Biophysics
A Computational Approach
Concepts, Models, Methods and Algorithms
Lectures 1 and 2: Methods

Dieter W. Heermann

Heidelberg University

August 14, 2017

Table of Contents

Introduction I

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

In these set of lectures you will find much more material than can be realistically taught in a course. Hence, each lecture will focus on particular aspects. The additional material may serve to further the understanding. We will cover a lot of ground starting off with basic computational methods. Clearly we will not be able to cover them all. Rather we focus on basic methods. To start off here is a very condensed account of the history of the development of computational methods pertinent to that we will encounter:

1953 Monte Carlo method applied to hard spheres [1]

1956 Molecular dynamics of hard spheres [2]

1964 Molecular dynamics of liquid argon [3]

1971 Molecular dynamics of liquid water [4]

1976 Simulation of protein dynamics [5]

1977 Non-Boltzmann sampling [6]

1992 Multi-Canonical Monte Carlo [7]

1999 Generalized and extended ensemble methods [8]

As always, some assumptions need to be made. Here it is assumed that there is

- a basic knowledge of statistical physics
- and some programming experience

to understand the methods and algorithms.
General literature for the course can for example be found in [9–12].

General Remarks I

Shown below are 256 Lennard-Jones particles using the parameters for argon at the density of 0.636 and reference temperature $T = 2.53$.

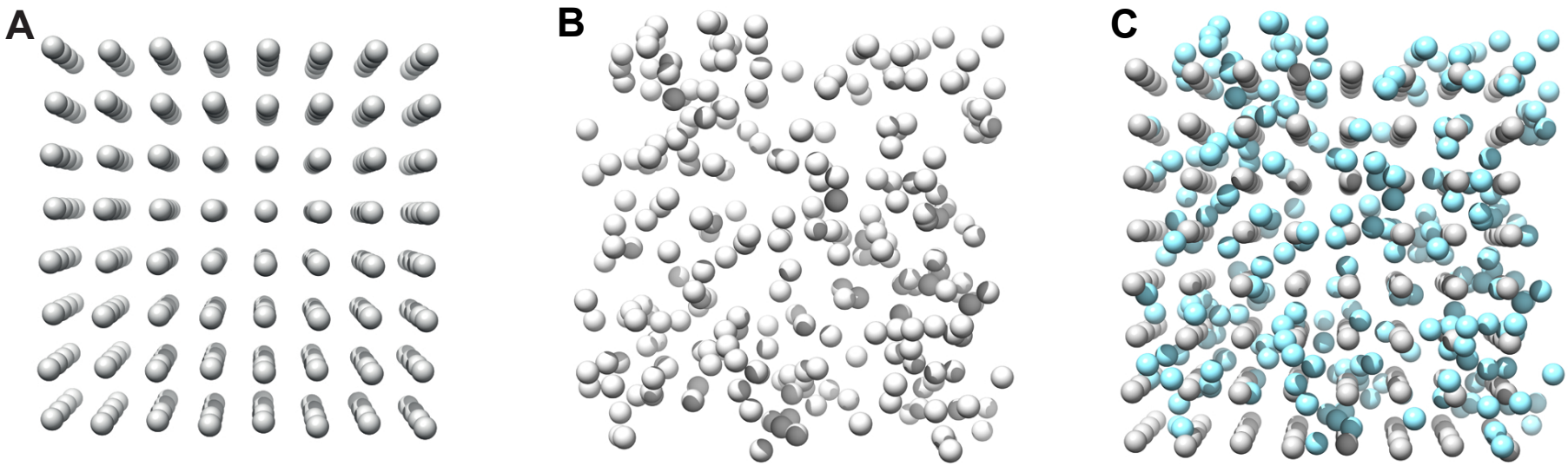


Figure: Panel A shows crystal structure, B shows a liquid structure and C an overlay of the two for comparison. Note that the radius with which the particles are depicted does not reflect the size of the particle.

General Remarks II

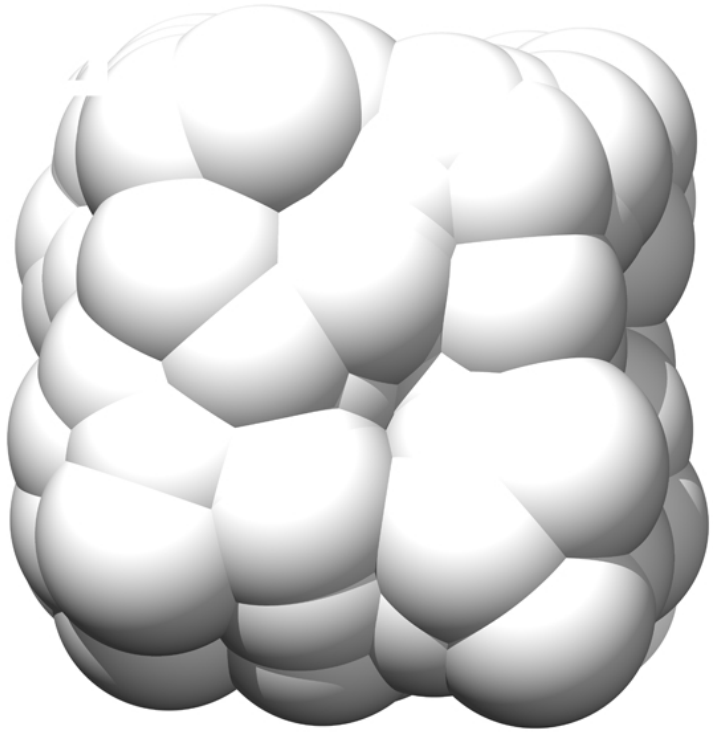


Figure: Lennard-Jones liquid where the particles are depicted with a radius where the Lennard-Jones potential is zero (see later in the section on force fields).

General Remarks III

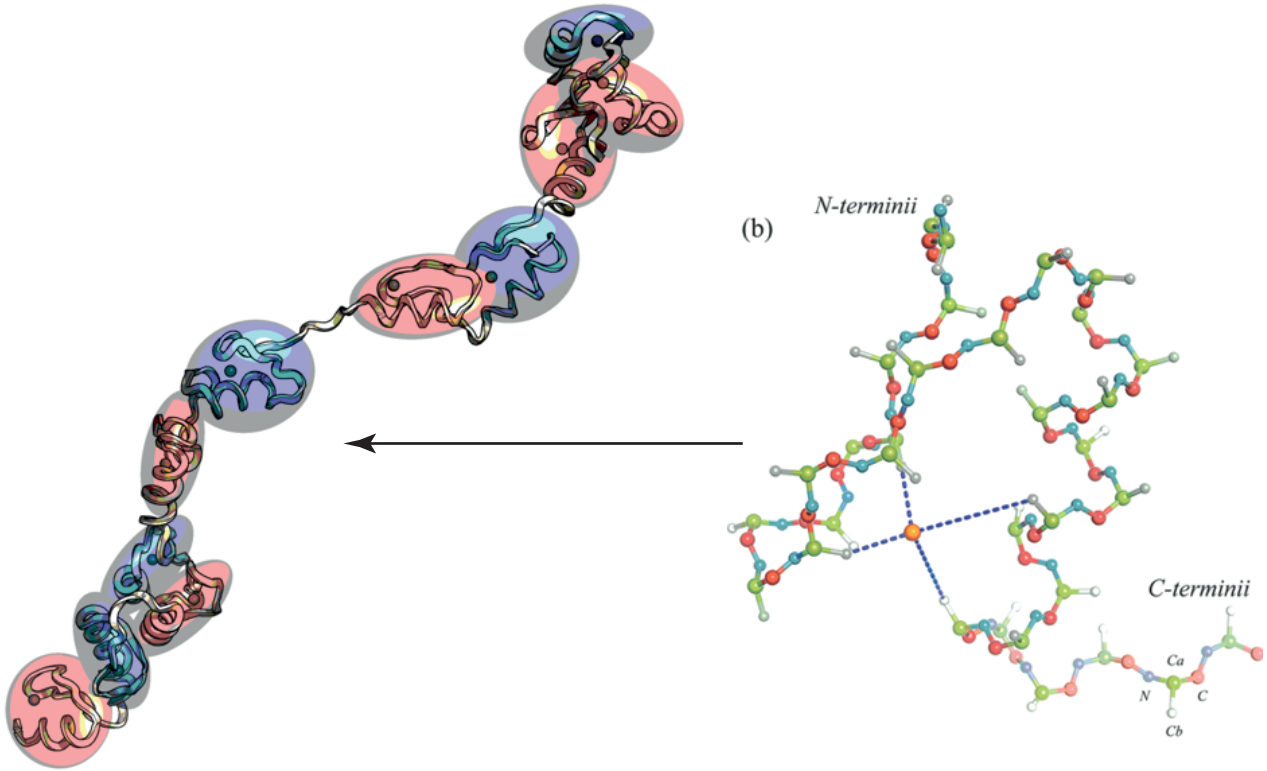


Figure: Shown is the model of a CTCF protein. Image taken from [13].

General Remarks IV

A molecular model requires typically

- definition of the degrees of freedom
- force fields
- boundary conditions

and a method to generate configuration or conformations (in the case of a macromolecule) as in the above two examples.
Typically for biological systems we are dealing with objects in Euclidean space $\mathbb{R}^3$. We will also be dealing with objects in more abstract spaces S.

- Let x denote a position in space which is derived from the degrees of freedom that are associated with space. Sometimes we will also use s or q. x can for example be the three-dimensional position $x = (x^1, x^2, x^3)^T$.
- Let p denote the dynamical variable (for example the momentum) $p = (p^1, p^2, p^3)^T$.

If we analyze the system in terms of dynamics then we need both the position and momentum of system in order to determine the future behavior of that system. Note that often position and momentum or in other circumstances we will use x to denote the **state of a system** which in the current case may comprise both position and momentum.
Recall that the

General Remarks V

- phase space $\mathcal{P}$ is the space of all possible states of a physical system.
- Let $t \in \mathbb{R}$ denote time, then a trajectory is a mapping

$$t \mapsto (q(t), p(t)) \tag{1}$$

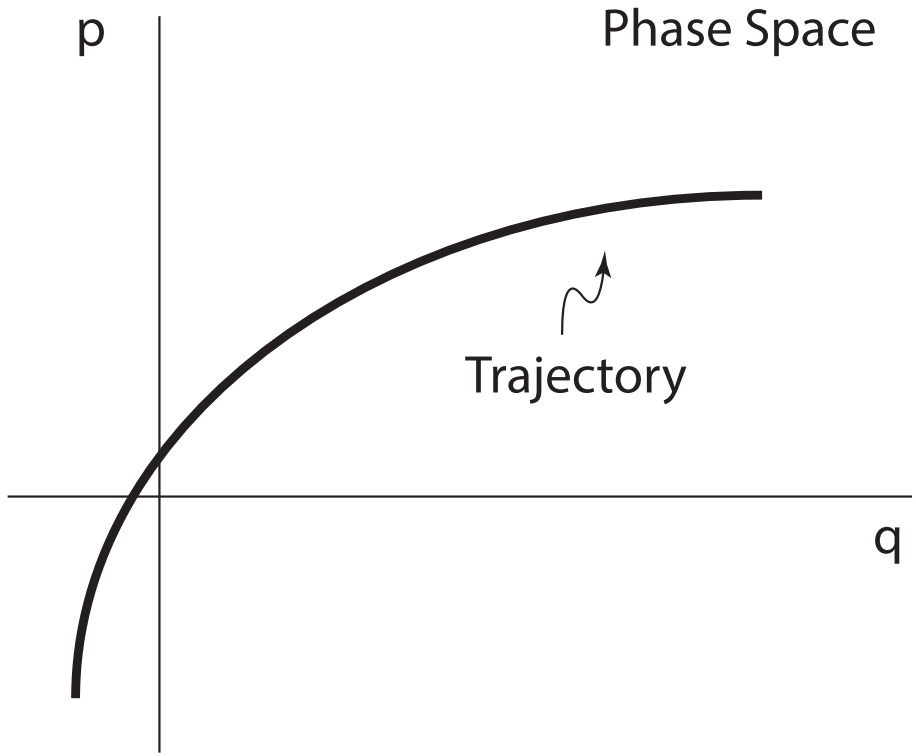


Figure: Phase space with generalized coordinate q and generalized momentum p.

General Remarks VI

More generally, we consider a state space $\mathbf{S}$ and a mapping T_t which transforms a state $s \in \mathbf{S}$ into a new state s' or $s(t)$

$$T_t \ : \ \mathbf{S} \to \mathbf{S} \tag{2}$$

The state space $\mathbf{S}$ or $\mathbf{X}$ may for example be $\mathbf{S} = S^{\mathbb{Z}^d}$ where $S = \{-1, 1\}$, or $S = \{0, 1\}, S = \{0, ..., q-1\}$.
This mapping may be a differential equation, as for example in the case of molecular dynamics, the discrete form of the differential equation, partial differential equation or stochastic processes.

Force Fields I

A central part of molecular dynamics, partially Monte Carlo, etc. is the computation of the force that is exerted on a particle or unit

$$F(r) = -\nabla U(r) \tag{3}$$

where U is the potential energy expanded into one-body, two-body, three-body etc. interactions

$$U(r) = U_0 + \sum_i U_1(r_i) + \sum_{i<j} U_2(r_i, r_j) + \sum_{i<j<k} U_3(r_i, r_j, r_k) + \cdots + U_N(r_i, r_j, \ldots, r_N), \tag{4}$$

One way of categorizing the basic potential energy is to partition it into intra- and and inter-molecular interaction. This distinction becomes relevant in the case of macromolecules like DNA, proteins etc.

Force Fields II

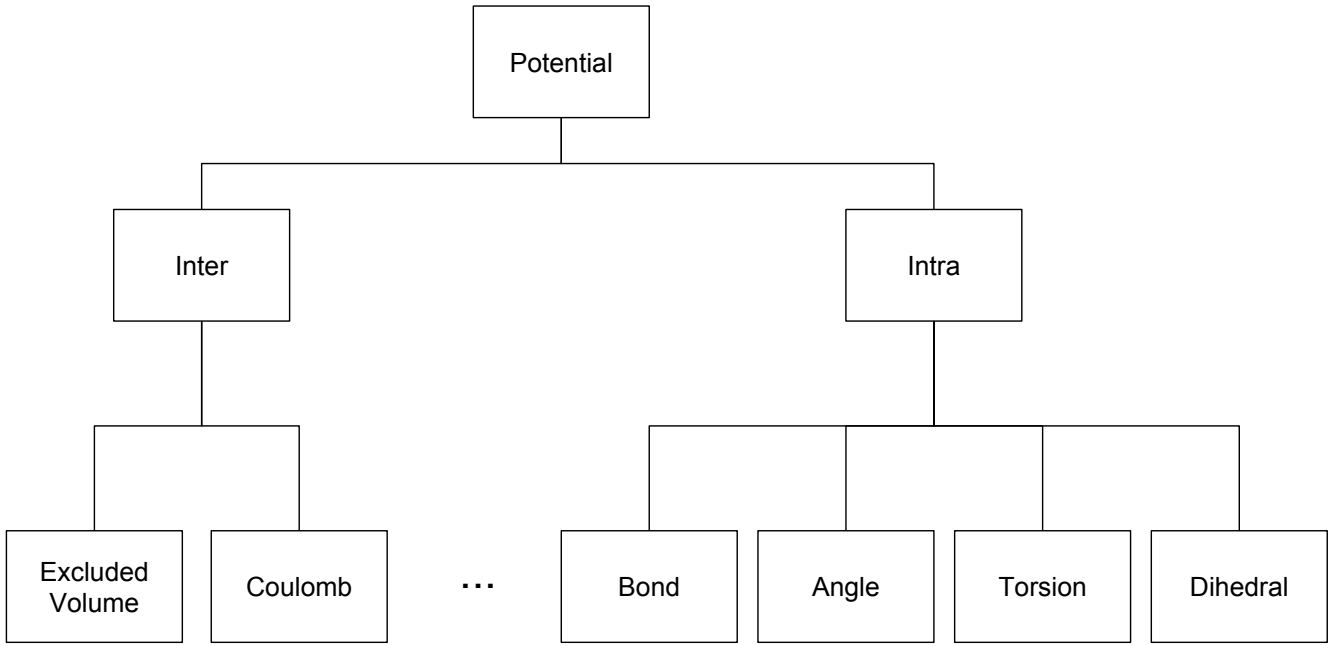


Figure: Shown is a possible classification scheme for potentials.

Force Fields III

- Bonded (Intra): This describes the covalent bonding between two atoms as: bond stretching and bending. Beyond the pair-potential in this class we also find the torsional potential.
  Assume a pair of nearest atoms (i, j) then (two-body potential)

$$U_{\mathrm{bond}}(r_{ij}) = \frac{1}{2} k_b (r_{ij} - l_0)^2 \tag{5}$$

  where $r_{ij} = ||r_i - r_j||$. Here l_0 is the equilibrium distance between i and j. k the is the spring constant or strength of the interaction.

Force Fields IV

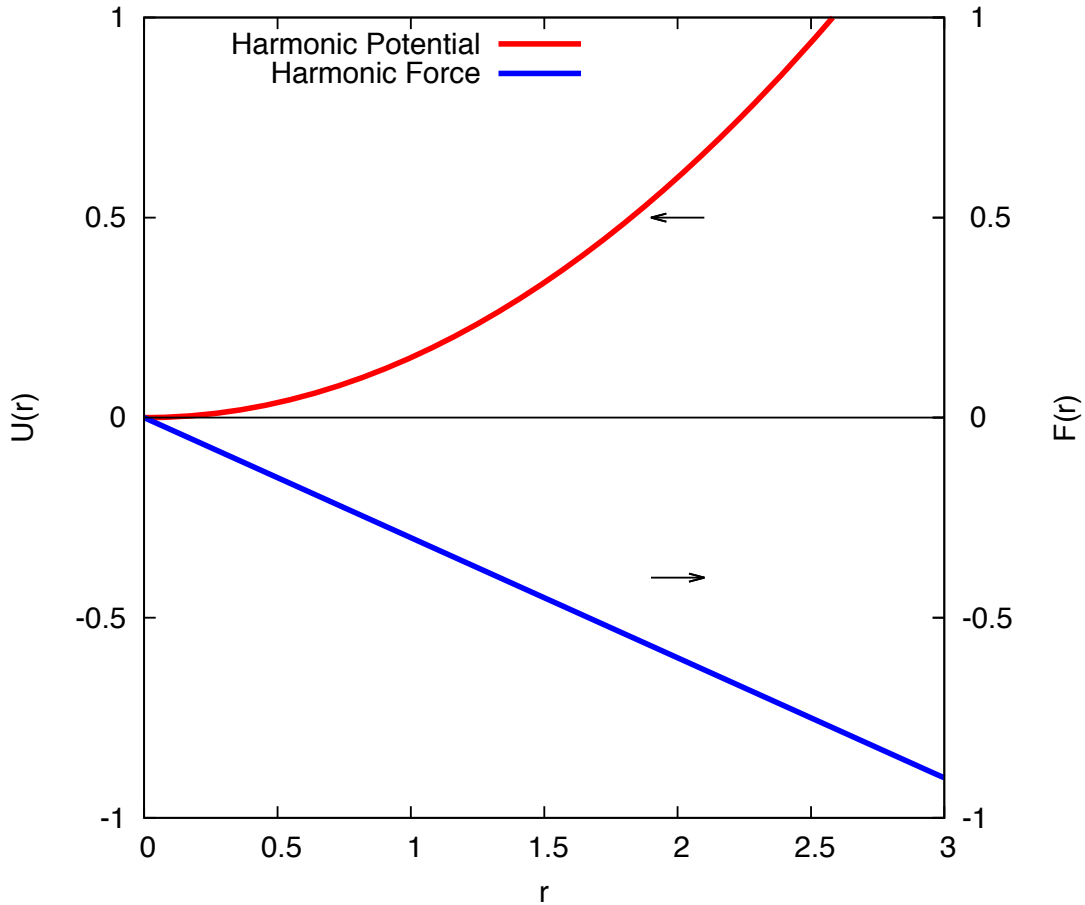


Figure: Harmonic potential as described in eq. 5 and the corresponding force.

The three-body interaction (i, j, k) describes the bond angle interaction

$$U_{\text{bond angle}}(\theta) = \frac{1}{2} k_\theta (\theta - \theta_0)^2 \tag{6}$$

where θ is the angle between the vectors $r_{ij} = r_j - r_i$ and $r_{jk} = r_k - r_j$. Sometimes this is augmented by an additional term that fixes a distance r_{ub} between the atoms (i, k) (Urey-Bradley term)

$$U_{\text{bond angle}}(\theta) = \frac{1}{2} k_\theta (\theta - \theta_0)^2 + \frac{1}{2} k_{ub} (r_{ik} - r_{\text{ub}})^2 \tag{7}$$

Then there is the four-body torsion or dihedral angle potential. Note that a given (i, j, k, l)-quadruple of atoms contributes multiple terms to the potential, each with its own set of parameterization

$$U_{\text{dihedral}} = \begin{cases} k(1 + \cos(n\psi + \phi)) & \text{if } n > 0, \\ (\psi + \phi)^2 & \text{if } n = 0 \end{cases} \tag{8}$$

The potential is between the planes formed by the first three and last three atoms of a consecutively bonded (i, j, k, l)-quadruple of atoms. ψ is the angle between the (i, j, k)-plane and the (j, k, l)-plane. ϕ is the phase shift angle.

- Non-bonded potential energy terms (Inter): Here we find van-der Waals, electrostatic, etc. These involve (in general) interactions between all (i, j)-pairs of atoms.

  The hard sphere potential describes the purely repulsive part of an interaction representing the excluded volume of the atom or particle

$$U_{\mathrm{HC}}(r_{ij}) = \begin{cases} \infty & r_{ij} \leq \sigma \\ 0 & r_{ij} > \sigma \end{cases} \tag{9}$$

  Here $r_{ij} = ||r_j - r_i||$ and σ is the excluded volume.

Force Fields VII

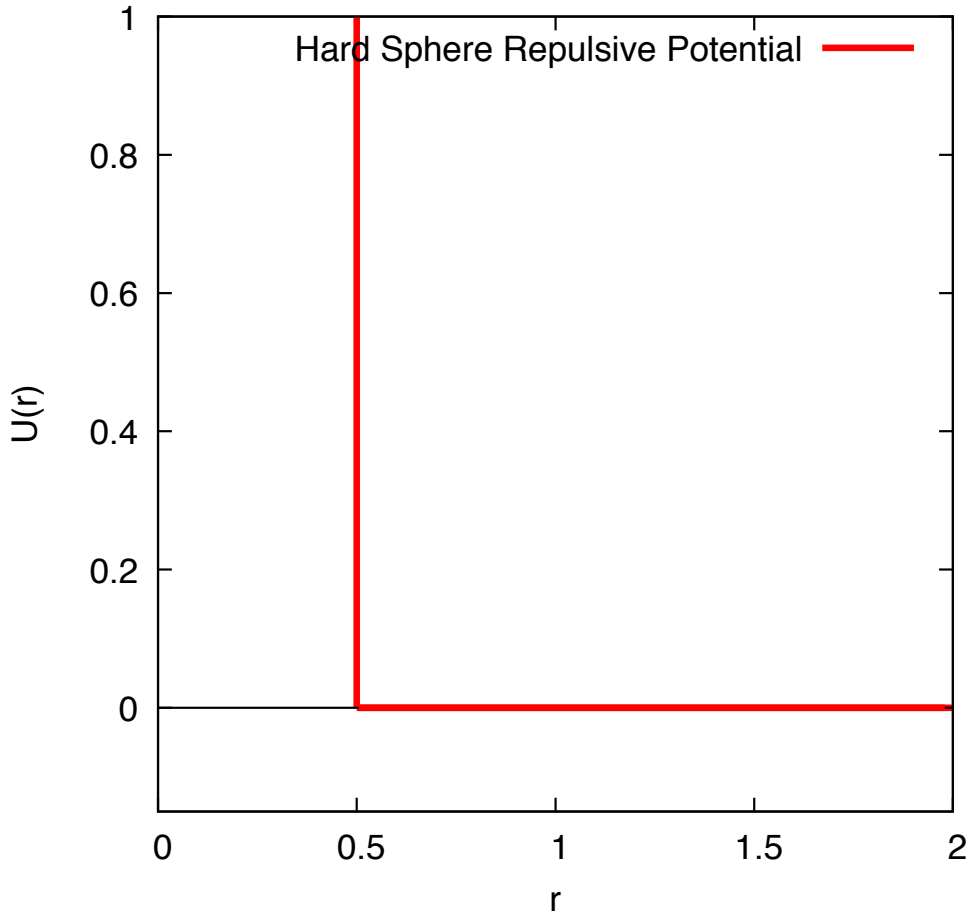


Figure: Excluded volume potential as described in eq. 9.

Another possible excluded volume interaction is given by the WCA (Weeks-Chandler-Andersen) potential [14], which was designed to model excluded volume interactions by a short-range repulsive force. The WCA potential is basically a truncated and shifted Lennard-Jones potential with the following functional form,

$$U_{\mathrm{WCA}}(r) = \begin{cases} 4\epsilon \left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} + c_{shift} \right) & r < r_{cut} \\ 0 & r \geq r_{cut} \end{cases} \tag{10}$$

Here $r_{cut} = 2^{1/6}$ and $c_{shift} = \frac{1}{4}$ are chosen such that the minimum of the potential is $U_{\mathrm{WCA}}(r_{min}) = 0$, the attractive part of the Lennard-Jones interaction being cut off. The WCA potential has two parameters ϵ and σ. σ defines the radius of the monomers' hard core. ϵ controls the energy penalty of another monomer penetrating this hard core.

Force Fields IX

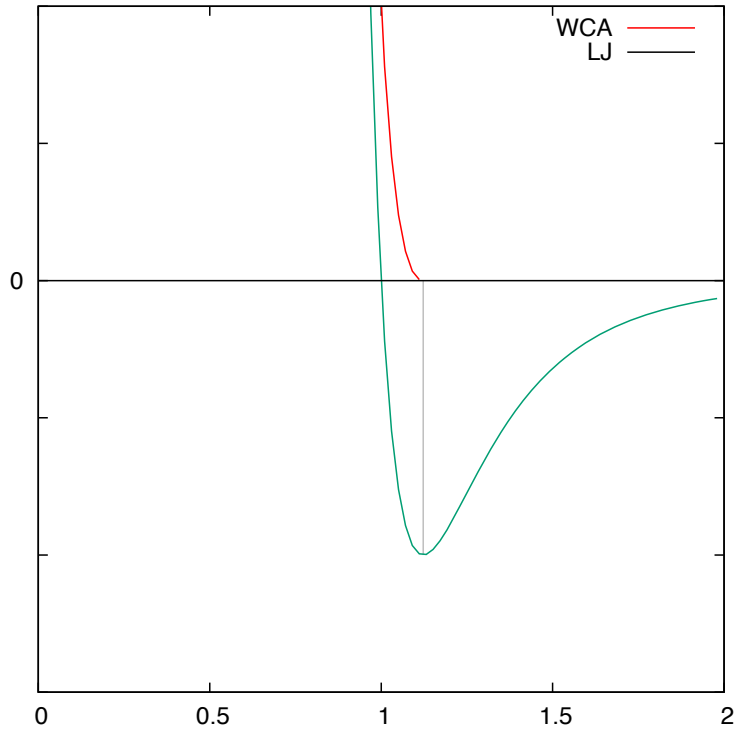


Figure: Weeks-Chandler-Andersen potential as well as the Lennard-Jones potential

The Lennard-Jones potential [15, 16] accounts for an excluded volume as well as for the weak dipole attraction between two atoms, i.e.due to instantaneous dipoles that arise during the fluctuations in the electron cloud

$$U_{\mathrm{LJ}}(r_{ij}) = 4\epsilon \left(\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right) \qquad r_{ij} \in [0, \infty] \tag{11}$$

Here $r_{ij} = ||r_j - r_i||$. ϵ is the energy parameter and σ sets the length scale, ie. the van der Waals radius of the atom (particle). Typical values are

atom	ϵ	σ
H	8.6	2.81
N	37.3	3.31
O	61.6	2.95

Table: Energy ϵ given in $1.38066 \times 10?23J$ and length σ given in $10^{-1}nm$

More generally we have

$$U_{\mathrm{LJ}}(r_{ij}) = 4\alpha\epsilon\left(\left(\frac{\sigma}{r_{ij}}\right)^{n} - \left(\frac{\sigma}{r_{ij}}\right)^{m}\right) \qquad r_{ij} \in [0, \infty] \tag{12}$$

with $n, m \in \mathbb{N}(n > m)$ and $\alpha = \frac{1}{n-m}\left(\frac{n^n}{m^m}\right)^{\frac{1}{n-m}}$. This potential is continuous and differentiable (C^∞)
Often the Lennard-Jones potential is truncated and shifted. For this the value at $r_c = 2.5\sigma$ is taken

$$U_{\mathrm{LJ}}(r_c) = 4\epsilon\left(\left(\frac{\sigma}{r_c}\right)^{12} - \left(\frac{\sigma}{r_c}\right)^{6}\right) \approx -0.0163\varepsilon \tag{13}$$

$$U_{\text{LJ-trunc}}(r) = \begin{cases} U_{\mathrm{LJ}}(r) - U_{\mathrm{LJ}}(r_c) & \text{if } r \leq r_c, \\ 0 & \text{if } r > r_c \end{cases} \tag{14}$$

This removes the discontinuity at r_c. The truncation though does effect the some properties like the location of phase transition points.

Force Fields XII

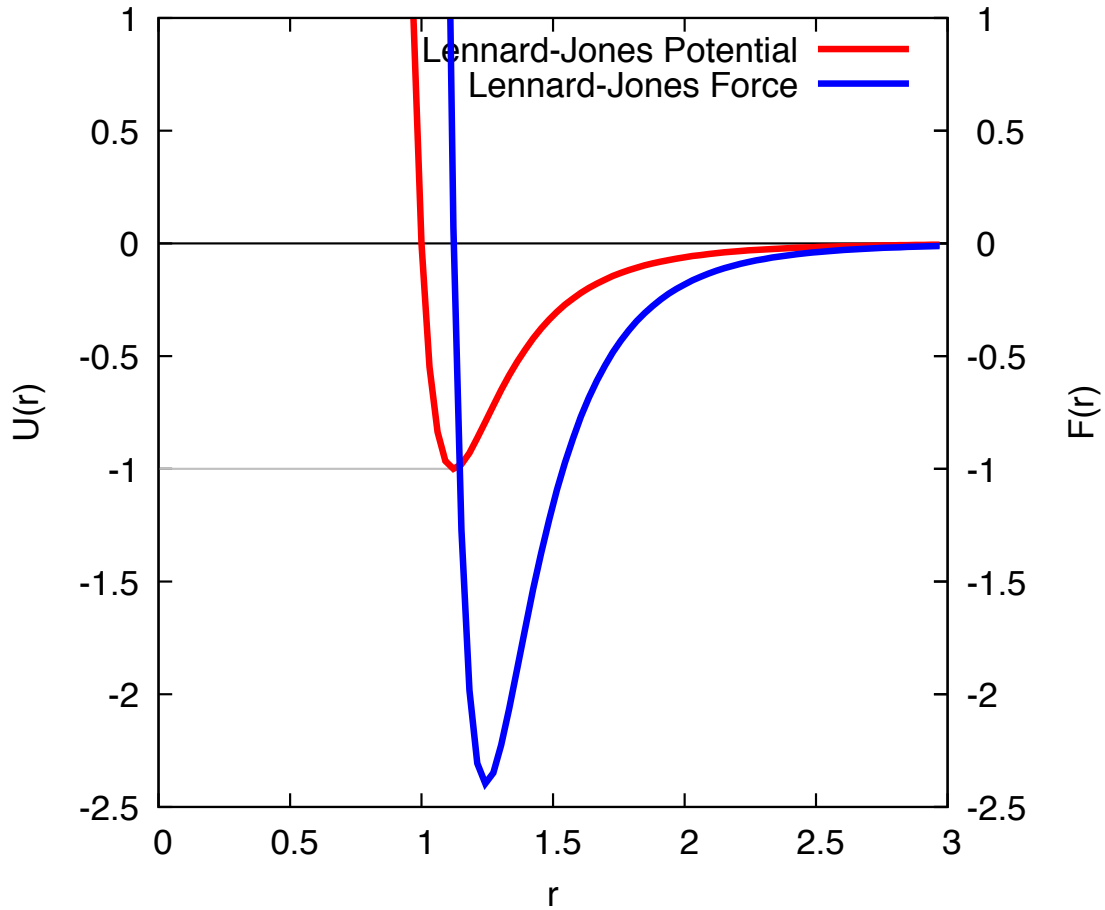


Figure: Lennard-Jones potential as described in eq. 12.

In the above we have assumed that all atoms (particles) are identical. For unlike pairs (ij) of atoms we need to define combination rules for the energy scale ϵ and the excluded volume σ:

$$\begin{aligned}
\epsilon_{ij} &= \sqrt{\epsilon_i \epsilon_j} & \sigma_{ij} &= \sqrt{\sigma_i \sigma_j} \\
\epsilon_{ij} &= \sqrt{\epsilon_i \epsilon_j} & \sigma_{ij} &= \frac{\sigma_i + \sigma_j}{2} \\
\epsilon_{ij} &= \frac{2\sigma_i^3 \sigma_j^3 \sqrt{\epsilon_i \epsilon_j}}{\sigma_i^6 + \sigma_j^6} & \sigma_{ij} &= \left(\frac{\sigma_i^6 + \sigma_j^6}{2} \right)^{1/6}
\end{aligned} \tag{15}$$

The electrostatic potential or Coulmb potential is repulsive for atomic charges q_1, q_2 with the same sign and attractive for atomic charges with opposite signs between two atoms (particles) (i, j).

$$U_C(r_{ij}) = \frac{q_1 q_2}{4\pi\epsilon_0} \frac{1}{r_{ij}} \qquad r_{ij} \in [0, \infty] \tag{16}$$

Here $r_{ij} = ||r_j - r_i||$.

Force Fields XV

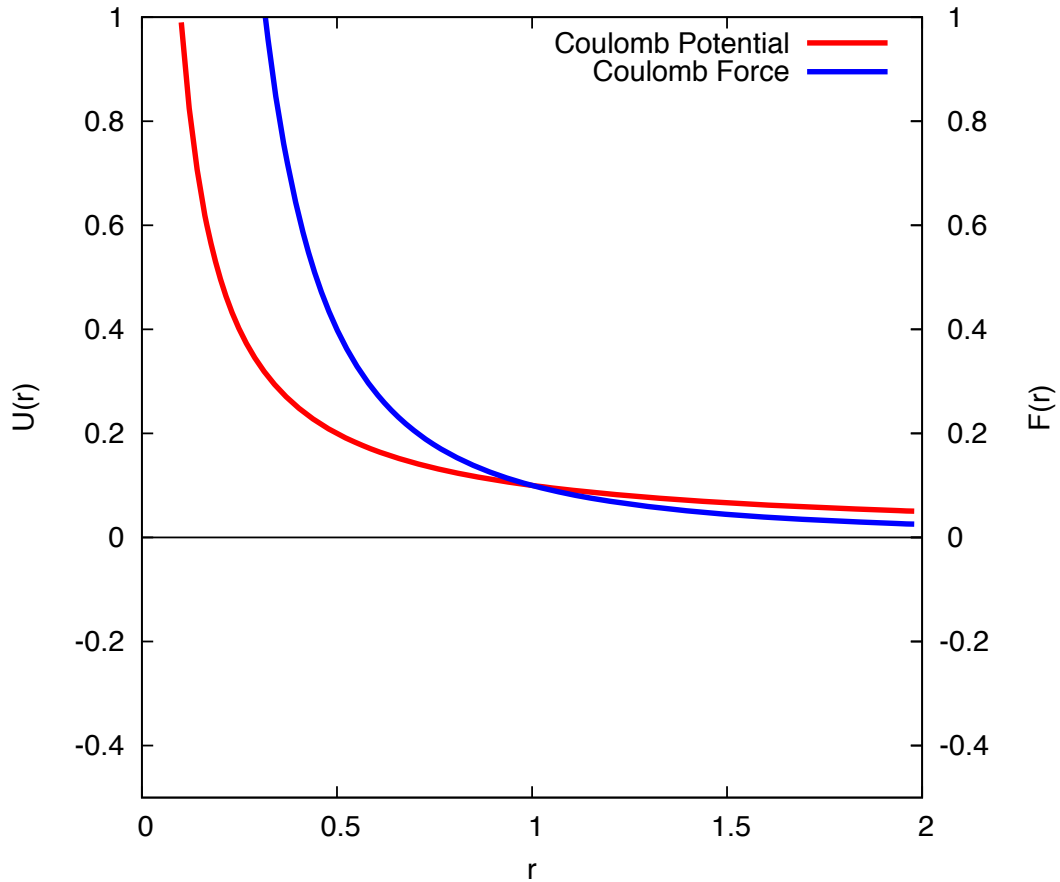


Figure: Lennard-Jones potential as described in eq. 16.

Force Fields XVI

The CHARMM force field [17] (here the CHARMM22 version) combines intra and inter atomic (molecular) potentials:

$$
\begin{aligned}
H \quad = \quad & \sum_{\text{bonds}} k_b \left(b - b_0\right)^2 + \sum_{\text{angles}} k_\theta \left(\theta - \theta_0\right)^2 + \sum_{\text{dihedrals}} k_\phi \left[1 + \cos\left(n\phi - \delta\right)\right] \\
& + \sum_{\text{impropers}} k_\omega \left(\omega - \omega_0\right)^2 + \sum_{\text{Urey}-\text{Bradley}} k_{ub} \left(r_{ik} - r_{ub}\right)^2 \\
& + \sum_{\text{nonbonded}} \epsilon \left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] + \frac{q_i q_j}{\epsilon r_{ij}}
\end{aligned}
\tag{17}
$$

Here $b = ||r_j - r_i||$ is the bond length. The non-bonded potentials are only applied to pairs separated by at least three bonds (intra-molecular potential) and to all other atoms (inter-molecular potential).

Force Fields: Empirical Potential I

Empirical potentials provide a way to specify for example the interactions in proteins. Specifically for proteins, the base for deriving potentials are protein databases. On the one hand these provide indeed through NMR or other experimental methods data on interatomic relations. Many structures are deposited daily. On the other hand, only those that have more or less fixed structure enter the data base. Proteins like the intrinsically disordered proteins (IDP's) do not, thus biasing the results. Hence implicit is the assumption that the set we are looking at represents the full spectrum of energies.
We will see below, that most of the time pair potentials are derived. This certainly is an assumption that may not apply. From solid state theory we know that certain crystal structures can not be obtained from pair potentials and require many body potentials.
Assume two atoms or particles A and B separated by a distance r. Then the probability P for finding them at distance r is given by

$$P_{AB}(r) = \frac{1}{Z} e^{-\beta U_{AB}(r)} \ . \tag{18}$$

where $\beta = 1/kT$ and Z is a normalizing constant

$$Z = \int_0^\infty e^{-\beta U_{AB}(r)} dr \ . \tag{19}$$

Force Fields: Empirical Potential II

Given a set of structures in a database like for example the protein database, a histogram of the distances that one finds in the database between A and B can be used to get an estimate of P_{AB}. Taking the logarithm of Eq 19 we obtain

$$\ln P_{AB} = -\beta U_{AB}(r) - \ln Z \ . \tag{20}$$

Since the constant $\ln Z$ is irrelevant we find the specific interaction potential. One can normalize this with the probability P_a for any pair

$$U_{AB}(r) = -kT \ln P_{AB} / \ln P_a(r) \ . \tag{21}$$

Clearly we need to ask, whether in the calculation of U_{AB} we include every occurrence of pairs AB or whether specific relations along the backbone of the protein need to be considered.
Also, we need to discretize the distances to ensure a sufficient statistics.

Force Fields: Empirical Potential III

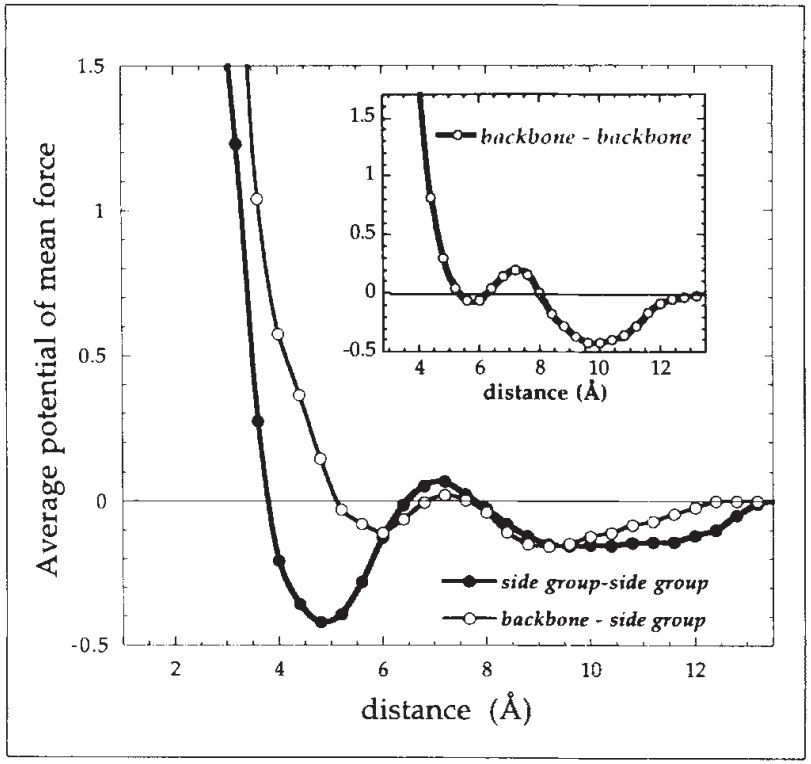


Figure: Empirical potential taken from *Structure-derived potentials and protein simulations* by Jernigan and Bahar [18]

Example

The H2AX protein (shown in Figure 12, Histone H2AX - P27661 (H2AX_MOUSE)) consists of 143 residues. It is a phosphorylated variant of histone H2A and associated with DNA damage. In this example the residues are mapped to a lattice model. To nevertheless capture specificity of each residue a residue-residue interaction and excluded volume constraints are implemented. Each residue interacts with the neighboring residues within a range of interaction using a generalized Lennard-Jones potential

$$U(r) = |\epsilon_{ij}| \left(\frac{\sigma}{r_{ij}}\right)^{12} + \epsilon_{ij} \left(\frac{\sigma}{r_{ij}}\right)^{6} \quad \text{for} \quad r_{ij} \leq r_c \tag{22}$$

where the ϵ_{ij}'s are parameterizations for Eq. 21 using X-ray crystallographic data [19] on a large number of protein structures from the protein data bank.

Force Fields: Empirical Potential V

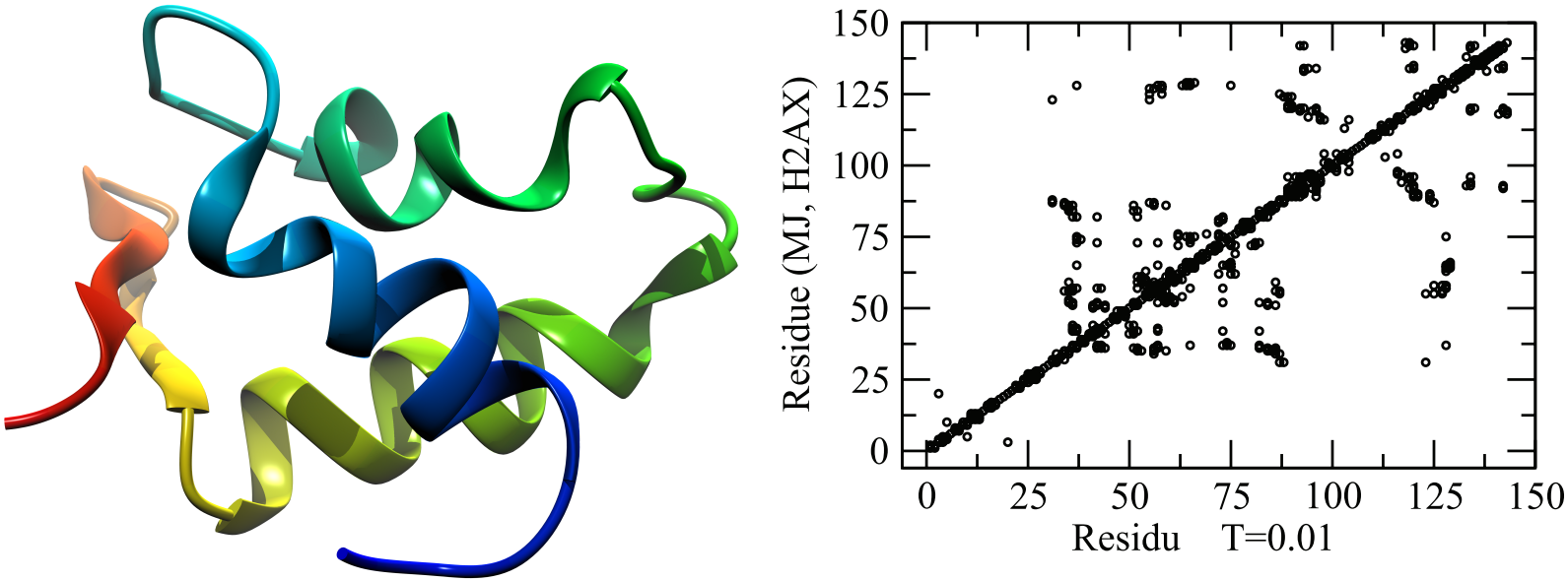


Figure: The left panel shows the ribbon images of H2AX. The right panel shows the contact probability between the residues at a specific temperature. Right image taken from Fritsche et. al. [20]

Force Fields: Non-Bonded Potential

Let's discuss the calculation of the potential which requires a summation over all unique pairs. The simplest algorithm doing so is to use a double loop. This is the most simplest realization of the **Nearest Neighbor Search** (NNS), i.e., find the k points that are closest to a given test point by the Euclidian distance metric. Because of the double loop the algorithm will scale with number of particles N as N^2. This is because for every particle we need to run through the entire list of particles to calculate the distance. If the potential is such that it is of finite range, then this is clearly not efficient. Later we will discuss more efficient algorithms.

```
1  /**
   * Force calculation
3  */
   void forceCalculation(float x[], float y[], float z[],
5                        float fx[],float fy[],float fz[],
                         float rcoffs,float *potential,
7                        float *virial, float side) {
     int i,j;
9    float xi,yi,zi;
     float rd, rd2, rd3, rd4, rd6, rd7;
11   float r148;
     float kx,ky,kz;
```

Force Fields: Non-Bonded Potential

```
  for (i = 0; i < 1; i++) {
    for (j = i+1; j < 3; j++) {
          xi = x[i] - x[j];
          yi = y[i] - y[j];
          zi = z[i] - z[j];
       minimumImage(&xi,&yi,&zi,side);
      rd = xi*xi + yi*yi + zi*zi;
      if (rd < rcoffs) {
        rd2  = rd * rd;
        rd3  = rd * rd2;
        rd4  = rd2 * rd2;
        rd6  = rd2 * rd4;
        rd7  = rd3 * rd4;
        *potential += ((1.0 /rd6)  - (1.0 / rd3));
        r148 = (1.0 / rd7) - (1.0 / rd4) * (float).5;
        *virial -= rd * r148;

        kx       = x[i] * r148;
        fx[i]    += kx;
        fx[j]    -= kx;

        ky       = y[i] * r148;
        fy[i]    += ky;
        fy[j]    -= ky;

        kz       = z[i] * r148;
        fz[i]    += kz;
        fz[j]    -= kz;
      }
    }
  }
}
```

Nearest-Neighbor Search I

Nearest-neighbor search, i.e., given a query point q, the task is to search for the k closest points to q among all points in a dataset.

- Linear search
- Space partitioning (data structure): The nearest neighbor search algorithms differ in their space-partitioning methods that split space using hyper-triangular or hyper-spherical bounding boxes, and build up a search tree on the resulting hierarchy of partitions. For the set of points, the root contains the entire search space, ie. all the points. This is then split into two partitions, depending on geometric shape. KD-trees and R-trees use rectangular shaped partitions. An alternative are metric-trees e.g. ball-tree [21, 22], in which the space partitioning is specifically optimized for the euclidean distance function.

Algorithm 1 Space Partitioning Algorithm

1: Compute the centroid or center of mass of the points.
2: Assign all points to the first partition
3: **while** points in partition **do**
4: Compute the farthest point from center of mass in the partition as the centroid of the first sub-partition
5: Select the farthest point from the first one as the centroid of the second sub-partition
6: **end while**

Note that this construction can lead to an unbalanced tree. The ideas listed below all use this basic algorithm differing in the implementation of the geometric shape of the partition, the space-partitioning strategy to build the search tree and the algorithm exploit the data structure.

- Euclidean Minimum Spanning Tree (EMST)
- Ball-tree [21, 22]: A ball-tree is an efficient metric-tree for euclidean distances. Each node in the ball-tree referred as ball contains a region of Euclidean points bounded by a hyper-sphere and interior balls are small containing their children balls. top down approach for building the tree recursively from top to down by choosing the split dimension and splitting value to find these values, balls are sorted along each dimension and store the cost in an array. Best dimension and split location can be found in $\mathcal{O}(n \log(n))$, so, time complexity to construct the ball-tree is $\mathcal{O}(n(\log n)^{\mathbf{2}})$.
- KD-tree [23]: A KD-tree is a generalization of binary trees, in which each internal node represents a rectangular partition of space and its subtree contains all data points that fall in the rectangle.
- quad-tree and oct-tree [24] Here the space is partitioned into four congurent squared and in the case of the oct-tree in three dimensions into eight cubes.

Nearest-Neighbor Search III

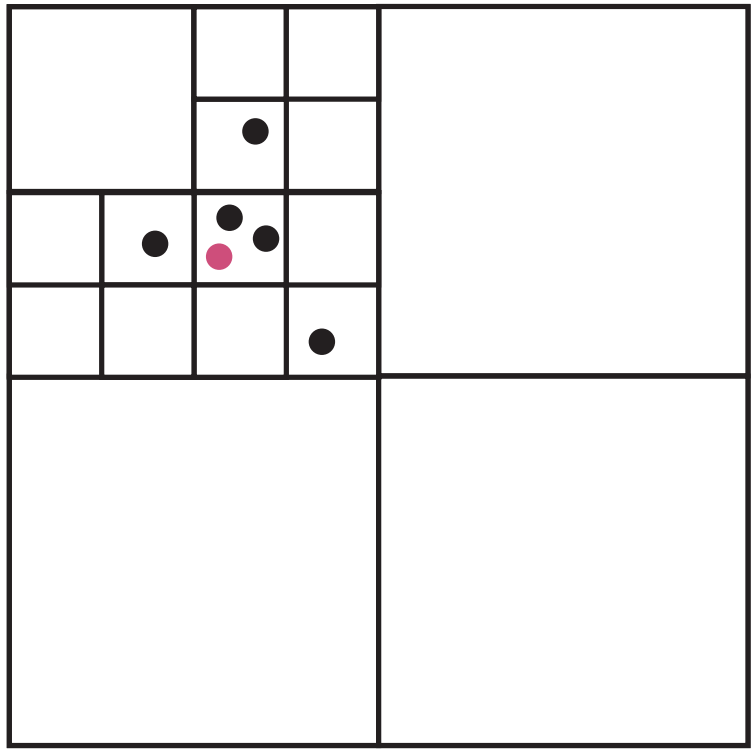


Figure: Example of a quad-tree space partitioning data structure. Shown is the space partitioning for the red particle, if only the black ones exist and that there is a finite cut-off of the potential consistent with the length scale of the last partitioning.

- R-tree [25] and R*-tree [26]
- Verlet table (Fixed-radius near neighbors)

- Ewald summation

The starting point for the Molecular Dynamics (MD) simulation [2, 3, 27–31] is thus a well-defined force field. Using Hamiltonian, Lagrangian or Newton's equations of motion these are approximated by suitable schemes Ψ such that they can be solved numerically.
We start the development with a Hamiltonian formulation with coordinates r and momenta p

$$H(r,p) = \frac{1}{2} p^T M^{-1} p + U(r) \quad . \tag{23}$$

Here M is the mass tensor. From this Hamiltonian we get the equations of motion

$$\frac{d}{dt} r = -\frac{\partial}{\partial r} H \tag{24}$$

$$\frac{d}{dt} p = \frac{\partial}{\partial p} H \quad . \tag{25}$$

An important consideration for any numerical integration scheme is that we want to conserve as many quantities during a numerical evaluation as possible that are conserved due to symmetries etc. This leads us to the concept of symplectic methods. Symplectic methods preserve certain abstract invariants of Hamiltonian systems [32–34] and are stable for linear systems for sufficiently small values of the step size.

We denote the trajectory that we want to generate by Γ

$$\Gamma(t) = \begin{pmatrix} r \\ p \end{pmatrix} \quad . \tag{26}$$

Then this trajectory obeys the equation of motion

$$\frac{d}{dt}\Gamma = J\nabla H(\Gamma) \tag{27}$$

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \quad , \tag{28}$$

where I denotes the unit matrix.
We compute an observable A along the trajectories and hence average along the states we find along the path

$$\langle A \rangle = \frac{1}{n_{\text{obs}}} \sum_{\nu=1}^{n_{\text{obs}}} A(\Gamma_\nu(t)) \quad . \tag{29}$$

Here n_{obs} is the number observations we took, i.e., how many iterations we took in the numerical integration of the equations of motion.

Let $\rho_0(\Gamma)$ denote the probability density at time $t = 0$: $\Gamma(0) = \rho_0(\Gamma)$ and let $\rho(\Gamma, t)$ denote the probability density for $\Gamma(t)$. Then we have the Liouville theorem for the trajectories.

$$\begin{aligned} \rho(\Gamma, 0) &= \rho_0 && (30) \\ \rho_t + \nabla \cdot (\rho J \nabla H) &= 0 && (31) \end{aligned}$$

This states that the flow in phase space is that of an incompressible fluid.
If $\rho_t = 0$, then

$$\nabla H \cdot J \cdot \nabla \rho = 0 \tag{32}$$

and with this

$$\rho(\Gamma) = \frac{e^{-H(\Gamma)/k_B T}}{\int e^{-H(\Gamma)/k_B T} d\Gamma} \quad . \tag{33}$$

Now let Ψ be a numerical integrator, i.e. a discretization of the equations of motion

$$\Gamma^{n+1} = \Psi(\Gamma^n) \quad , \tag{34}$$

then the phase space volume needs to be conserved

$$\det \partial_\Gamma \Psi(\Gamma) = 1 \quad . \tag{35}$$

The integrator is sympletic, if

$$(\partial_\Gamma \Psi(\Gamma)^T J(\partial_\Gamma \Psi(\Gamma) = J \tag{36}$$

i.e. phase space volume and the energy is conserved.

Molecular Dynamics: Discretization I

The most straightforward discretization of the equations of motion that involve differentials comes from the Taylor expansion. The idea is to base the algorithm on a discrete version of the differential operator. With suitable assumptions we can expand the variable r in a Taylor series

$$r(t+\Delta t) = r(t) + \sum_{i=1}^{n-1} \frac{\Delta^i t}{i!} r^{(i)}(t) + R_n \quad . \tag{37}$$

where R_n gives the error involved in the approximation.
Using the forward $t + \Delta t$ and the backward difference $t - \Delta t$

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{F(t)}{2m}\Delta^2 t + \frac{d^3 r}{dt^3}\frac{\Delta^3 t}{3!} + R_4 \tag{38}$$

$$r(t-\Delta t) = r(t) - v(t)\Delta t + \frac{F(t)}{2m}\Delta^2 t - \frac{d^3 r}{dt^3}\frac{\Delta^3 t}{3!} + R_4 \tag{39}$$

$$\tag{40}$$

If we add the two equations, we obtain

$$r(t+\Delta t) = 2r(t) - r(t-\Delta t) + \frac{F(t)}{2m}\Delta^2 t + R_4^* \tag{41}$$

If we subtract the two equations we obtain

$$r(t + \Delta t) + r(t - \Delta t) = 2v(t)\Delta t + R_3^* \tag{42}$$

and hence an estimator for the velocity

$$v(t) = \frac{r(t + \Delta t) + r(t - \Delta t)}{2\Delta t} + R_2^* \quad . \tag{43}$$

The estimator for the position and the velocity together comprise what is known as the **Verlet algorithm** [31]. The Verlet algorithm is a second-order method that is indeed symplectic.
So, much hinges on the simulation step-size, since this determines the time-scales, that we can cover. As we have seen above the choice of step size is dominated by stability demands and not by accuracy demands.

Molecular Dynamics: Basic Algorithm

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Algorithm 2 Molecular Dynamics Algorithm

1: $n = 0$
2: Specify positions r_i^{-1} and r_i^0
3: Set Δt
4: **while** $n \neq maxSteps$ **do**
5: Compute the forces at time step n: F_i^n
6: Compute the positions at time step $n+1$ r_i^{n+1}
7: Compute the velocities at time step n: v_i^n
8: $n = n + 1$
9: **end while**

Note that in this formulation of the molecular dynamics algorithm we need two starting positions, rather than position and velocity!

Molecular Dynamics: Boundary Conditions I

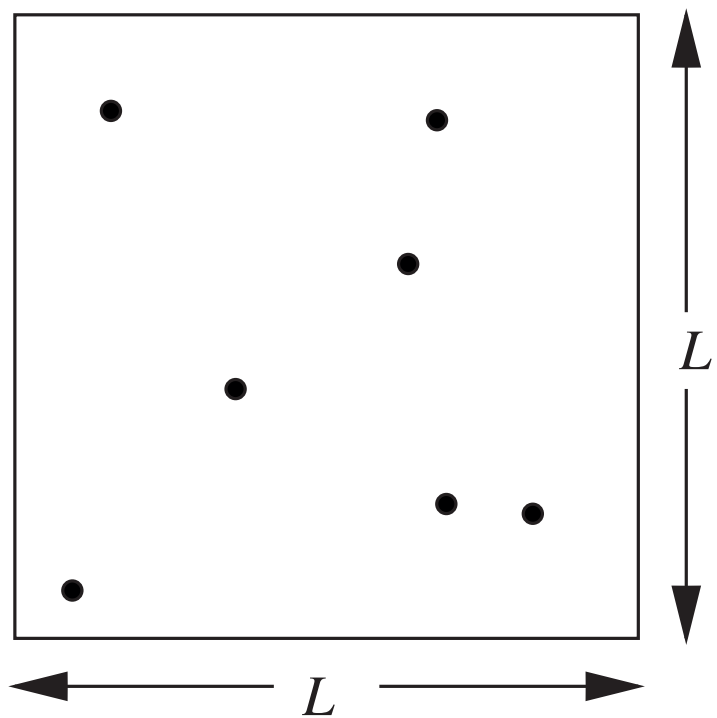
The computational volume is given by

$$\Omega = (a_1, b_1) \times \ldots \times (a_d, b_d) \subset \mathbb{R}^d \tag{44}$$

where here for simplicity we assume

$$b_i - a_i = L, \quad a_i, b_i \in \mathbb{R} \tag{45}$$

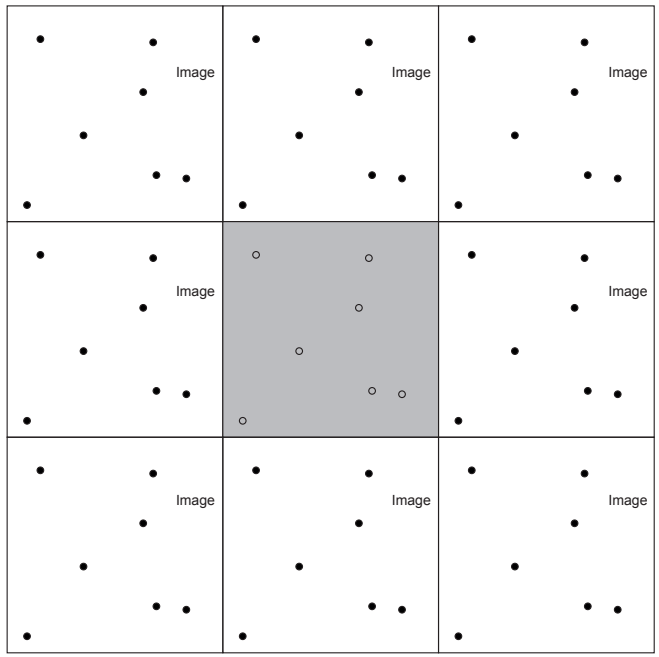
Of course, more complicated situations may be considered.



Molecular Dynamics: Boundary Conditions

$$F : \overline{\Omega} \to \mathbf{R}$$

$$\forall r = (x_1, \ldots, x_d) \in \Omega, i = 1, \ldots, d :$$
$$F(x_1, \ldots, x_{i-1}, a_i, x_{i+1}, \ldots, x_d) = F(x_1, \ldots, x_{i-1}, b_i, x_{i+1}, \ldots, x_d) \quad (46)$$



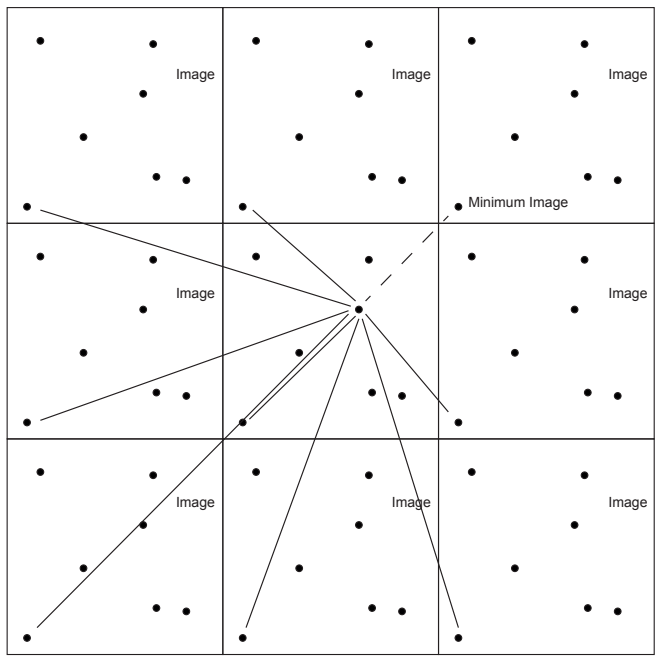
Molecular Dynamics: Boundary Conditions


```
/**
 * Periodic boundary conditions
 */
void applyPeriodicBoundary(float x[], float y[], float z[],float side) {
    for (int n = 0; n < N; n++) {
        if ( x[n] < 0 )    x[n] += side;
        if ( x[n] > side ) x[n] -= side;
        if ( y[n] < 0 )    y[n] += side;
        if ( y[n] > side ) y[n] -= side;
        if ( z[n] < 0 )    z[n] += side;
        if ( z[n] > side ) z[n] -= side;
    }
}
```

Molecular Dynamics: Boundary Condition

Algorithm 3 Minimum Image Criterion (here for $d = 1$)

1: $dx = x_j - x_i$
2: **if** dx > L * 0.5) **then**
3: dx = dx - L
4: **end if**
5: **if** dx <= -L * 0.5 **then**
6: dx = dx + L
7: **end if**



Molecular Dynamics: Boundary Condition


```
/**
 * Minimum image convention
 */
void minimumImage(float* xi, float* yi, float* zi,float side){
  float sideh;
  sideh = side * 0.5;
  if (*xi < -sideh) { *xi += side;}
  if (*xi >  sideh) { *xi -= side;}
  if (*yi < -sideh) { *yi += side;}
  if (*yi >  sideh) { *yi -= side;}
  if (*zi < -sideh) { *zi += side;}
  if (*zi >  sideh) { *zi -= side;}
}
```

Molecular Dynamics: Force Calculation I

- TBD

Molecular Dynamics: Verlet Algorithm

The Verlet algorithm can be reformulated in such a way as to give a numerically more stable method [36, 37]. Define

$$z_i^n = \frac{r_i^{n+1} - r_i^n}{h} \tag{47}$$

The equations

$$\begin{aligned} r_i^n &= r_i^{n-1} + h z_i^{n-1} \\ z_i^n &= z_i^{n-1} + h F_i^n / m \end{aligned} \tag{48}$$

are called the summed form. A further reformulation yields the velocity form of the Verlet algorithm.

Algorithm 4 Molecular Dynamics Algorithm: NVE Velocity Form

1: $n = 1$
2: Set Δt
3: Specify the initial positions r_i^0, r_i^1
4: Specify the initial velocities v_i^0, v_i^1
5: Compute the forces F_i^1
6: **while** $n \neq maxSteps$ **do**
7: $\quad r_i^{n+1} = 2r_i^n - r_i^{n-1} + F_i^n h^2/m$
8: $\quad$ Compute F_i^{n+1}
9: $\quad v_i^{n+1} = v_i^n + h(F_i^{n+1} + F_i^n)/m$
10: $\quad n = n + 1$
11: **end while**

Molecular Dynamics: Example I

TBD

Molecular Dynamics: Constant Temperature I

One way of achieving energy fluctuations for a constant temperature is to supplement the equations of motion with an equation of constraint. Alternatively one can add to the forces in the equations of motion a force of constraint (damped-force method) [38–43]. It can be shown [44] that the damped-force method is a special case of the constraint method. Another possibility is that of immersing the system in a heat bath by introducing a stochastic force simulating collisions with virtual particles. Later we take up the idea of stochastic supplements to the equations of motion. A natural choice for the constraint is to fix the kinetic energy to a given value during the course of a simulation. Such a constraint may be the non-holonomic constraint [44]

$$\Lambda = \frac{1}{2}\sum_i m v_i^2 = const \tag{49}$$

(isokinetic MD) or one may take the total kinetic energy proportional to time with a vanishing proportionality constant if the system has reached a constant temperature (Gaussian isokinetic MD) [45]

$$\frac{1}{2}\sum_i m v_i^2 = \alpha t \tag{50}$$

$$\beta = \left[(3N-4)k_B T_{\mathrm{ref}} / \sum_i m v_i^2\right]^{1/2} \tag{51}$$

so that after the scaling step we have

$$\frac{1}{2}\sum_i m v_i^2 = \frac{1}{2}(3N-4)k_B T_{\text{ref}} \quad . \tag{52}$$

Algorithm 5 Molecular Dynamics Algorithm: NVT

1: $n = 1$
2: Set Δt
3: Specify the initial positions r_i^1
4: Specify the initial velocities v_i^1
5: Compute the forces F_i^1
6: **while** $n \neq maxSteps$ **do**
7: $\quad r_i^{n+1} = r_i^n + hv_i^n + h^2 F_i^n / 2m$
8: $\quad v_i^{n+1} = v_i^n + h(F_i^{n+1} + F_i^n)/2m$
9: $\quad$ Compute $\sum_i (v_i^{n+1})^2$ and the scaling factor β.
10: $\quad$ Scale all velocities $v_i^{n+1} \rightarrow v_i^{n+1}\beta$
11: $\quad n = n + 1$
12: **end while**

Molecular Dynamics: Constant Pressure I

For the isolated N-particle system the energy E and the volume V are the independent variables. If we fix the pressure, then the volume, being the variable conjugate to the pressure, must be allowed to fluctuate. For a constant particle number N and constant pressure P, the enthalpy H is conserved

$$H = E + PV \tag{53}$$

To make the volume fluctuations possible we introduce the volume V as a new dynamical variable. As such it is also assigned a mass M. To develop equations of motion for the particles and the volume we further take PV as the potential energy corresponding to the new dynamic variable [46–48]. The Lagrangian now looks like

$$\mathcal{L}(r, \dot{r}, V, \dot{V}) = \frac{1}{2}\sum_i m\dot{r}_i^2 - U(r) + \frac{1}{2}M\dot{V} + P_E V \tag{54}$$

Of course, the variables $\mathbf{r}$ and V are not coupled. To proceed we appeal to intuition. If the system is subjected to pressure, the distances between the particles will change. Conversely, if the distances change, the pressure changes. The crucial step is the replacement of the coordinates $\mathbf{r_i}$ of the particles by the scaled coordinates $\rho_\mathbf{i}$ i.e.,

$$\rho_i = \frac{r_i}{V^{1/3}} = \frac{r_i}{L} \tag{55}$$

Now all components of the position vectors of the particles are dimensionless numbers within the unit interval $[0, 1]$. With the transformation, the integrals of $\mathbf{r_i}$ over the

fluctuating volume V become integrals of $\rho_{\mathbf{i}}$ over the unit cube. Having written down (55) we have made the implicit assumption that each spatial point responds in the same way. Due to this, there is no consistent physical interpretation of the approach. The equation (55) couples the dynamical variables $\mathbf{r}$ to the volume. Taking the first time derivative we obtain

$$\dot{r}_i = L\dot{\rho}_i + \rho\dot{L} \tag{56}$$

In equilibrium, the changes in the volume can be regarded as slow. Therefore we may assume

$$\frac{p_i}{m} = L\dot{\rho}_i \tag{57}$$

as the momentum conjugate to ρ_i and the Lagrangian becomes

$$\mathcal{L}(\rho, \dot{\rho}, V, \dot{V}) = \frac{1}{2}L^2 \sum_i m\dot{\rho}_i^2 - U(L\rho) + \frac{1}{2}M\dot{V}^2 - P_E V \tag{58}$$

Recall that we anticipate possible effects on the intrinsic dynamics when we modify the equations. However, the static properties should not be affected. Concerning this point, note that the potential energy does not involve the new coordinates ρ but the true r. In a somewhat loose way the Hamiltonian of the system is formulated as [49, 50]

$$\mathcal{H} = \frac{1}{2}L^2 \sum_i m\dot{\rho}_i^2 + U(L\rho) + \frac{1}{2}M\dot{V}^2 + P_E V \tag{59}$$

Here M is still a free parameter, about which we will have more to say later. Having set up the Hamiltonian the next task is to derive the equations of motion for the particles and the volume. These equations will now be coupled. In the Newtonian formulation they are

$$\begin{aligned} \frac{d^2\dot{\rho}_i}{dt^2} &= \frac{F_i}{mL} - \frac{2}{3}\dot{\rho}_i\left(\frac{\dot{V}}{V}\right), \\ \frac{d^2V}{dt^2} &= \frac{P - P_E}{M} \end{aligned} \tag{60}$$

with the pressure P computed from the virial

$$P = \frac{1}{3L}\left[\sum_i m\dot{\rho}_i^2 + \sum_{i<j} r_{ij}F_{ij}\right] \tag{61}$$

Algorithm 6 Molecular Dynamics Algorithm: NVT

1: $n = 1$
2: Set Δt
3: Specify the initial positions r_i^1
4: Specify the initial velocities v_i^1
5: Specify an initial volume V^0 consistent with the required density
6: Specify an initial velocity for the volume
7: Compute the forces F_i^1
8: **while** $n \neq maxSteps$ **do**
9: Compute $\rho^{\mathbf{n+1}}$ and V^{n+1}
10: $r_i^{n+1} = r_i^n + hv_i^n + h^2 F_i^n / 2m$
11: $v_i^{n+1} = v_i^n + h(F_i^{n+1} + F_i^n)/2m$
12: Compute the pressure P^{n+1}
13: $n = n + 1$
14: **end while**

Langevin Dynamics

Another approach to understand how structure arises is to model part of the system chemically as realistic as possible and take the interaction of atoms with the environment into account only through their stochastic influence. For example, the water molecules and all other possible solvent molecules are not explicitly taken into account.

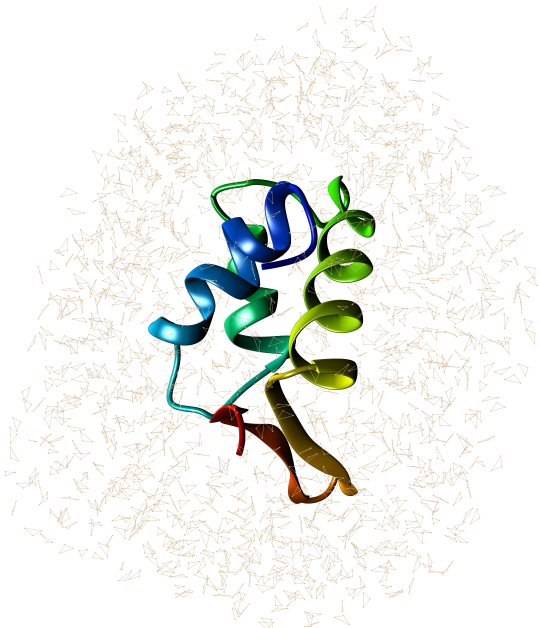


Figure: H2AX protein with solvants

Langevin Dynamics I

Thus we start off with a Langevin equation [51]

$$M\ddot{r} + \eta\dot{r} + \nabla V = \xi(t) \quad , \tag{62}$$

where M is the mass matrix, η the damping matrix and ξ a normalized white noise resulting from a Wiener process.
Because of the independence of the coordinates, the Langevin equation (62) only depends on the covariance matrix of the noise $\xi(t) = D\dot{W}(t)$ (here D is the diffusion matrix and W a Wiener process)

$$DD^T = \langle \xi\xi^T dt \rangle \quad . \tag{63}$$

The fluctuation-dissipation theorem relates the diffusion matrix to the damping matrix and the temperature

$$DD^T = 2k_B T\eta \quad . \tag{64}$$

Because of the lack of knowledge of the detailed damping a further reduction in complexity is usual taken by setting the damping matrix proportional to the mass matrix

$$\eta = \gamma M \tag{65}$$

with a damping constant γ. We then get

$$D = (2k_B T\gamma M)^{1/2} \quad . \tag{66}$$

Let ΔE be the energy barrier which the molecule has to take to get to a new state. Then the activation energy can be related to the mean frequency of transition f by an **Arrhenius law**, i.e., the rate increases exponentially with the absolute temperature

$$f = \frac{k_B T}{h} \exp\left(-\frac{\Delta E}{k_B T}\right) \tag{67}$$

where h is the Plank constant.
We shall now assume that we are at low temperatures, where the motions of the involved atoms are small. The protein is further assumed in a local minimum x_{min} and we are interested in the high frequency modes. The highest frequencies are of the order of 10^{14} sec (roughly the C-H bond vibrations).
For simplicity we shall also assume that the eigenmodes are non-degenerate. Then we can expand the potential up to second order

$$V(r) \approx V(r_{min}) + \frac{1}{2}(r - r_{min})^T \bar{V}(r - r_{min}) \tag{68}$$

where $\bar{V} = \nabla^2 V(r_{min})$. Ignoring in the limit of low temperature and high frequency the damping and the random force term we obtain

Langevin Dynamics III

$$M\ddot{r} + \bar{V}(r - r_{min}) = 0 \tag{69}$$

with the general solution

$$r = r_{min} + \sum_{l} u_l \exp(i\omega_l t) \tag{70}$$

Here ω_l are the frequencies and u_l are the normal modes.

Monte Carlo Method I

What are Monte Carlo Methods?

- In the widest sense of the term, Monte Carlo (MC) simulations mean any simulation (not even necessarily a computer simulation) that utilizes random numbers in the simulation algorithm.
- Monte Carlo simulations are statistical and non-deterministic. Hence each simulation will give a different result, but the results will be related via some statistical error.
- The Monte Carlo algorithm was named the top algorithm of the 20th century by mathematicians and physicists.

Why do we need this?

- Multidimensional integrals
- Systems with a large number of degrees of freedom
  - Many atoms in a gas, liquid, solid
  - Many electrons in an atom
  - Gene expression
  - Networks
  - ...

Typical Parts of a Monte Carlo Method

- Probability Distribution Functions
- Random Number Generator
- Sampling Rule
- Evaluating
- Error Estimation

Bayes Theorem

- Let A and B be events

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\bar{A})P(\bar{A})} \tag{71}$$

$$= \frac{P(B|A)P(A)}{P(B)} \tag{72}$$

Random Numbers I

> *Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.*
> *John Von Neumann, 1951*

Intuitively, we can list a number of criteria that a sequence of numbers must fulfill to pass as a random number sequence:

- unpredictability,
- independence,
- without pattern.

These criteria appear to be the minimum request for an algorithm to produce random numbers. More precisely we can formulate:

- uniform distribution,
- uncorrelated,
- passes every test of randomness,
- large period before the sequence repeats (see later),
- sequence repeatable and possibility to vary starting values,
- fast algorithm.

Random Numbers II

The most common generators use very basic operations and apply them repeatedly on the numbers generated in previous steps. We formulate this as a recursion relation

$$x_{i+1} = G(x_i), \qquad x_0 = \text{initial value} \quad , \tag{73}$$

where we have made explicit only the dependence on the immediate predecessor. The most important representatives of this class of generators are the

- linear congruential,
- lagged Fibonacci,
- shift-register or a
- combination of linear congruential.

Linear Congruential Generators
A very simple generator is constructed using the modulo function.

$$G(x) = (xa + b) \bmod M \tag{74}$$

This function produces a dilatation, translation and a folding back into the interval. Random number generators based on this function are called **linear congruential generators** or LCG(a,b,M) for short. If we assume integers as the set on which the modulo function is defined, then for example, the range of integer numbers for a 32-bit architecture is at most $M = 2^{31} - 1$. Here we assume that one bit is taken for the sign of the number. Then the numbers range at most from 0 to $M - 1$. Of course, we can map these onto the real interval between 0 and 1, recognizing that this is an approximation to the real-valued random numbers.

Inverse Congruential Generators
A very simple generator is constructed using the modulo function.

$$x_{n+1} = (x_n^{-1} a + b) \bmod M \quad , \tag{75}$$

where x_n^{-1} is the multiplicative inverse of x_n in the integers mod m with 0^{-1} defined as 0.

- The choice of the parameters a, b and M determine the statistical properties and how many different numbers we can expect before the sequence repeats itself.

Random Numbers IV

- The period can be shown to be maximal, if M is chosen to be a prime number. Then the whole range of numbers occurs.
- Here we only consider modulo generators with $b = 0$.
- Such generators are called **multiplicative** and the short form MLCG(a,M) is used for such generators.
- These are the most commonly used, since one can show that **additive** generators, i.e. generators with b in general non zero have undesirable statistical properties.
- The choices for the parameter a are manifold. For example $a = 16807$, 630360016 or 397204094 are possible choices with $M = 2^{31} - 1$.

Random Numbers V

```
/*-------------------------------------------------------------------------- */
/*                          Modulo Generator                                 */
/*-------------------------------------------------------------------------- */
int ModGenerator(modul,multi,inc,seed,max_sweeps,x)
    int             modul;
    int             multi;
    int             inc;
    int             seed;
    int             max_sweeps;
    float           *x;
{
    /*-------------------------------------------------------- */
    /*                      Declarations                       */
    /*-------------------------------------------------------- */
    int i;
    double r;
    double factor, increment, modulus;
    /*-------------------------------------------------------- */
    /*                End of declares                          */
    /*-------------------------------------------------------- */

    r         = (double) seed;
    factor    = (double) multi;
    increment = (double) inc;
    modulus   = (double) modul;

    for(i=0; i< max_sweeps; i++) {
      r=fmod(r*factor + increment,modulus);
      x[i] = (float) r / modulus;
    }
    return 0;
}
```

Figure: C source for a modulo random number generator

```c
#include <stdlib.h>
int iseed, randInt;
float randFloat;

srand(iseed);
randInt = rand();
randFloat = (float) randInt / (float) RAND MAX;
```

Figure: C source for the implicit modulo random number generator

Add-with-Carry/Subtract-with-Carry Generators

- Add-with-carry and subtract-with-carry generators rely on two numbers, the carry c and the modular base M.
- Add-with-carry generator;

$$x_{n+1} = (x_{n-s} + x_{n-r} + c) \bmod M \tag{76}$$

- Subtract-with-carry

$$x_{n+1} = (x_{n-s} - x_{n-r} - c) \bmod M \tag{77}$$

- Problems:
  - Require an initial seed of a sufficiently long sequence.
  - Pairs (or triplets) of terms fall on planes (see modulo generator).

Fibonacci Generators

The **lagged Fibonacci generator**, symbolically denoted by LF(p,q,$\otimes$) with $p > q$, is based on a Fibonacci sequence of numbers with respect to an operation which we have given the generic symbol $\otimes$.

Let $\mathcal{S}$ be the model set for the operation $\otimes$, for example the positive real numbers, the positive integers, or the set $\mathcal{S} = \{0, 1\}$. The binary operation $\otimes$ computes a new number from previously generated numbers with a lag p

$$x_n = x_{n-p} \otimes x_{n-q} \quad , \quad p > q \qquad . \tag{78}$$

To start the generator we need p numbers. These can be generated using for example a modulo generator. The advantage of the lagged Fibonacci generator, apart from removing some of the deficiencies that are build into the modulo type generators, is that one can operate on the level of numbers or on the level of bits.

```
for(i=0; i< max_sweeps; i++) {
    mf[p] = mf[p] + mf[q];
    if ( mf[p] > 1 ) mf[p] -= 1;
    x[i] = mf[p];
    if ( ++p == lagP-1 )  p = 0;
    if ( ++q == lagP-1 )  q = 0;
}
```

Figure: Fibonicci

Random Numbers VIII

In the following I have listed some lagged Fibonacci generators:

Recursion Relation	Period
$x_i = x_{i-17} - x_{i-5} \bmod (2^n)$	$(2^{17}-1)2^{n-1}$
$x_i = x_{i-17} + x_{i-5} \bmod 2^n$	$(2^{17}-1)2^{n-1}$
$x_i = x_{i-31} - x_{i-13} \bmod 2^n$	$(2^{31}-1)2^{n-1}$
$x_i = x_{i-55} - x_{i-24}$	$2^{24}(2^{97}-1)$ with 24 Bit Mantissa

For example, we can construct a **generalized shift-register generator** GFSR(p,q,$\otimes$), where the operation is interpreted as the **exclusive or**, which acts on every of the 32 bits in a computer word. This generator is also known under the name of *R*250. (Follow this link to access the code for the R250.c.)

Random Numbers IX

```
R250.C                                                          Page 1 of 2
                                                    Printed For: Heermann

# include <math.h>
# define RAND_MAX 2147483647

/*=============================================================*/
/*                                                             */
/*       Random Number Generator:  R 2 5 0                     */
/*                                                             */
/*       program version 1.0 for C                             */
/*       Dieter W. Heermann                                    */
/*       may 1990                                              */
/*=============================================================*/

int init_r250(seed, m_f_ptr)
   int seed;
   int *m_f_ptr;
   {

    int  i,tmp, dummy, one ;
    int  *start;

    start = m_f_ptr;
    srand(seed);
    one    = 1;

    /* warm up the usual random number generator */
    for (i=0; i< 100; i++)
       {dummy = rand();
        }

    /* now draw the 250 (251)initial bit sequences */
    for (i=0 ; i<251; i++)
       {*m_f_ptr++ = rand();
       }

    /* now orthogonalize as best as we can  */
    m_f_ptr = start;
    for (i=0; i < 30; i++)
        { tmp      = *m_f_ptr;
  *m_f_ptr = tmp | one;
  one      = one << 1;
  m_f_ptr++;
        }

    return 0;
  }

int r250 (n, x_ptr, m_f_ptr, save)
   int    n;
   float  *x_ptr;
   int    *m_f_ptr;
   int    save;
   {

    int ind ;
    int j, min,k,ll;
    float *ran_ptr;

    ind     = save;
    ll      = n + 250;
    ran_ptr = x_ptr;
    j       = 1;
```

Figure: The shift bit register generator R250

Non Uniform Distributions

- Let us turn now to the generation of non-uniform distributions. First we look at the normal or Gaussian distribution.

- Typically algorithms generating non-uniform variates do so by converting uniform variates.
- In its most straightforward form a normal deviate x with mean $< x >$ and standard deviation σ is produced as follows:
- Let n be an integer, determined by the needed accuracy. Then
  - sum n uniform random numbers r_i from the interval $(-1, 1)$:

$$s_n = \sum_{i=\mathbf{1}}^{n} r_i$$

  - and let $x =< x > +\sigma s_n \sqrt{3.0/n}$.

Let $G(x)$ be a function on the interval $[a, b]$ with $0 < G(x) < 1$ and $f(x)$ the probability distribution $f(x) = a \exp[-G(x)]$, where a is a constant.

Algorithm 7 Algorithm

1: Generate r from a uniform distribution on $(0, 1)$
2: Set $x = a + (b - a)r$
3: Calculate $t = G(x)$
4: Generate $r_1, r_2, ..., r_k$ from a uniform distribution on $(0, 1)$ (k is determined from the condition $t > r_1 > r_2 > ... > r_{k-1} < r_k$)
5: **if** $t < r_1$ **then**
6: $k = 1$
7: **end if**
8: **if** k is even **then**
9: reject x and go to 1
10: **else**
11: x is a sample
12: **end if**

An interesting method for generating normal variates is the **polar method**. It has the advantage that two independent, normally distributed variates are produced with practically no additional cost in computer time.

Algorithm 8 Polar Algorithm

1: Generate two independent random variables, U_1, U_2 from the interval $(0, 1)$.
2: Set $V_1 = 2U_1 - 1$, $V_2 = 2U_2 - 1$
3: Compute| $S = V_1^2 + V_2^2$
4: **if** $S \geq 1$ **then**
5: return to step 1
6: **else**
7: $x_1 = V_1 \sqrt{-2 \ln S / S}$
8: $x_2 = V_2 \sqrt{-2 \ln S / S}$
9: **end if**

Accept/Reject Method I

- Another idea of converting one distribution into another is to accept or reject drawn number for an initial distribution such that the accepted numbers have the desired distribution.
- Assume that we are given a uniform random number generator $U \sim (0, 1)$ and $X \sim g$.
- We want to generate $Y \sim f$.
- Assume that there exists a constant c such that $f(x) < cg(x)$ for all x.

Algorithm 9 Accept/Reject Algorithm

1: Generate $X \sim g$
2: Generate $U \sim (0, 1)$
3: **if** $U \leq f(X)/cg(X)$ **then**
4: $\quad Y = X$
5: **else**
6: $\quad$ Goto 1
7: **end if**

Accept/Reject Method II

- To proof that this is correct we show that

$$P(X < y | U \leq f(X)/cg(X)) = P(Y \leq y) \quad .$$

Note that

$$\frac{P(X < y | U \leq f(X)/cg(X)) = P(Y \leq y)}{P(U \leq f(X)/cg(X))} = \frac{\int_{-\infty}^{y} \int_{0}^{f(x)/cg(x)} g(x) du dx}{\int_{-\infty}^{\infty} \int_{0}^{f(x)/cg(x)} g(x) du dx}$$

which simplifies to

$$\int_{-\infty}^{y} f(x) dx \quad .$$

Gibbs-Sampler I

Assume $x = (x^1, x^2)$ with target $\pi(x)$

Gibbs-Sampler Algorithm:

Algorithm 10 Gibbs Sampler Algorithm

1: initialize $x_0 = (x_0^1, x_0^2)$
2: **while** $i \leq$ mcsmax **do**
3: sample $x_i^1 \sim \pi(x^1 | x_{i-1}^2)$
4: sample $x_i^2 \sim \pi(x^2 | x_i^1)$
5: **end while**

$\{x^1, x^2\}$ is a Markov chain

Gibbs-Sampler I

Generalization: $x = (x^1, ..., x^p), p > 2$

Algorithm 11 Generalized Gibbs Sampler Algorithm

1: initialize $x_0 = (x^1, ..., x^p)$
2: **while** $i \leq$ mcsmax **do**
3: **for** $k = 1 \rightarrow p$ **do**
4: sample $x_i^k \sim \pi(x^k | x_i^{-k})$
5: **end for**
6: **end while**

where $x_i^{-k} = (x_i^1, ..., x_i^{k-1}, x_{i-1}^{k+1}, ..., x_{i-1}^p)$

Markov Chain Monte Carlo I

A **Markov chain** is a sequence of random variables $\{x_n; n \in \mathbb{N}\}$ which satisfies the property

$$P(x_n|x_0, ...x_{n-1}) = P(x_n|x_{n-1})$$

Goal: Given a distribution π, construct transition probabilities P such that asymptotically as $n \to \infty$

$$\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) \to \int \phi(x)\pi(x)dx$$

and

$$X_i \sim \pi$$

Examples:

- $\pi(x) = Z^{-1}\exp\{-\beta H(x)\}$
- Autoregression for $|\alpha| < 1$

$$X_n = \alpha X_{n-1} + V_n, V_n \sim N(0, \sigma^2)$$

Here $\pi(x) = \mathcal{N}(x; 0, \frac{\sigma^2}{1-\alpha^2})$

- Random Walk on a circle
  - The walker at time state i is in one of the four points
  - At state $i+1$ the walker jumps to one of the two neighbours with probability p and stays at the same point with probability $1-2p$.
- Let $P(x_{i+1}|x_i) =: P_{x_i,x_{i+\mathbf{1}}}$ then

$$P = \begin{pmatrix} 1-2p & p & 0 & p \\ p & 1-2p & p & 0 \\ 0 & p & 1-2p & p \\ p & 0 & p & 1-2p \end{pmatrix}, p \leq 1/2$$

- $P_{x,y} \geq 0, \sum_y P_{xy} = 1$
- Calculate P^n

$$\lim_{n\to\infty} P^n = \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix} = \pi$$

a uniform distribution, as expected!

Markov Chain Monte Carlo I

We want:

- The desired distribution π to be a fixed point of the algorithm

$$\int P(x, y)\pi dx = \pi(y)$$

- The successive distributions of the Markov chain converges towards π
- The estimator $\frac{1}{n}\sum_i^n \phi(x_i)$ converges and asymptotically $X_n \sim \pi$
- For every π there exists infinitely many P that have π as there invariant distribution.
- How to choose a good one? *Criterion*: Rate of convergence
- Convergence ensured if the chain is irreducible, aperiodic and every state can be reached.

We require:

- Irreducibility: From any state we can go to any state

$$\forall x, y \in S \quad \exists n \in \mathbb{N}, n > 0 : (P^n)_{xy} > 0$$

- Aperiodicty

then, if P is irreducible and aperiodic we have

- $\lim_{n\to\infty} P^n = \pi$

- if π_y is not identically zero, then

$$\sum_{y \in S} \pi_y = 1, \quad \sum_{y \in S} \pi_y P_{yx} = \pi_x$$

- π_x is a unique non-negative solution of the above equation and a probability distribution on S.

If we know π_x construct P such that π_x is its equilibrium distribution. For this it is enough to choose P such that

1. $\sum_{y \in S} \pi_y P_{yx} = \pi_x$
2. P is irreducible and aperiodic

Replace (1) by the **detailed balance condition**

$$\pi_y P_{yx} = \pi_x P_{xy}$$

which implies (1).

Metropolis-Hastings Monte Carlo I

We will now construct an algorithm to that realizes a constant temperature ensemble, i.e.

$$\pi(x) = Z^{-1} \exp\{-\beta H(x)\}$$

The state space can for example be configurations of spins (see later Ising model), positions of atoms in a liquid, polymers conformation etc.

- Let $P^0 = \{p_{xy}^{(0)}\}$ be a irreducible transition matrix.
- We will use P^0 to propose transitions from x to y.
- These transitions will then either be accepted with a probability a_{xy} and rejected $1 - a_{xy}$.

Metropolis-Hastings Monte Carlo II

- The complete transition matrix P is then constructed as:

$$\begin{aligned} p_{xy} &:= p^{(0)}_{xy} a_{xy} \quad \text{if } x \neq y\,, \\ p_{xx} &:= p^{(0)}_{xx} + \sum_{x \neq y} p^{(0)}_{xy}(1 - a_{xy}) \quad \text{[zero transition]}\,, \\ \text{where} \quad & \forall \;\; x, y \in S : 0 \leq a_{xy} \leq 1\,. \end{aligned}$$

With this we have

$$\frac{a_{xy}}{a_{yx}} = \frac{\pi_y p^{(0)}_{yx}}{\pi_x p^{(0)}_{xy}} \quad \forall x, y \in S : x \neq y\,.$$

Metropolis-Hastings Monte Carlo III

- If we use

$$a_{xy} := F\left(\frac{\pi_y \rho^{(0)}_{yx}}{\pi_x \rho^{(0)}_{xy}}\right) \; \forall x, y \in \mathcal{S} : x \neq y \tag{79}$$

then

$$\frac{a_{xy}}{a_{yx}} = \frac{F\left(\frac{\pi_y p^{(0)}_{yx}}{\pi_x p^{(0)}_{xy}}\right)}{F\left(\frac{\pi_x p^{(0)}_{xy}}{\pi_y p^{(0)}_{yx}}\right)} = \frac{F(z)}{F(1/z)} \stackrel{!}{=} z \tag{80}$$

with

$$z := \frac{\pi_x p^{(0)}_{xy}}{\pi_y p^{(0)}_{yx}} \quad . \tag{81}$$

- The condition of detailed balance is satisfied if

$$\forall z : \frac{F(z)}{F(1/z)} \stackrel{!}{=} z \quad . \tag{82}$$

Metropolis-Hastings Monte Carlo IV

- Often used choices are

$$F(z) = \min(z, 1) \tag{83}$$

and

$$F(z) = \frac{z}{1+z} \quad . \tag{84}$$

- Note that the proposals of states need not to be symmetric. As a further point we note that is was proven[52] that the choice $F(z) = \min(z, 1)$ is optimal in that suitable candidates are rejected least often and hence statistical efficiency is optimized

Metropolis-Hastings Monte Carlo Algorithm I

[1, 53]

Algorithm 12 Metropolis-Hastings Monte Carlo Algorithm

1: **for** i=0; i $<$ mcsmax **do**
2: sample a new state x;
3: set $y = x_i$;
4: sample a uniform random number r from $(0, 1)$;
5: **if** $r \leq \min\{\frac{p^{\mathbf{0}}_{xy}\pi_x}{p^{\mathbf{0}}_{x}y\pi_y}, 1\}$ **then**
6: $x_{i+1} = x$;
7: **else**
8: $x_{i+1} = y$;
9: **end if**
10: **end for**

- What does this mean if we calculate the time average of an observable A, which by necessity can cover only a finite observation time?
- Let us consider the statistical error for n successive observations $A_i, i = 1, ..., n$:

$$\left\langle (\delta A)^2 \right\rangle = \left\langle \left[n^{-1} \sum_{i=1}^{n} (A_i - \langle A \rangle)^2 \right] \right\rangle \quad . \tag{85}$$

- In terms of the autocorrelation function for the observable A

$$\phi_A(t) = \frac{\langle A(0)A(t) \rangle - \langle A \rangle^2}{\langle A^2 \rangle - \langle A \rangle^2} \tag{86}$$

We define two characteristic correlation times.

- **Exponential autocorrelation time**
  - Typically we expect that (asymptotically, for large t) one gets an exponential behavior

$$\Phi_A(t) \propto \exp\left(-\frac{t}{\tau_{A,exp}} \right) \tag{87}$$

  - We do expect, though, that the complete expression involves a sum over several such terms; here we consider only the asymptotically most leading term with largest autocorrelation time.

- **Integrated autocorrelation time**

$$\tau_A^{int} = \int_0^\infty \phi_A(t)dt \quad . \tag{88}$$

- We can rewrite the statistical error as

$$\left\langle (\delta A)^2 \right\rangle \cong \frac{2\tau_A}{n\delta t} \left[\left\langle A^2 \right\rangle - \langle A \rangle^2 \right] \quad , \tag{89}$$

  where δt is the time between observations, i.e., $n\delta t$ is the total observation time τ_{obs}.
- We notice that the error does not depend on the spacing between the observations but on the total observation time.
- Also the error is not the one which one would find if all observations were independent.
- The error is enhanced by the characteristic (integral) correlation time between configurations.
- Only an increase in the sample size and/or a reduction in the characteristic correlation time τ_A can reduce the error.

Hybrid (Hamiltonian) Monte Carlo I

- In conventional Monte-Carlo (MC) calculations of condensed matter systems, such as an N-particle system with a Hamiltonian $\mathcal{H} = \mathcal{U}$, only local moves (displacement of a single particle) are made.
- Updating more than one particle typically results in a prohibitively low average acceptance probability $\langle P_A \rangle$.
- This implies large relaxation times and high autocorrelations especially for macromolecular systems.
- In a Molecular Dynamics (MD) simulation, with $\mathcal{H} = \mathcal{T} + \mathcal{U}$, on the other hand, global moves are made.
- The MD scheme, however, is prone to errors and instabilities due to the finite step size in time.
- In order to introduce temperature in the microcanonical context, isokinetic MD schemes are often used.
- However, they do not yield the canonical probability distribution, unlike Monte-Carlo calculations.

Hybrid (Hamiltonian) Monte Carlo II

- The Hybrid Monte-Carlo (HMC) method combines the advantages of Molecular Dynamics and Monte-Carlo methods:
  - it allows for global moves (which essentially consist in integrating the system through **phase** space);
  - HMC is an exact method, i.e., the ensemble averages do not depend on the step size chosen;
  - algorithms derived from the method do not suffer from numerical instabilities due to finite step size as MD algorithms do;
  - and temperature is incorporated in the correct statistical mechanical sense.
- In the HMC scheme global moves can be made while keeping the average acceptance probability $\langle P_A \rangle$ high.

Hybrid (Hamiltonian) Monte Carlo III

- One global move in **configuration** space consists in integrating the system through **phase** space for a fixed time t using some discretization scheme (δt denotes the step size)

$$\begin{aligned} g^{\delta t}: \quad \boldsymbol{R}^{6N} &\longrightarrow \boldsymbol{R}^{6N} \\ (x,p) &\longrightarrow g^{\delta t}(x,p) =: (x',p') \end{aligned}$$

  of Hamilton's equations

$$\begin{aligned} \frac{dx}{dt} &= \frac{\partial \mathcal{H}}{\partial p} \\ \frac{dp}{dt} &= -\frac{\partial \mathcal{H}}{\partial x}. \end{aligned} \tag{90}$$

- Since the system is moved deterministically through phase space, the conditional probability of suggesting configuration x' starting at x is given by

$$p_C(x \to x')dx' = p_C(p)dp. \tag{91}$$

Hybrid (Hamiltonian) Monte Carlo IV

- The initial momenta are drawn from a Gaussian distribution at inverse temperature β:

$$p_C(p) \propto e^{-\beta \sum_{j=1}^{N} \frac{p_j^2}{2m}}. \tag{92}$$

- Thus

$$P_A((x,p) \to g^{\delta t}(x,p)) = \min\{1, e^{-\beta \delta \mathcal{H}}\}, \tag{93}$$

where

$$\delta \mathcal{H} = \mathcal{H}(g^{\delta t}(x,p)) - \mathcal{H}(x,p)$$

is the discretization error associated with $g^{\delta t}$. Using the algebraic identity

$$e^{-\mathcal{H}(x,p)} \min\{1, e^{-\delta \mathcal{H}}\} = e^{-\mathcal{H}(g^{\delta t}(x,p))} \min\{e^{\delta \mathcal{H}}, 1\} \tag{94}$$

- it can be shown that for a discretization scheme which is **time-reversible**

$$g^{-\delta t} \circ g^{\delta t} = id \tag{95}$$

Hybrid (Hamiltonian) Monte Carlo V

- and **area-preserving**

$$\det \frac{\partial g^{\delta t}(x,p)}{\partial (x,p)} = 1, \tag{96}$$

detailed balance is satisfied:

$$\begin{aligned} p(x)p_M(x \to x')dxdp &= p(x)p_C(p)P_A((x,p) \to g^{\delta t}(x,p))dxdp \\ &= p(x')p_C(p')P_A(g^{\delta t}(x,p) \to (x,p))dxdp \\ &= p(x')p_C(p')P_A((x',p') \to g^{-\delta t}(x',p'))dxdp \\ &= p(x')p_C(p')P_A((x',p') \to g^{-\delta t}(x',p'))dx'dp' \\ &= p(x')p_M(x' \to x)dx'dp'. \end{aligned}$$

- Thus, provided the discretization scheme used is **time-reversible** and **area-preserving**, the HMC algorithm generates a Markov chain with the stationary probability distribution $p(x)$.

- The probability distribution is entirely determined by the detailed balance condition.
- Therefore neither $p(x)$ nor any ensemble averages depend on the step size δt chosen.

Hybrid (Hamiltonian) Monte Carlo VI

- However, the average acceptance probability $\langle P_A \rangle$, because of (93), depends on the average discretization error $\langle \delta\mathcal{H} \rangle$ and hence does depend on δt.
- It can be shown that for $(\varrho, T) \neq (\varrho_c, T_c)$

$$\langle P_A \rangle = \operatorname{erfc}(\frac{1}{2}\sqrt{\beta \langle \delta\mathcal{H} \rangle})$$

  is a good approximation for sufficiently large systems ($N \to \infty$) and small step sizes ($\delta t \to 0$).

- From normalization and the area-preserving property one has

$$\langle e^{-\beta \delta\mathcal{H}} \rangle = 1. \tag{97}$$

- Equation (97) can be expanded into cumulants

$$\langle \delta\mathcal{H} \rangle = \frac{\beta}{2} \langle (\delta\mathcal{H} - \langle \delta\mathcal{H} \rangle)^2 \rangle + \cdots \quad .$$

- In order to obtain a nonzero average acceptance probability $\langle P_A \rangle$ in the limit $N \to \infty$ one has to let $\delta t \to 0$, keeping $\langle (\delta\mathcal{H} - \langle \delta\mathcal{H} \rangle)^2 \rangle$ fixed.

Hybrid (Hamiltonian) Monte Carlo VII

- In this limit higher-order cumulants will vanish. The resulting distribution of the discretization error will thus be gaussian with mean and width related through

$$\langle \delta\mathcal{H} \rangle = \frac{\beta}{2} \langle (\delta\mathcal{H} - \langle \delta\mathcal{H} \rangle)^2 \rangle. \tag{98}$$

- From (93) and (98) one has in this case

$$\begin{aligned} \langle P_A \rangle &= \frac{1}{\sqrt{2\pi \langle (\delta\mathcal{H} - \langle \delta\mathcal{H} \rangle)^2 \rangle}} \int_{-\infty}^{\infty} dt \, \min\{1, e^{-\beta t}\} e^{-\frac{(t - \langle \delta\mathcal{H} \rangle)^2}{2\langle (\delta\mathcal{H} - \langle \delta\mathcal{H} \rangle)^2 \rangle}} \\ &= \operatorname{erfc}\left(\frac{1}{2}\sqrt{\beta \langle \delta\mathcal{H} \rangle}\right). \end{aligned} \tag{99}$$

- The square root in (99) is always well defined since (97) implies

$$\langle \delta\mathcal{H} \rangle \geq 0.$$

- Equality holds in the limit $\delta t \to 0$, where energy is conserved exactly and $\langle P_A \rangle = 1$.
- Increasing the step size will result in a lower average acceptance probability $\langle P_A \rangle$. Varying δt, the average acceptance probability $\langle P_A \rangle$ can thus be adjusted to minimize autocorrelations.

Hybrid (Hamiltonian) Monte Carlo VIII

- The momenta do not necessarily have to be drawn from the Gaussian distribution.
- A particularly simple and computationally efficient alternative to would be a uniform momentum distribution.
- This choice, however, did not prove successful, since a cut-off has to be introduced for computational reasons. This cut-off must be taken into account in P_A, leading to a very low average acceptance probability $\langle P_A \rangle$.
- It is clear that instead of choosing a discretization scheme of Hamilton's equations (90) any time-reversible and area-preserving discrete mapping can be used to propagate the system through phase space.

Rejection-Free Monte Carlo I

- This simulates a very common phenomenon: [54]

  There is some underlying dynamic system running along according to simple and uncertain dynamics, but we cannot see it.

- All we can see are some noisy signals arising from the underlying system. From those noisy observations we want to do things like predict the most likely underlying system state, or the time history of states, or the likelihood of the next observation

What are Hidden Markov Models good for?

- useful for modeling protein/DNA sequence patterns
- probabilistic state-transition diagrams
- Markov processes - independence from history
- hidden states

Where does one use HMM's

- protein families
- DNA patterns
- secondary structure (helix, strand, coil (each has 20x20 table with transition frequencies between neighbors $a_i \rightarrow a_{i+1}$))
- protein fold recognition

- fold classification
- gene silencing
- ...

- **Example: CpG - Islands**

  - Regions labeled as CpG - islands $\longrightarrow$ + model
  - Regions labeled as non-CpG - islands $\longrightarrow$ - model
  - Maximum likelihood estimators for the transition probabilities for each model

$$a_{st} = \frac{c_{st}}{\sum_{t'} c_{st'}}$$

  and analogously for the - model. c_{st} is the number of times letter t followed letter s in the labeled region

Definition I

- A Hidden Markov Model is a two random variable process, in which one of the random variables is hidden, and the other random variable is observable.
- It has a finite set of states, each of which is associated with a probability distribution.
- Transitions among the states are governed by transition probabilities.
- In a particular state an observation can be generated, according to the associated probability distribution.
- It is only the observation, not the state visible to an external observer, and therefore states are "hidden" from the observer.

- **Example:**
  - For DNA, let + denote coding and - non-coding. Then a possible observed sequence could be

$$O = AACCTTCCGCGCAATATAGGTAACCCCGG$$

    and

$$Q = - - + + + + + + + + + + + + + + + + + + - - - - - - - -$$

- Question: How can one find CpG - islands in a long chain of nucleotides?
- Merge both models into one model with small transition probabilities between the chains.

Definition II

A Hidden Markov Model (HMM) $\lambda =< Y, S, A, B >$ consists of:

- an output alphabet $Y = \{1, ..., b\}$
- a state space $S = \{1, ..., c\}$ with a unique initial state s_0
- a transition probability distribution $A(s'|s)$
- an emission probability distribution $B(y|s, s')$

- HMMs are equivalent to (weighted) finite state automata with outputs on transitions
- Unlike MMs, constructing HMMs, estimating their parameters and computing probabilities are not so straightforward

Definition III

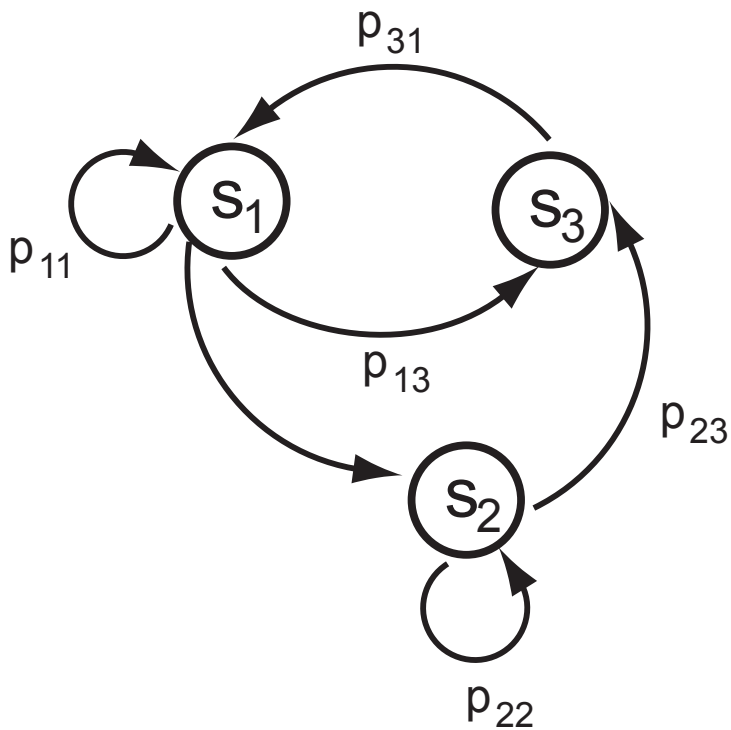


Figure: Example of a Hidden Markov Model.

Definition IV

Given a HMM λ and a state sequence $S = (s_1, ..., s_{t+1})$, the probability of an output sequence $O = (o_1, ..., o_t)$ is

$$P(O|S, \lambda) = \prod_{i=1}^{t} P(o_i|s_i, s_{i+1}, \lambda) = \prod_{i=1}^{t} B(o_i|s_i, s_{i+1}). \tag{100}$$

Given λ, the probability of a state sequence $S = (s_1, ..., s_{t+1})$ is

$$P(S|\lambda) = \prod_{i=1}^{t} P(s_{i+1}|s_i) = \prod_{i=1}^{t} A(s_{i+1}|s_i). \tag{101}$$

Of importance is the probability of an output sequence $O = (o_1, ..., o_t)$ under a given λ. It is easy to show that the straightforward computation yields

$$P(O|\lambda) = \sum_{S} \prod_{i=1}^{t} A(s_{i+1}|s_i) B(o_i|s_i, s_{i+1}) \tag{102}$$

with a computational complexity of $(2c + 1) * c^{t+1}$ multiplications.

Definition V

Multiple Sequence Alignments

- In theory, making an optimal alignment between two sequences is computationally straightforward (Smith-Waterman algorithm), but aligning a large number of sequences using the same method is almost impossible (e.g. $O(t^N)$).
- The problem increases exponentially with the number of sequences involved (the product of the sequence lengths).
- Statistical Methods:
  - Expectation Maximization Algorithm (deterministic)
  - Gibbs Sampler (stochastic)
  - Hidden Markov Models (stochastic)
- Advantages for HMM: theoretical explanation, no sequence ordering, no insertion and deletion penalties, using prior information
- Disadvantage for HMM: large number of sequences for training

Definition VI

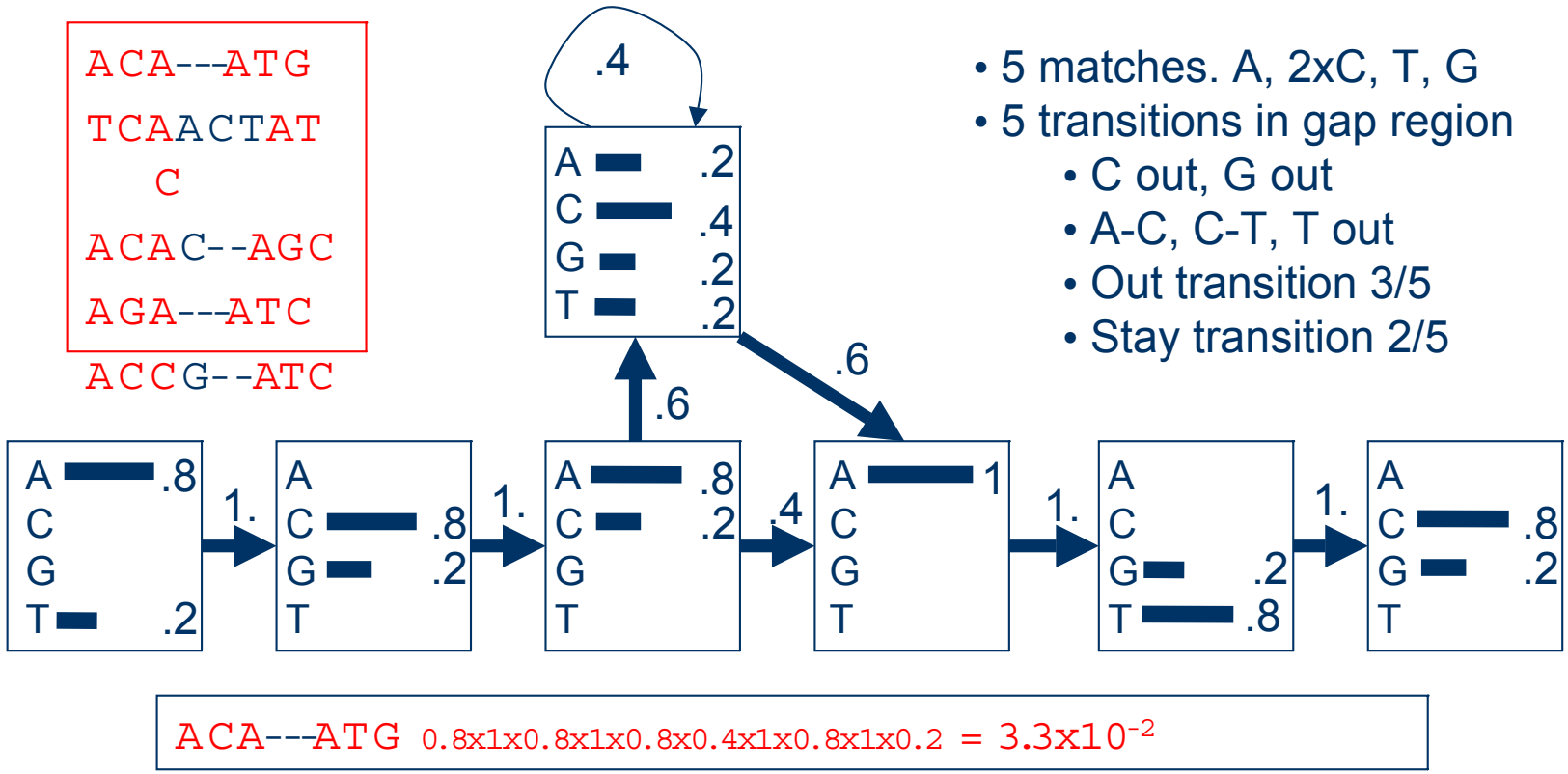


Figure: Example of a Hidden Markov Model.

Definition VII

There are three basic problems:

1. Given a model, how likely is a specific sequence of observed values (evaluation problem).
2. Given a model and a sequence of observations, what is the most likely state sequence in the model that produces the observations (decoding problem).
3. Given a model and a set of observations, how should the model's parameters be updated so that it has a high probability of generating the observations (learning problem).

Forward algorithm I

- We define $\alpha_s(i)$ as the probability being in state s at position i:

$$\alpha_s(i) = P(o_1, ..., o_i, s_i = s|\lambda) \tag{103}$$

- Base case: $\alpha_s(1)$ if $s = s_0$ and $\alpha_s(0) = 0$ otherwise
- Induction:

$$\alpha_s(i+1) = \max_{s \in S} A(s|s')B(o_i|s', s)\alpha_s(i) \tag{104}$$

- Finally, at the end:

$$P(o_1, ..., o_k|\lambda) = \sum_{s \in S} \alpha_s(k) \tag{105}$$

- Partial sums could as well be computed right to left (backward algorithm), or from the middle out
- In general, for any position i:

$$P(O|\lambda) = \sum_{s \in S} \alpha_s(i)\beta_s(i) \tag{106}$$

- This algorithm could be used, e.g. to identify which λ is most likely to have produced an output sequence O

Forward algorithm II

What is the most probable path given observations (decoding problem)?

- given $o_1, ..., o_t$ what is

$$\text{argmax}_S P(s, o_1, ...o_t|\lambda)$$

- slow and stupid answer:

$$\text{argmax}_S \frac{P(o_1, ..., o_t|s)P(s)}{P(o_1, ..., o_t)}$$

Viterbi algorithm I

- We define $\delta_s(i)$ as the probability of the most likely path leading to state s at position i:

$$\delta_s(i) = \max_{s_1,\ldots,s_{i-1}} P(s_1, ..., s_{i-1}, o_1, ..., o_{i-1}, s_i = s|M) \tag{107}$$

- Base case: $\delta_s(1)$ if $s = s_0$ and $\delta_s(0) = 0$ otherwise
- Again we proceed recursively:

$$\delta_s(i+1) = \max_{s \in S} A(s|s')B(o_i|s', s)\delta_s(i) \tag{108}$$

  and since we want to know the identity of the best state sequence and not just its probability, we also need

$$\Psi(i+1) = \text{argmax}_{s \in S} A(s|s')B(o_i|s', s)\delta_s(i) \tag{109}$$

- Finally, we can follow Ψ backwards from the most likely final state
- The Viterbi algorithm efficiently searches through $|S|^T$ paths for the one with the highest probability in $O(T|S|^2)$ time
- In practical applications, use log probabilities to avoid underflow errors
- Can be easily modified to produce the n best paths
- A beam search can be used to prune the search space further when $|S|$ is very large (n-gram models)

- Predicting the next state s_n depending on $s_1, ..., s_{n-1}$ results in

$$P(s_n|s_1, ..., s_{n-1})$$

- Markov Assumption $(n-1)$th order : last $n-1$ states are in the same equiv. class

Parameter estimation

- Given an HMM with a fixed architecture, how do we estimate the probability distributions A and B?
- If we have labeled training data, this is not any harder than estimating non-Hidden Markov Models (supervised training):

$$A(s'|s) = \frac{C(s \to s')}{\sum_{s''} C(s \to s'')} \quad (110)$$

$$B(o|s, s') = \frac{C(s \to s', o)}{C(s \to s')} \quad (111)$$

Forward-Backward Algorithm I

- Also known as the *Baum-Welch algorithm*
- Instance of the *Expectation Maximization (EM) algorithm*:
  1. Choose a model at random
  2. E: Find the distribution of state sequences given the model
  3. M: Find the most likely model given those state sequences
  4. Go back to 2.
- Our estimate of A is:

$$A(s'|s) = \frac{E[C(s \to s')]}{E[C(s \to ?)]} \tag{112}$$

- We estimate $E[C(s \to s')]$ via $\tau_t(s, s')$, the probability of moving from state s to state s' at position t given the output sequence O:

$$\begin{aligned} \tau_t(s, s') &= P(s_t = s, s_{t+1} = s'|O, \lambda) && (113)\\ &= \frac{P(s_t = s, s_{t+1} = s', O|\lambda)}{P(O|\lambda)} && (114)\\ &= \frac{\alpha_s(t)A(s|s')B(o_{t+1}|s, s')\beta_{s'}(t+1)}{\sum_{s''} \alpha_{s''}} && (115) \end{aligned}$$

Forward-Backward Algorithm II

- This lets us estimate A:

$$A(s'|s) = \frac{\sum_t \tau_t(s, s')}{\sum_t \sum_{s''} \tau_t(s, s'')} \tag{116}$$

- We can estimate B along the same lines, using $\sigma_t(o, s, s')$, the probability of emitting o while moving from state s to state s' at position t given the output sequence O
- Alternate re-estimating A from τ, then τ from A, until estimates stop changing
- If the initial guess is close to the right solution, this will converge to an optimal solution

Excercises I

Exercise 1: **Hard Disk Potential.**

$$u_{ij}(r) = \begin{cases} 0 & \text{if} \quad \mathbf{r} > r_c \\ \infty & \text{otherwise} \end{cases}$$

Exercise 2: **Lorentz-Berelot combining rule.**

$$\sigma_{\alpha\beta} = \frac{\sigma_{\alpha\alpha} + \sigma\beta\beta}{2} \tag{117}$$

$$\epsilon_{\alpha\beta} = \sqrt{\epsilon_{\alpha\alpha} + \epsilon_{\beta\beta}} \tag{118}$$

Exercise 3: **Morse Potenial.**

$$u_{ij}(r) = k\left(1 - e^{a(r - r_0)}\right)^2$$

Exercise 4: **Mie-Potential** Mie Potential[55]

$$U(r) = A/r^m - B/r^n$$

Exercise 5: **Random Number Generator**

For some purposes the simple method will be sufficient, but if good accuracy is needed the above algorithm should be avoided. More efficient and accurate is the idea of von Neumann [56] with the modification of Forsythe [57].

Let $G(x)$ be a function on the interval $[a, b]$ with $0 < G(x) < 1$ and $f(x)$ the probability distribution $f(x) = a \exp[-G(x)]$, where a is a constant.

Algorithm 13 v. Neumann Algorithm

1: Generate r from a uniform distribution on $(0, 1)$
2: Set $x = a + (b - a)r$
3: Calculate $t = G(x)$
4: Generate $r_1, r_2, ..., r_k$ from a uniform distribution on $(0, 1)$
5: k is determined from the condition $t > r_1 > r_2 > ... > r_{k-1} < r_k$
6: **if** $t < r_1$ **then**
7: $k = 1$
8: **end if**
9: **if** k is even **then**
10: reject x and go to 1
11: **else**
12: x is a sample
13: **end if**

Excercises IV

Exercise 6: **Permutation test**

Divide up the sequence random numnbers into non-overlapping subsequences of a fixed length n. For each subsequence, assume that the values in the subsequence are unique. Replace each value by its ranking based on its magnitude. The largest value is replaced by n. Each subsequence represents a permutation of the integers from 1 to n.
Now count how often each permutation occurs. Each permutation should occur with same frequency.
Use your favourite random numnber generator and perfom the permutation test.

Exercise 7: Generate a random sequence of a total of n letter from the follwing set $\{'A';' C';' G';' T'\}$. How do you test for randomness of the sequence?

Bibliography I

[1] M. N. Rosenbluth A. H. Teller N. Metropolis, A. W. Rosenbluth and E. Teller. *J. Chem. Phys.*, 21:1087–1091, 1953.

[2] T. E. Wainwright B. J. Alder. *J. Chem. Phys.*, 27(1208), 1957.

[3] A. Rahman. *Phys. Rev. A*, 136:405, 1964.

[4] A. Rahman and F. Stillinger. Molecular dynamics study of liquid water. *J. Chem. Phys.*, 55(5), 1971.

[5] M. Karplus P.G. Wolynes J.A. McCammon, B.R. Gelin. *Nature*, 262:325–26, 1976.

[6] J. P. Valleau G. Torrie. *J. Comput. Phys*, 23:187, 1977.

[7] Bernd A. Berg and Thomas Neuhaus. Multicanonical ensemble: A new approach to simulate first-order phase transitions. *Physical Review Letters*, 68(1):9–12, 01 1992.

[8] Fugao Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical Review Letters*, 86(10):2050–2053, 03 2001.

[9] Berend Smit and Daan Frenkel. *Understanding Molecular Simulation*. Academic Press, 2001.

[10] Andrew R. Leach. *Molecular Modelling: Principles and Applications*. Prentice-Hall, 20012.

Bibliography II

[11] K. Binder and D.W. Heermann. *Monte Carlo Simulation in Statistical Physics*. Springer-Verlag (first editon 1988), 2017.

[12] P. Allen and D.J. Tildesley. *Computer Simulations of Liquids*. Clarendon Press, Oxford, 1987.

[13] Lei Liu and Dieter W Heermann. The interaction of dna with multi-cys 2 his 2 zinc finger proteins. *Journal of Physics: Condensed Matter*, 27(6):064107, 2015.

[14] D. Chandler J. D. Weeks and H. C. Andersen. *J. Chem. Phys.*, 54:5237, 1971.

[15] J. E. Jones. On the determination of molecular fields. i. from the variation of the viscosity of a gas with temperature. *Proceedings of the Royal Society of London. Series A*, 106(738):441, 10 1924.

[16] J. E. Jones. On the determination of molecular fields. ii. from the equation of state of a gas. *Proceedings of the Royal Society of London. Series A*, 106(738):463, 10 1924.

[17] A. D. MacKerell, D. Bashford, M. Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiórkiewicz-Kuczera, D. Yin, and M. Karplus. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *The Journal of Physical Chemistry B*, 102(18):3586–3616, 04 1998.

Bibliography III

[18] Robert L Jernigan and Ivet Bahar. Structure-derived potentials and protein simulations. *Current Opinion in Structural Biology*, 6(2):195–209, 1996.

[19] S. Miyazawa and R.L. Jernigan. *Macromolecules*, 18:534–552, 1985.

[20] Miriam Fritsche, Ras B. Pandey, Barry L. Farmer, and Dieter W. Heermann. Variation in structure of a protein (h2ax) with knowledge-based interactions. *PLoS ONE*, 8(5):e64507–, 05 2013.

[21] S.M. Omohundro. Five balltree construction algorithms. Technical report, Tech. rep., ICSI Berkeley, 1989.

[22] Ashraf M. Kibriya and Eibe Frank. *An Empirical Comparison of Exact Nearest Neighbour Algorithms*, pages 140–151. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[23] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[24] M. S. Warren and J. K. Salmon. A parallel hashed oct-tree n-body algorithm, 1993.

[25] A. Guttman. R-trees: a dynamic index structure for spatial searching. *ACM*, 14, 1984.

[26] Kriegel H.P. Schneider R. Seeger B. Beckmann, N. The r*-tree: An efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331, 1990.

Bibliography IV

[27] B. J. Alder T. E. Wainwright. *Nuovo cimento, Suppl. Sec.*, 9:116, 1958.

[28] T. E. Wainwright B. J. Alder. *J. Chem. Phys.*, 31:456, 1959.

[29] T. E. Wainwright B. J. Alder. *J. Chem. Phys.*, 33:1439, 1960.

[30] 1439 B. J. Alder, T. E. Wainwright **33**. *J. Chem. Phys.*, 33:1439, 1960. R. Beeler, Jr:. In *Physics of Many-Particle Syslems*, ed. by C. Meeron (Gordon and Breach, New York 1964).

[31] L. Verlet. *Phys. Rev.*, 159:98, 1967.

[32] Y. B. Suris. *Comput. Math. Phys.*, 27:149–156, 1987.

[33] M. Duncan B. Gladman and J. Candy. Symplectic integrators for long-term integrations in celestial mechanics. *Celestial Mech. Dynam. Astronom*, 52:221–240, 1991.

[34] J. M. Sanz-Serna and M. P. Calvo. *Numerical Hamiltonian Problems*. Chapman and Hall, London, 1994.

[35] Randall B. Shirts, Scott R. Burt, and Aaron M. Johnson. Periodic boundary condition induced breakdown of the equipartition principle and other kinetic effects of finite sample size in classical hard-sphere molecular dynamics simulation. *The Journal of Chemical Physics*, 125(16), 2006.

[36] M. L. Klein S. Nose. *J. Chem. Phys.*, 78:6928, 1983. Dahlquist, A. Bjorck: Numerical Methods (Prentice Hall, Englewood Cliffs, NJ 1964).

Bibliography V

[37] P. H. Berens K. R. Wilson: W. C. Swope, H. C. Andersen. *J. Chem. Phys.*, 76:637, 1982.

[38] R. B. Hickman A.J.C. Ladd W.T. Ashurst B. Moran W. G. Hoover, D. J. Evans. *Phys. Rev. A*, 22(690), 1980.

[39] W. G. Hoover. *Physica A*, 18, 1983.

[40] W. G. Hoover. In H.J. Hanley, editor, *In Nonlinear Fluid Behaviour*. North-Holland, Amsterdam, 1983.

[41] B. Moran W. G. Hoover, A. J.C. Ladd. *Phys. Rev. Lett.*, 48:3297, 1983.

[42] G. P. Morriss D. J. Evans. *Chem. Phys.*, 77(63), 1983.

[43] G. P. Morriss D. J. Evans. *Chem. Phys.*, 77:63, 1983.

[44] S. Gupta J. M. Haile. *J. Chem. Phys*, 79:3067, 1983.

[45] G. P. Morriss D. M. Heyes, D. J. Evans. 1985. In *Daresbury Lab. Information Quarterly for Computer Simulation of Condensed Phases*, volume 17. 1985.

[46] J. H.R. Clarke D. Brown. 1984. *Mol. Phys*, (1243), 1984.

[47] J. R. Ray. *Am. J. Phys.*, 40:179, 1972.

[48] H. C. Andersen. *J. Chem. Phys.*, (72):2384, 1980.

[49] H. C. Andersen. *J. Chem. Phys.*, (72):2384, 1980.

[50] H. W. Graben J. M. Haile. *J. Chem. Phys.*, 73:2412, 1980.

Bibliography VI

[51] G. van Kampen. *Stochastic Processes in Physics and Chemistry*. North Holland, Amsterdam, 1981.

[52] P. H. Peskun. *Biometrika*, 60:607–612, 1973.

[53] W. K. Hastings. *Biometrika*, 57:97–109, 1970.

[54] A. Krogh R. Durbin, S.R. Eddy and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.

[55] Gustav Mie. Zur kinetischen theorie der einatomigen körper. *Annalen der Physik*, 11:657–697, 1903.

[56] 1192 S. K. Park, K. W. Miller *Comm. ACM* **31**. J. 1988. von Neumann: Various Techniques Used in Connection with Random Digits, Collected Works, Vol. 5 (Pergamon, New York 1963).

[57] 817 G. E. Forsythe: Math. Comput. **26**. 1972. 1972.

Index I

Index II

Index III