\langle\cdot\rangle$ represents an *unbiased* ensemble average and $P_i(\xi)$ is the marginal probability density of the i-th window. Note that, being an ensemble average, $\left\langle \frac{1}{e^{-\beta V_i^{\text{bias}}(\xi)}} \right\rangle$ does not depend on ξ, and therefore it is a (in general unknown) constant[6]. As a consequence, we can obtain the unbiased marginal probability density up to a multiplicative constant:

$$\mathcal{P}_i(\xi) = P_i^b(\xi) e^{\beta V_i^{\text{bias}}(\xi)} \propto P_i(\xi). \tag{8.10}$$

[6]This constant will take different values in different windows, since it is an ensemble average of a window-dependent quantity, $V_i^{\text{bias}}(\xi)$.

where I use the symbol $\mathcal{P}_i(\xi)$ in place of $P_i(\xi)$, since the former is unnormalised and therefore not a proper probability density. This procedure is known as *unbiasing*, and it is a special case of *histogram reweighting*[7]. Applying Eq. (8.10) yields N functions $\mathcal{P}_i(\xi)$ that are shifted relative to each other because of the unknown constant. The *total* $\mathcal{P}(\xi)$ can be recoverd by stitching together all the $\mathcal{P}_i(\xi)$, utilising the regions of the ξ-space where each pair of windows overlap significantly to find the unknown multiplying constants. There are several methods available to perform this task. Here I will present two such methods: a simple least-squares fit and the (much more powerful) WHAM.

On the discrete nature of $P_i(\xi)$
The derivation above has been carried out by considering continuous functions for the sake of clarify. However, the simulation output is always a *histogram*, *i.e.* $P^b_{i,\zeta} = P^b_i(\xi_\zeta)$ (and, equivalently, $\mathcal{P}_{i,\zeta}$), where the greek subscript ζ runs over the histogram bins and corresponds to a value of the RC, ξ_ζ. In the following derivations I will use this latter notation.

Least-squares method

Consider two windows i and j (with $|j - i| = 1$), whose unnormalised marginal probability densities overlap in a ξ-region $\{\xi_o\}$. We want to find the constant C_{ij} that, multiplying $\mathcal{P}_{j,\zeta}$, minimises the mean-squared error between the two overlapping portions of the histograms, which is defined as

$$\mathrm{MSE}_{ij} = \sum_{\zeta \in \xi_o} \left(\mathcal{P}_{i,\zeta} - C_{ij}\mathcal{P}_{i,\zeta}\right)^2 . \tag{8.11}$$

Imposing $\frac{d\mathrm{MSE}_{ij}}{dc_{ij}} = 0$ we find

$$C_{ij} = \frac{\sum_{\zeta\in\xi_o} \mathcal{P}_{i,\zeta}\mathcal{P}_{i,\zeta}}{\sum_{\zeta\in\xi_o} \mathcal{P}^2_{j,\zeta}}. \tag{8.12}$$

Exercise
From a numerical point of view, it is often better to stitch the free-energy profiles $F_{i,\zeta} = -k_BT \ln \mathcal{P}_{i,\zeta}$ rather than the bare $\mathcal{P}_{i,\zeta}$, since in the former case the unknown constant is additive rather than multiplicative. Try to derive an expression for such an additive constant A_{ij}.

In practice, with this method the 0-th window data are unchanged, while all the subsequent ones are rescaled one after the other by repeatedly applying Eq. (8.12). Once the final histogram $\mathcal{P}_\zeta$ is obtained, we can either normalise it (if we need a proper probability density), or use it to compute the associated free-energy profile

$$F_\zeta = -k_BT \ln \mathcal{P}_\zeta + \mathrm{const}, \tag{8.13}$$

where I made explicit the fact that classical free energies are always specified up to an additive constant, which can be chosen freely. If we are interested in a reaction between states A and B, it is common to set $F(\xi_A)$ or $F(\xi_B)$ to 0, while for potentials of mean force (see for instance the Section 8.3.6 below) it is customary to set $\lim_{\xi\to\infty} F(\xi) = 0$.

[7]Under certain conditions, histograms computed with some parameters (*e.g.* temperature or chemical potential) can be reweighted to obtain the same quantity for some other (usually nearby) values of the same parameters, without having to run additional simulations.

The Weighted Histogram Analysis Method (WHAM)

Kumar et al. [1992] is a widely used reweighting technique for combining data from multiple biased simulations to obtain an unbiased estimate of the free energy profile. The basic idea behind WHAM is to reweight the probability distributions obtained from each window simulation such that they are consistent with each other and with the unbiased distribution. The reweighting process involves applying a set of equations that account for the biasing potentials applied in each window and the overlap between adjacent windows.

The following derivation has been inspired by Guillaume Bouvier's "traditional" derivation, which in turn is based on the Kumar et al. [1992].

We start by defining some accessory quantities that make it possible to generalise the method beyond a specific Umbrella Sampling to any case where one wants to join multiple overlapping histograms:

- Latin letters i, j and k will be used to index different *simulations* rather than windows, since one may want to run multiple simulations for the same window to improve statistics. The total number of simulations is $\#_{\text{sims}}$.
- The reaction coordinate is split into $\#_{\text{bins}}$ bins, and ζ is the ζ-th bin, *i.e.* ξ_ζ.
- $u_{i,\zeta}$ is the biasing factor applied to the ζ-th bin of the i-th simulation. For the type of Umbrella Sampling described above, $u_{i,\zeta} = e^{-\beta V_i^{\text{bias}}(\xi_\zeta)}$.
- The output of simulation i is the best estimate for the *biased* probability $q^b_{i,\zeta}$, which we define as $P^b_{i,\zeta} \equiv \frac{m_{i,\zeta}}{M_i}$, where $m_{i,\zeta}$ is the number of counts in bin ζ and M_i is the number of generated samples, with both quantities referring to the i-th simulation.
- p°_ζ is the (unbiased) probability of bin ζ, that is, the quantity we wish to calculate.
- The biased probability of bin ζ of simulation i is then:

$$\mathrm{q}^b_{i,\zeta} = u_{i,\zeta} p^\circ_\zeta f_i \quad (8.14)$$

where f_i is a normalising constant that ensures that $\sum_\zeta q^b_{i,\zeta} = 1$, *viz.*

$$f_i^{-1} = \sum_\zeta u_{i,\zeta} p^\circ_\zeta. \quad (8.15)$$

Using the above definitions, the best estimate for the *unbiased* probability of the ζ-th bin of simulation i is

$$P_{i,\zeta} = \frac{P^b_{i,\zeta}}{u_{i,\zeta} f_i} = \frac{m_{i,\zeta}}{M_i u_{i,\zeta} f_i}. \quad (8.16)$$

We assume that p°_ζ can be written as a weighted sum of all the reweighted histograms, $P_{i,\zeta}$, as follows:

$$p^\circ_\zeta = \sum_i \omega_{i,\zeta} P_{i,\zeta}, \quad (8.17)$$

where the set of weights $\{\omega_{i,\zeta}\}$ are normalised, *e.g.* $\sum_i \omega_{i,\zeta} = 1$. The WHAM method boils down to ensuring that the weights are chosen so as to minimise the expected variance of p°_ζ, which is defined as

$$\operatorname{Var} p^\circ_\zeta = \left\langle \left(p^\circ_\zeta - \langle p^\circ_\zeta \rangle \right)^2 \right\rangle = \left\langle \left(\sum_i \omega_{i,\zeta} (P_{i,\zeta} - \langle P_{i,\zeta} \rangle) \right)^2 \right\rangle. \quad (8.18)$$

Defining $\delta P_{i,\zeta} \equiv P_{i,\zeta} - \langle P_{i,\zeta} \rangle$ we obtain

$$\begin{aligned}\operatorname{Var} p_\zeta^\circ &= \left\langle \left(\sum_i \omega_{i,\zeta} \delta P_{i,\zeta} \right)^2 \right\rangle = \\ &= \left\langle \sum_i \omega_{i,\zeta}^2 \delta P_{i,\zeta}^2 \right\rangle + 2 \left\langle \sum_{j \neq k} \omega_{j,\zeta} \omega_{k,\zeta} \delta P_{j,\zeta} \delta P_{k,\zeta} \right\rangle = \\ &= \sum_i \omega_{i,\zeta}^2 \left\langle \delta P_{i,\zeta}^2 \right\rangle + 2 \sum_{j \neq k} \omega_{j,\zeta} \omega_{k,\zeta} \left\langle \delta P_{j,\zeta} \delta P_{k,\zeta} \right\rangle\end{aligned} \tag{8.19}$$

We note that $\left\langle \delta P_{i,\zeta}^2 \right\rangle = \operatorname{Var}(P_{i,\zeta})$. Assuming that simulations j and k are uncorrelated[8], $\langle \delta P_{j,\zeta} \delta P_{k,\zeta} \rangle = 0$ and therefore

$$\operatorname{Var} p_\zeta^\circ = \sum_i \omega_{i,\zeta}^2 \operatorname{Var}(P_{i,\zeta}). \tag{8.20}$$

We can minimise the variance with respect to the set of weights $\{\omega_{i,\zeta}\}$ subjects to the constraints $\sum_i \omega_{i,\zeta} = 1$ by using a Lagrange multiplier λ_ζ. The quantity to minimise is therefore

$$L \equiv \sum_i \omega_{i,\zeta}^2 \operatorname{Var}(P_{i,\zeta}) + \lambda_\zeta \sum_i \omega_{i,\zeta}, \tag{8.21}$$

whose derivative reads

$$\frac{\partial L}{\partial \omega_{j,\zeta}} = 2\omega_{j,\zeta} \operatorname{Var}(P_{j,\zeta}) + \lambda_\zeta \tag{8.22}$$

whence we obtain

$$\omega_{j,\zeta} = -\frac{\lambda_\zeta}{2 \operatorname{Var}(P_{j,\zeta})}. \tag{8.23}$$

Applying the normalisation constraint we find

$$\sum_j \omega_{j,\zeta} = -\sum_j \frac{\lambda_\zeta}{2 \operatorname{Var}(P_{j,\zeta})} = 1, \tag{8.24}$$

which can be solved for λ_ζ, yielding

$$\lambda_\zeta = -2 \sum_j \operatorname{Var}(P_{j,\zeta}). \tag{8.25}$$

Using this latter relation in Eq. (8.23) gives the optimal value for weight $\omega_{j,\zeta}$:

$$\omega_{j,\zeta} = \frac{\sum_k \operatorname{Var}(P_{k,\zeta})}{\operatorname{Var}(P_{j,\zeta})}. \tag{8.26}$$

We now need to write $\operatorname{Var}(P_{k,\zeta})$ in terms of the simulation output. Recalling that $\operatorname{Var}(aX) = a^2 \operatorname{Var}(x)$ and using Eq. (8.16) we can write

$$\operatorname{Var}(P_{k,\zeta}) = \frac{\operatorname{Var}(m_{k,\zeta})}{M_k^2 u_{k,\zeta}^2 f_k^2}. \tag{8.27}$$

The variance of the original histograms $m_{k,\zeta}$ can be approximated by first considering that the probability of having m counts in a specific histogram bin is given by the binomial distribution,

$$P(m) = \binom{M}{m} p^m (1-p)^{M-m}, \tag{8.28}$$

[8]This uncorrelation seems obvious, but there are computational techniques such as parallel tempering where this assumption may not hold.

where q is the probability of the event associated with the bin and M is the total number of counts stored in the histogram. According to the Poisson limit theorem, in the limit of large M and small p[9] the binomial distribution can be approximated by the Poisson distribution

$$P(m) = e^{-Mp}\frac{(Mp)^m}{m!}, \tag{8.29}$$

whose mean and variance are both equal to mp. We connect this result with our derivation by noting that the probability p for bin ζ and simulation i is the *biased* probability $q^b_{i,\zeta} = u_{i,\zeta}p^\circ_\zeta f_i$, and therefore

$$\mathrm{Var}(m_{k,\zeta}) = M_i u_{i,\zeta} p^\circ_\zeta f_i. \tag{8.30}$$

Using this relation in Eq. (8.27) yields

$$\mathrm{Var}(P_{k,\zeta}) = \frac{p^\circ_\zeta}{M_k u_{k,\zeta} f_k}, \tag{8.31}$$

which gives for the weights

$$\omega_{j,\zeta} = \frac{p^\circ_\zeta M_j u_{j,\zeta} f_j}{\sum_k p^\circ_\zeta M_k u_{k,\zeta} f_k} = \frac{M_j u_{j,\zeta} f_j}{\sum_k M_k u_{k,\zeta} f_k}. \tag{8.32}$$

Substituting these weights in Eq. (8.17) and using the fact that $M_j u_{j,\zeta} f_j P_{j,\zeta} = m_{j,\zeta}$ (see Eq. (8.16)) we obtain the WHAM set of equations

$$p^\circ_\zeta = \frac{\sum_j M_j u_{j,\zeta} f_j P_{j,\zeta}}{\sum_k M_k u_{k,\zeta} f_k} = \frac{\sum_j m_{j,\zeta}}{\sum_k M_k u_{k,\zeta} f_k}. \tag{8.33}$$

Note that this is a system of $\#_{\mathrm{bins}}$ *non-linear* equations, which are complemented by the $\#_{\mathrm{sims}}$ equations (8.15) that define the f_i. This sytem of equations can be solved either with non-linear solvers (such as scipy's fsolve) or iteratively, setting the $\{f_i\}$ to some initial values (*e.g.* all equal to 1), using them to evaluate the $\{p^\circ_\zeta\}$ through Eq. (8.33), which in turn are used to update $\{f_i\}$, and so on, repeating the process until convergence is achieved.

8.3.6 A real-world example

As discussed in the chapter on coarse-grained force fields, the effective interaction between two objects composed of multiple interacting units (atoms, molecules or beads) can be estimated in the dilute limit (*i.e.* at low density) as

$$U_{\mathrm{eff}}(R) = -k_B T \ln g(R), \tag{8.34}$$

where R is the distance between the two objects (defined *e.g.* as the distance between the two centres of mass) and $g(R)$ is the associated radial distribution function. For complicated objects composed of many parts, estimating $g(R)$ with sufficient accuracy through means of unbiased simulations requires an ungodly amount of computation time and is therefore unfeasible. Here I show an example where this issue has been overcome by using umbrella sampling.

Using the language introduced in this section, R is the reaction coordinate and $g(R) = P(R)/4\pi R^2$ the observable of interest, where $P(R)$ is the marginal probability density.

Figure 8.6: The results of umbrella sampling simulations: the raw data is unbiased and then combined together to yield the final free-energy profile. Here the reaction coordinate R is the distance between the centres of mass of two polymers.

[9] Both assumptions are reasonable for most real-world examples, since M should be large to have a good statistics, and p should be small in a well-sampled, biased simulation, where many bins should have non-zero counts.

Figure **??** shows the results of umbrella sampling simulations of a system composed of two polymer chains, where the chosen reaction coordinate is the distance between the two centres of mass, R, and the final output is the effective chain-chain interaction as a function of R. Figure **??** shows a snapshot of the two chains, Figure **??** shows the raw (biased) $g_i^b(R)$ data for all the windows i, and Figure **??** contains the $g_i^u(R)$, unbiased according to Eq. (8.10). Finally, Figure 8.6 contains the effective interaction, obtained with the WHAM method and shifted so that it vanishes at large distances.

8.4 Thermodynamic & Hamiltonian integration

8.5 Metadynamics

8.6 Forward-flux sampling

Warning
Remove?

Bibliography

P. D. Adams, P. V. Afonine, K. Baskaran, H. M. Berman, J. Berrisford, G. Bricogne, D. G. Brown, S. K. Burley, M. Chen, Z. Feng, C. Flensburg, A. Gutmanas, J. C. Hoch, Y. Ikegawa, Y. Kengaku, E. Krissinel, G. Kurisu, Y. Liang, D. Liebschner, L. Mak, J. L. Markley, N. W. Moriarty, G. N. Murshudov, M. Noble, E. Peisach, I. Persikova, B. K. Poon, O. V. Sobolev, E. L. Ulrich, S. Velankar, C. Vonrhein, J. Westbrook, M. Wojdyr, M. Yokochi, and J. Y. Young. Announcing mandatory submission of PDBx/mmCIF format files for crystallographic depositions to the Protein Data Bank (PDB). *Acta Crystallographica Section D Structural Biology*, 75(4):451–454, 4 2019. ISSN 2059-7983. doi: 10.1107/s2059798319004522. URL `http://dx.doi.org/10.1107/S2059798319004522`.

M. P. Allen and D. J. Tildesley. *Computer simulation of liquids*. Oxford university press, 2017.

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 10 1990. ISSN 0022-2836. doi: 10.1016/s0022-2836(05)80360-2. URL `http://dx.doi.org/10.1016/S0022-2836(05)80360-2`.

H. C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *The Journal of Chemical Physics*, 72(4):2384–2393, 2 1980. ISSN 1089-7690. doi: 10.1063/1.439486. URL `http://dx.doi.org/10.1063/1.439486`.

P. W. Anderson. More Is Different: Broken symmetry and the nature of the hierarchical structure of science. *Science*, 177(4047):393–396, 8 1972. ISSN 1095-9203. doi: 10.1126/science.177.4047.393. URL `http://dx.doi.org/10.1126/science.177.4047.393`.

C. B. Anfinsen. Principles that Govern the Folding of Protein Chains. *Science*, 181(4096):223–230, 7 1973. ISSN 1095-9203. doi: 10.1126/science.181.4096.223. URL `http://dx.doi.org/10.1126/science.181.4096.223`.

C. B. Anfinsen, E. Haber, M. Sela, and F. H. White. The KINETICS OF FORMATION OF NATIVE RIBONUCLEASE DURING OXIDATION OF THE REDUCED POLYPEPTIDE CHAIN. *Proceedings of the National Academy of Sciences*, 47(9):1309–1314, 9 1961. ISSN 1091-6490. doi: 10.1073/pnas.47.9.1309. URL `http://dx.doi.org/10.1073/pnas.47.9.1309`.

A. Badasyan, A. Giacometti, R. Podgornik, Y. Mamasakhlisov, and V. Morozov. Helix-coil transition in terms of Potts-like spins. *The European Physical Journal E*, 36(5), 5 2013. ISSN 1292-895X. doi: 10.1140/epje/i2013-13046-7. URL `http://dx.doi.org/10.1140/epje/i2013-13046-7`.

A. V. Badasyan, A. Giacometti, Y. S. Mamasakhlisov, V. F. Morozov, and A. S. Benight. Microscopic formulation of the Zimm-Bragg model for the helix-coil transition. *Physical Review E*, 81(2), 2 2010. ISSN 1550-2376. doi: 10.1103/physreve.81.021921. URL `http://dx.doi.org/10.1103/PhysRevE.81.021921`.

A. Banerjee, M. Anand, S. Kalita, and M. Ganji. Single-molecule analysis of DNA base-stacking energetics using patterned DNA nanostructures. *Nature Nanotechnology*, 18(12):1474–1482, 8 2023. ISSN 1748-3395. doi: 10.1038/s41565-023-01485-1. URL `http://dx.doi.org/10.1038/s41565-023-01485-1`.

C. Bartels and M. Karplus. Multidimensional adaptive umbrella sampling: Applications to main chain and side chain peptide conformations. *Journal of Computational Chemistry*, 18(12):1450–1462, 9 1997. ISSN 1096-987X. doi: 10.1002/(sici)1096-987x(199709)18:12<1450::aid-jcc3>3.0.co;2-i. URL http://dx.doi.org/10.1002/(SICI)1096-987X(199709)18:12%3C1450::AID-JCC3%3E3.0.CO;2-I.

C. Bartels and M. Karplus. Probability Distributions for Complex Systems: Adaptive Umbrella Sampling of the Potential Energy. *The Journal of Physical Chemistry B*, 102(5):865–880, 1 1998. ISSN 1520-5207. doi: 10.1021/jp972280j. URL http://dx.doi.org/10.1021/jp972280j.

H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *The Journal of Chemical Physics*, 81(8):3684–3690, 10 1984. ISSN 1089-7690. doi: 10.1063/1.448118. URL http://dx.doi.org/10.1063/1.448118.

D. S. Berkholz, C. M. Driggers, M. V. Shapovalov, R. L. Dunbrack, and P. A. Karplus. Nonplanar peptide bonds in proteins are common and conserved but not biased toward active sites. *Proceedings of the National Academy of Sciences*, 109(2):449–453, 12 2011. ISSN 1091-6490. doi: 10.1073/pnas.1107115108. URL http://dx.doi.org/10.1073/pnas.1107115108.

S. H. Bernhart, U. Mückstein, and I. L. Hofacker. Rna Accessibility in cubic time. *Algorithms for Molecular Biology*, 6(1), 3 2011. ISSN 1748-7188. doi: 10.1186/1748-7188-6-3. URL http://dx.doi.org/10.1186/1748-7188-6-3.

W. Bialek. *Biophysics: searching for principles*. Princeton University Press, 2012.

G. Blackshields, F. Sievers, W. Shi, A. Wilm, and D. G. Higgins. Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms for Molecular Biology*, 5 (1), 5 2010. ISSN 1748-7188. doi: 10.1186/1748-7188-5-21. URL http://dx.doi.org/10.1186/1748-7188-5-21.

L. Böttcher and H. J. Herrmann. *Computational Statistical Physics*. Cambridge University Press, 2021.

D. R. Bowler and T. Miyazaki. \mathcalO(N) methods in electronic structure calculations. *Reports on Progress in Physics*, 75(3):036503, 2 2012. ISSN 1361-6633. doi: 10.1088/0034-4885/75/3/036503. URL http://dx.doi.org/10.1088/0034-4885/75/3/036503.

S. E. Brenner, C. Chothia, and T. J. P. Hubbard. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proceedings of the National Academy of Sciences*, 95(11):6073–6078, 5 1998. ISSN 1091-6490. doi: 10.1073/pnas.95.11.6073. URL http://dx.doi.org/10.1073/pnas.95.11.6073.

G. Bussi, D. Donadio, and M. Parrinello. Canonical sampling through velocity rescaling. *The Journal of Chemical Physics*, 126(1), 1 2007. ISSN 1089-7690. doi: 10.1063/1.2408420. URL http://dx.doi.org/10.1063/1.2408420.

E. Callaway. 'It will change everything': Deepmind's AI makes gigantic leap in solving protein structures. *Nature*, 588(7837):203–204, 11 2020. ISSN 1476-4687. doi: 10.1038/d41586-020-03348-4. URL http://dx.doi.org/10.1038/d41586-020-03348-4.

R. Car and M. Parrinello. Unified Approach for Molecular Dynamics and Density-Functional Theory. *Physical Review Letters*, 55(22):2471–2474, 11 1985. ISSN 0031-9007. doi: 10.1103/physrevlett.55.2471. URL http://dx.doi.org/10.1103/PhysRevLett.55.2471.

F. Cava, H. Lam, M. A. de Pedro, and M. K. Waldor. Emerging knowledge of regulatory roles of d-amino acids in bacteria. *Cellular and Molecular Life Sciences*, 68(5):817–831, 12 2010. ISSN 1420-9071. doi: 10.1007/s00018-010-0571-8. URL http://dx.doi.org/10.1007/s00018-010-0571-8.

H. S. Chan and K. A. Dill. Compact polymers. *Macromolecules*, 22(12):4559–4573, 12 1989. ISSN 1520-5835. doi: 10.1021/ma00202a031. URL http://dx.doi.org/10.1021/ma00202a031.

C. Chothia. Conformation of twisted β-pleated sheets in proteins. *Journal of Molecular Biology*, 75 (2):295–302, 4 1973. ISSN 0022-2836. doi: 10.1016/0022-2836(73)90022-3. URL http://dx.doi.org/10.1016/0022-2836(73)90022-3.

C. Chothia. One thousand families for the molecular biologist. *Nature*, 357(6379):543–544, 6 1992. ISSN 1476-4687. doi: 10.1038/357543a0. URL http://dx.doi.org/10.1038/357543a0.

M. Cobb. 60 years ago, Francis Crick changed the logic of biology. *PLOS Biology*, 15(9):e2003243, 9 2017. ISSN 1545-7885. doi: 10.1371/journal.pbio.2003243. URL http://dx.doi.org/10.1371/journal.pbio.2003243.

F. CRICK. Central Dogma of Molecular Biology. *Nature*, 227(5258):561–563, 8 1970. ISSN 1476-4687. doi: 10.1038/227561a0. URL http://dx.doi.org/10.1038/227561a0.

F. H. C. Crick. The Fourier transform of a coiled-coil. *Acta Crystallographica*, 6(8):685–689, 9 1953. ISSN 0365-110X. doi: 10.1107/s0365110x53001952. URL http://dx.doi.org/10.1107/S0365110X53001952.

D. Dalkara, L. C. Byrne, R. R. Klimczak, M. Visel, L. Yin, W. H. Merigan, J. G. Flannery, and D. V. Schaffer. In Vivo–Directed Evolution of a New Adeno-Associated Virus for Therapeutic Outer Retinal Gene Delivery from the Vitreous. *Science Translational Medicine*, 5(189), 6 2013. ISSN 1946-6242. doi: 10.1126/scitranslmed.3005708. URL http://dx.doi.org/10.1126/scitranslmed.3005708.

T. Darden, D. York, and L. Pedersen. Particle mesh Ewald: An Nlog(N) method for Ewald sums in large systems. *The Journal of Chemical Physics*, 98(12):10089–10092, 6 1993. ISSN 1089-7690. doi: 10.1063/1.464397. URL http://dx.doi.org/10.1063/1.464397.

K. A. Dill. Dominant forces in protein folding. *Biochemistry*, 29(31):7133–7155, 8 1990. ISSN 1520-4995. doi: 10.1021/bi00483a001. URL http://dx.doi.org/10.1021/bi00483a001.

P. Doty and J. T. Yang. Polypeptides. VII. POLY-γ-BENZYL-L-GLUTAMATE: The HELIX-COIL TRANSITION IN SOLUTION1. *Journal of the American Chemical Society*, 78(2):498–500, 1 1956. ISSN 1520-5126. doi: 10.1021/ja01583a070. URL http://dx.doi.org/10.1021/ja01583a070.

S. R. Eddy. Where did the BLOSUM62 alignment score matrix come from? *Nature Biotechnology*, 22(8):1035–1036, 8 2004. ISSN 1546-1696. doi: 10.1038/nbt0804-1035. URL http://dx.doi.org/10.1038/nbt0804-1035.

U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen. A smooth particle mesh Ewald method. *The Journal of Chemical Physics*, 103(19):8577–8593, 11 1995. ISSN 1089-7690. doi: 10.1063/1.470117. URL http://dx.doi.org/10.1063/1.470117.

M. G. Evans and M. Polanyi. Some applications of the transition state method to the calculation of reaction velocities, especially in solution. *Transactions of the Faraday Society*, 31:875, 1935. ISSN 0014-7672. doi: 10.1039/tf9353100875. URL http://dx.doi.org/10.1039/TF9353100875.

P. P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Annalen der Physik*, 369(3):253–287, 1 1921. ISSN 1521-3889. doi: 10.1002/andp.19213690304. URL http://dx.doi.org/10.1002/andp.19213690304.

H. Eyring. The Activated Complex in Chemical Reactions. *The Journal of Chemical Physics*, 3(2): 107–115, 2 1935. ISSN 1089-7690. doi: 10.1063/1.1749604. URL http://dx.doi.org/10.1063/1.1749604.

A. V. Finkelstein and O. Ptitsyn. *Protein physics: a course of lectures*. Elsevier, 2002.

A. V. Finkelstein and O. Ptitsyn. *Protein physics: a course of lectures*. Elsevier, 2016.

H. Flöckner, M. Braxenthaler, P. Lackner, M. Jaritz, M. Ortner, and M. J. Sippl. Progress in fold recognition. *Proteins: Structure, Function, and Bioinformatics*, 23(3):376–386, 11 1995. ISSN 1097-0134. doi: 10.1002/prot.340230311. URL http://dx.doi.org/10.1002/prot.340230311.

M. Fodje and S. Al-Karadaghi. Occurrence, conformational features and amino acid propensities for the π-helix. *Protein Engineering, Design and Selection*, 15(5):353–358, 5 2002. ISSN 1741-0126. doi: 10.1093/protein/15.5.353. URL http://dx.doi.org/10.1093/protein/15.5.353.

M. E. Fornace, J. Huang, C. T. Newman, N. J. Porubsky, M. B. Pierce, and N. A. Pierce. Nupack: Analysis and Design of Nucleic Acid Structures, Devices, and Systems. 11 2022. doi: 10.26434/chemrxiv-2022-xv98l. URL http://dx.doi.org/10.26434/chemrxiv-2022-xv98l.

FOSSEY, S.A.; NEMETHY, G.; GIBSON, K.D.; SCHERAGA, H.A. Conformational ENERGY STUDIES OF BETA-SHEETS OF MODEL SILK FIBROIN PEPTIDES. I. SHEETS OF POLY(ALA-GLY) CHAINS, 1991. URL https://www.modelarchive.org/doi/10.5452/ma-cjmb5.

D. Frenkel and B. Smit. *Understanding molecular simulation: from algorithms to applications*. Elsevier, 2023.

R. Geyer, J. R. Jambeck, and K. L. Law. Production, use, and fate of all plastics ever made. *Science Advances*, 3(7), 7 2017. ISSN 2375-2548. doi: 10.1126/sciadv.1700782. URL http://dx.doi.org/10.1126/sciadv.1700782.

F. Giustino. *Materials modelling using density functional theory: properties and predictions*. Oxford University Press, 2014.

S. S. Gomez and L. Rovigatti. Diffusion, viscosity, and linear rheology of valence-limited disordered fluids. *The Journal of Chemical Physics*, 160(18), 5 2024. ISSN 1089-7690. doi: 10.1063/5.0209151. URL http://dx.doi.org/10.1063/5.0209151.

L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 12 1987. ISSN 0021-9991. doi: 10.1016/0021-9991(87)90140-9. URL http://dx.doi.org/10.1016/0021-9991(87)90140-9.

J.-P. Hansen and I. R. McDonald. *Theory of simple liquids: with applications to soft matter*. Academic press, 2013.

A. S. Hauser, S. Chavali, I. Masuho, L. J. Jahn, K. A. Martemyanov, D. E. Gloriam, and M. M. Babu. Pharmacogenomics of GPCR Drug Targets. *Cell*, 172(1–2):41–54.e19, 1 2018. ISSN 0092-8674. doi: 10.1016/j.cell.2017.11.033. URL http://dx.doi.org/10.1016/j.cell.2017.11.033.

R. Helling, H. Li, R. Mélin, J. Miller, N. Wingreen, C. Zeng, and C. Tang. The designability of protein structures. *Journal of Molecular Graphics and Modelling*, 19(1):157–167, 2 2001. ISSN 1093-3263. doi: 10.1016/s1093-3263(00)00137-6. URL http://dx.doi.org/10.1016/S1093-3263(00)00137-6.

S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 11 1992. ISSN 1091-6490. doi: 10.1073/pnas.89.22.10915. URL http://dx.doi.org/10.1073/pnas.89.22.10915.

J. Hénin, T. Lelièvre, M. R. Shirts, O. Valsson, and L. Delemotte. Enhanced Sampling Methods for Molecular Dynamics Simulations [Article v1.0]. *Living Journal of Computational Molecular Science*, 4(1), 2022. ISSN 2575-6524. doi: 10.33011/livecoms.4.1.1583. URL http://dx.doi.org/10.33011/livecoms.4.1.1583.

P. Hohenberg and W. Kohn. Inhomogeneous Electron Gas. *Physical Review*, 136(3B):B864–B871, 11 1964. ISSN 0031-899X. doi: 10.1103/physrev.136.b864. URL http://dx.doi.org/10.1103/PhysRev.136.B864.

W. G. Hoover. Constant-pressure equations of motion. *Physical Review A*, 34(3):2499–2500, 9 1986. ISSN 0556-2791. doi: 10.1103/physreva.34.2499. URL http://dx.doi.org/10.1103/PhysRevA.34.2499.

J. Hutter. Car–Parrinello molecular dynamics. *WIREs Computational Molecular Science*, 2(4):604–612, 9 2011. ISSN 1759-0884. doi: 10.1002/wcms.90. URL http://dx.doi.org/10.1002/wcms.90.

J. N. Israelachvili. *Intermolecular and surface forces*. Academic press, 2011.

J. Jin, A. J. Pak, A. E. P. Durumeric, T. D. Loose, and G. A. Voth. Bottom-up Coarse-Graining: Principles and Perspectives. *Journal of Chemical Theory and Computation*, 18(10):5759–5791, 9 2022. ISSN 1549-9626. doi: 10.1021/acs.jctc.2c00643. URL http://dx.doi.org/10.1021/acs.jctc.2c00643.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 7 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL http://dx.doi.org/10.1038/s41586-021-03819-2.

S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences*, 87(6):2264–2268, 3 1990. ISSN 1091-6490. doi: 10.1073/pnas.87.6.2264. URL http://dx.doi.org/10.1073/pnas.87.6.2264.

M. Kellis. *Computational Biology: Genomes, Networks, Evolution. MIT course 6.047 / 6.878*. MIT OpenCourseWare, 2016. URL https://ocw.mit.edu/courses/6-047-computational-biology-fall-2015/pages/readings/.

W. Kohn and L. J. Sham. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical Review*, 140(4A):A1133–A1138, 11 1965. ISSN 0031-899X. doi: 10.1103/physrev.140.a1133. URL http://dx.doi.org/10.1103/PhysRev.140.A1133.

F. Korkmaz-Özkan, S. Köster, W. Kühlbrandt, W. Mäntele, and Ö. Yildiz. Correlation between the OmpG Secondary Structure and Its pH-Dependent Alterations Monitored by FTIR. *Journal of Molecular Biology*, 401(1):56–67, 8 2010. ISSN 0022-2836. doi: 10.1016/j.jmb.2010.06.015. URL http://dx.doi.org/10.1016/j.jmb.2010.06.015.

F. W. Kotch, I. A. Guzei, and R. T. Raines. Stabilization of the Collagen Triple Helix byO-Methylation of Hydroxyproline Residues. *Journal of the American Chemical Society*, 130(10):2952–2953, 2 2008. ISSN 1520-5126. doi: 10.1021/ja800225k. URL http://dx.doi.org/10.1021/ja800225k.

S. Kumar, J. M. Rosenberg, D. Bouzida, R. H. Swendsen, and P. A. Kollman. The weighted histogram analysis method for freeenergy calculations on biomolecules. I. The method. *Journal of Computational Chemistry*, 13(8):1011–1021, 10 1992. ISSN 1096-987X. doi: 10.1002/jcc.540130812. URL http://dx.doi.org/10.1002/jcc.540130812.

F. Lapenta, J. Aupič, Ž. Strmšek, and R. Jerala. Coiled coil protein origami: from modular design principles towards biotechnological applications. *Chemical Society Reviews*, 47(10):3530–3542, 2018. ISSN 1460-4744. doi: 10.1039/c7cs00822h. URL http://dx.doi.org/10.1039/C7CS00822H.

K. F. Lau and K. A. Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10):3986–3997, 10 1989. ISSN 1520-5835. doi: 10.1021/ma00200a030. URL http://dx.doi.org/10.1021/ma00200a030.

A. R. Leach. *Molecular modelling: principles and applications*. Pearson education, 2001.

C.-H. Lee, M.-S. Kim, B. M. Chung, D. J. Leahy, and P. A. Coulombe. Structural basis for heteromeric assembly and perinuclear organization of keratin filaments. *Nature Structural amp; Molecular Biology*, 19(7):707–715, 6 2012. ISSN 1545-9985. doi: 10.1038/nsmb.2330. URL http://dx.doi.org/10.1038/nsmb.2330.

A. L. Lehninger, D. L. Nelson, and M. M. Cox. *Lehninger principles of biochemistry*. Macmillan, 2005.

H. Li, R. Helling, C. Tang, and N. Wingreen. Emergence of Preferred Structures in a Simple Model of Protein Folding. *Science*, 273(5275):666–669, 8 1996. ISSN 1095-9203. doi: 10.1126/science.273.5275.666. URL http://dx.doi.org/10.1126/science.273.5275.666.

F. London. Zur Theorie und Systematik der Molekularkrfte. *Zeitschrift fr Physik*, 63(3–4):245–279, 3 1930. ISSN 1434-601X. doi: 10.1007/bf01421741. URL http://dx.doi.org/10.1007/BF01421741.

Z. J. Lu, D. H. Turner, and D. H. Mathews. A set of nearest neighbor parameters for predicting the enthalpy change of RNA secondary structure formation. *Nucleic Acids Research*, 34(17):4912–4924, 8 2006. ISSN 1362-4962. doi: 10.1093/nar/gkl472. URL http://dx.doi.org/10.1093/nar/gkl472.

G. J. Martyna, M. L. Klein, and M. Tuckerman. Nosé–Hoover chains: The canonical ensemble via continuous dynamics. *The Journal of Chemical Physics*, 97(4):2635–2643, 8 1992. ISSN 1089-7690. doi: 10.1063/1.463940. URL http://dx.doi.org/10.1063/1.463940.

G. J. Martyna, D. J. Tobias, and M. L. Klein. Constant pressure molecular dynamics algorithms. *The Journal of Chemical Physics*, 101(5):4177–4189, 9 1994. ISSN 1089-7690. doi: 10.1063/1.467468. URL http://dx.doi.org/10.1063/1.467468.

I. Miklós. Introduction to algorithms in bioinformatics. 2016.

M. Mirdita, K. Schütze, Y. Moriwaki, L. Heo, S. Ovchinnikov, and M. Steinegger. Colabfold: making protein folding accessible to all. *Nature Methods*, 19(6):679–682, 5 2022. ISSN 1548-7105. doi: 10.1038/s41592-022-01488-1. URL http://dx.doi.org/10.1038/s41592-022-01488-1.

S. Miyazawa and R. L. Jernigan. Estimation of effective interresidue contact energies from protein crystal structures: quasi-chemical approximation. *Macromolecules*, 18(3):534–552, 3 1985. ISSN 1520-5835. doi: 10.1021/ma00145a039. URL http://dx.doi.org/10.1021/ma00145a039.

S. MorenoHernández and M. Levitt. Comparative modeling and proteinlike features of hydrophobic–polar models on a twodimensional lattice. *Proteins: Structure, Function, and Bioinformatics*, 80 (6):1683–1693, 4 2012. ISSN 1097-0134. doi: 10.1002/prot.24067. URL http://dx.doi.org/10.1002/prot.24067.

A. G. Murzin and A. V. Finkelstein. General architecture of the α-helical globule. *Journal of Molecular Biology*, 204(3):749–769, 12 1988. ISSN 0022-2836. doi: 10.1016/0022-2836(88)90366-x. URL http://dx.doi.org/10.1016/0022-2836(88)90366-X.

N. Nagano, M. Ota, and K. Nishikawa. Strong hydrophobic nature of cysteine residues in proteins. *FEBS Letters*, 458(1):69–71, 9 1999. ISSN 1873-3468. doi: 10.1016/s0014-5793(99)01122-9. URL http://dx.doi.org/10.1016/S0014-5793(99)01122-9.

S. Nosé. A molecular dynamics method for simulations in the canonical ensemble. *Molecular Physics*, 52(2):255–268, 6 1984. ISSN 1362-3028. doi: 10.1080/00268978400101201. URL http://dx.doi.org/10.1080/00268978400101201.

R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman. Algorithms for Loop Matchings. *SIAM Journal on Applied Mathematics*, 35(1):68–82, 7 1978. ISSN 1095-712X. doi: 10.1137/0135006. URL http://dx.doi.org/10.1137/0135006.

J. N. Onuchic, P. G. Wolynes, Z. Luthey-Schulten, and N. D. Socci. Toward an outline of the topography of a realistic protein-folding funnel. *Proceedings of the National Academy of Sciences*, 92(8):3626–3630, 4 1995. ISSN 1091-6490. doi: 10.1073/pnas.92.8.3626. URL http://dx.doi.org/10.1073/pnas.92.8.3626.

J. N. Onuchic, Z. Luthey-Schulten, and P. G. Wolynes. Theory OF PROTEIN FOLDING: The Energy Landscape Perspective. *Annual Review of Physical Chemistry*, 48(1):545–600, 10 1997. ISSN 1545-1593. doi: 10.1146/annurev.physchem.48.1.545. URL http://dx.doi.org/10.1146/annurev.physchem.48.1.545.

M. Parrinello and A. Rahman. Crystal Structure and Pair Potentials: A Molecular-Dynamics Study. *Physical Review Letters*, 45(14):1196–1199, 10 1980. ISSN 0031-9007. doi: 10.1103/physrevlett.45.1196. URL http://dx.doi.org/10.1103/PhysRevLett.45.1196.

J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized Gradient Approximation Made Simple. *Physical Review Letters*, 77(18):3865–3868, 10 1996. ISSN 1079-7114. doi: 10.1103/physrevlett.77.3865. URL http://dx.doi.org/10.1103/PhysRevLett.77.3865.

J. V. Pratap, B. F. Luisi, and C. R. Calladine. Geometric principles in the assembly of α-helical bundles. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1993):20120369, 6 2013. ISSN 1471-2962. doi: 10.1098/rsta.2012.0369. URL http://dx.doi.org/10.1098/rsta.2012.0369.

E. Protozanova, P. Yakovchuk, and M. D. Frank-Kamenetskii. Stacked–Unstacked Equilibrium at the Nick Site of DNA. *Journal of Molecular Biology*, 342(3):775–785, 9 2004. ISSN 0022-2836. doi: 10.1016/j.jmb.2004.07.075. URL http://dx.doi.org/10.1016/j.jmb.2004.07.075.

O. Ptitsyn, A. Finkelstein, and A. Murzin. Structural model for interferons. *FEBS Letters*, 186(2):143–148, 7 1985. ISSN 1873-3468. doi: 10.1016/0014-5793(85)80697-9. URL http://dx.doi.org/10.1016/0014-5793(85)80697-9.

P. Pulay. Ab initiocalculation of force constants and equilibrium geometries in polyatomic molecules: I. Theory. *Molecular Physics*, 17(2):197–204, 1 1969. ISSN 1362-3028. doi: 10.1080/00268976900100941. URL http://dx.doi.org/10.1080/00268976900100941.

M. Raden, M. M. Mohamed, S. M. Ali, and R. Backofen. Interactive implementations of thermodynamics-based RNA structure and RNA–RNA interaction prediction approaches for example-driven teaching. *PLOS Computational Biology*, 14(8):e1006341, 8 2018. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006341. URL http://dx.doi.org/10.1371/journal.pcbi.1006341.

G. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, 7(1):95–99, 7 1963. ISSN 0022-2836. doi: 10.1016/s0022-2836(63)80023-6. URL http://dx.doi.org/10.1016/S0022-2836(63)80023-6.

A. Rich. The Era of RNA Awakening: Structural biology of RNA in the early years. *Quarterly Reviews of Biophysics*, 42(2):117–137, 5 2009. ISSN 1469-8994. doi: 10.1017/s0033583509004776. URL http://dx.doi.org/10.1017/S0033583509004776.

L. Rovigatti, N. Gnan, A. Parola, and E. Zaccarelli. How soft repulsion enhances the depletion mechanism. *Soft Matter*, 11(4):692–700, 2015. ISSN 1744-6848. doi: 10.1039/c4sm02218a. URL http://dx.doi.org/10.1039/C4SM02218A.

M. Rubinstein and R. H. Colby. *Polymer physics*. Oxford university press, 2003.

J. Ruiz-Franco, L. Rovigatti, and E. Zaccarelli. On the effect of the thermostat in non-equilibrium molecular dynamics simulations. *The European Physical Journal E*, 41(7), 7 2018. ISSN 1292-895X. doi: 10.1140/epje/i2018-11689-4. URL http://dx.doi.org/10.1140/epje/i2018-11689-4.

J. SantaLucia and D. Hicks. The Thermodynamics of DNA Structural Motifs. *Annual Review of Biophysics and Biomolecular Structure*, 33(1):415–440, 6 2004. ISSN 1545-4266. doi: 10.1146/annurev.biophys.32.110601.141800. URL http://dx.doi.org/10.1146/annurev.biophys.32.110601.141800.

T. Schlick. *Molecular modeling and simulation: an interdisciplinary guide*, volume 2. Springer, 2010.

R. Sharma, A. S. Patelli, L. De Bruin, and J. H. Maddocks. cgna+web : A Visual Interface to the cgNA+ Sequence-dependent Statistical Mechanics Model of Double-stranded Nucleic Acids. *Journal of Molecular Biology*, 435(14):167978, 7 2023. ISSN 0022-2836. doi: 10.1016/j.jmb.2023.167978. URL http://dx.doi.org/10.1016/j.jmb.2023.167978.

F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of highquality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(1), 1 2011. ISSN 1744-4292. doi: 10.1038/msb.2011.75. URL http://dx.doi.org/10.1038/msb.2011.75.

B. E. K. Snodin, F. Randisi, M. Mosayebi, P. Šulc, J. S. Schreck, F. Romano, T. E. Ouldridge, R. Tsukanov, E. Nir, A. A. Louis, and J. P. K. Doye. Introducing improved structural properties and salt dependence into a coarse-grained model of DNA. *The Journal of Chemical Physics*, 142(23), 6 2015. ISSN 1089-7690. doi: 10.1063/1.4921957. URL http://dx.doi.org/10.1063/1.4921957.

P. J. Stephens, F. J. Devlin, C. F. Chabalowski, and M. J. Frisch. Ab Initio Calculation of Vibrational Absorption and Circular Dichroism Spectra Using Density Functional Force Fields. *The Journal of Physical Chemistry*, 98(45):11623–11627, 11 1994. ISSN 1541-5740. doi: 10.1021/j100096a001. URL http://dx.doi.org/10.1021/j100096a001.

P. Šulc, J. P. Doye, and A. A. Louis. *Introduction to molecular simulation.* MIT Press, 2018.

I. M. Svishchev and P. G. Kusalik. Structure in liquid water: A study of spatial distribution functions. *The Journal of Chemical Physics*, 99(4):3049–3058, 8 1993. ISSN 1089-7690. doi: 10.1063/1.465158. URL http://dx.doi.org/10.1063/1.465158.

A. E. Torda. *Protein Threading*, pages 921–938. Humana Press, 2005. ISBN 9781592598908. doi: 10.1385/1-59259-890-0:921. URL http://dx.doi.org/10.1385/1-59259-890-0:921.

G. Torrie and J. Valleau. Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling. *Journal of Computational Physics*, 23(2):187–199, 2 1977. ISSN 0021-9991. doi: 10.1016/0021-9991(77)90121-8. URL http://dx.doi.org/10.1016/0021-9991(77)90121-8.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need, 2017. URL https://arxiv.org/abs/1706.03762.

P. Virnau and M. Müller. Calculation of free energy through successive umbrella sampling. *The Journal of Chemical Physics*, 120(23):10925–10930, 6 2004. ISSN 1089-7690. doi: 10.1063/1.1739216. URL http://dx.doi.org/10.1063/1.1739216.

D. Voet and A. Rich. *The Crystal Structures of Purines, Pyrimidines and Their Intermolecular Complexes*, pages 183–265. Elsevier, 1970. ISBN 9780125400107. doi: 10.1016/s0079-6603(08)60565-6. URL http://dx.doi.org/10.1016/S0079-6603(08)60565-6.

A. Vologodskii and M. D. Frank-Kamenetskii. Dna melting and energetics of the double helix. *Physics of Life Reviews*, 25:1–21, 8 2018. ISSN 1571-0645. doi: 10.1016/j.plrev.2017.11.012. URL http://dx.doi.org/10.1016/j.plrev.2017.11.012.

J. D. WATSON and F. H. C. CRICK. Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature*, 171(4356):737–738, 4 1953. ISSN 1476-4687. doi: 10.1038/171737a0. URL http://dx.doi.org/10.1038/171737a0.

T. M. Weaver. The πhelix translates structure into function. *Protein Science*, 9(1):201–206, 1 2000. ISSN 1469-896X. doi: 10.1110/ps.9.1.201. URL http://dx.doi.org/10.1110/ps.9.1.201.

P. Yakovchuk. Base-stacking and base-pairing contributions into thermal stability of the DNA double helix. *Nucleic Acids Research*, 34(2):564–574, 1 2006. ISSN 1362-4962. doi: 10.1093/nar/gkj454. URL http://dx.doi.org/10.1093/nar/gkj454.

Y. Zhou, C. K. Hall, and M. Karplus. The calorimetric criterion for a twostate process revisited. *Protein Science*, 8(5):1064–1074, 1 1999. ISSN 1469-896X. doi: 10.1110/ps.8.5.1064. URL http://dx.doi.org/10.1110/ps.8.5.1064.

B. H. Zimm and J. K. Bragg. Theory of the Phase Transition between Helix and Random Coil in Polypeptide Chains. *The Journal of Chemical Physics*, 31(2):526–535, 8 1959. ISSN 1089-7690. doi: 10.1063/1.1730390. URL http://dx.doi.org/10.1063/1.1730390.

L. Zubcevic and S.-Y. Lee. The role of π-helices in TRP channel gating. *Current Opinion in Structural Biology*, 58:314–323, 10 2019. ISSN 0959-440X. doi: 10.1016/j.sbi.2019.06.011. URL http://dx.doi.org/10.1016/j.sbi.2019.06.011.

J. Zuber, S. J. Schroeder, H. Sun, D. H. Turner, and D. H. Mathews. Nearest neighbor rules for RNA helix folding thermodynamics: improved end effects. *Nucleic Acids Research*, 50(9):5251–5262, 5 2022. ISSN 1362-4962. doi: 10.1093/nar/gkac261. URL http://dx.doi.org/10.1093/nar/gkac261.

M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981. ISSN 1362-4962. doi: 10.1093/nar/9.1.133. URL http://dx.doi.org/10.1093/nar/9.1.133.