UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Theoretical Biophysics
A Computational Approach
Concepts, Models, Methods and Algorithms
Methods

Dieter W. Heermann

April 20, 2020

Heidelberg University

Table of Contents

Introduction

Introduction I

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.

Let us consider the time and space scales that are involved modelling biophysical systems. Biological systems range in spatial scales from the

- atom level $10^{-10}m$
- molecular scale $10^{-9}m$
- organism level 1 m

to times scales of

- typical vibrational frequencies $10^{-13}s$
- molecular interaction $10^{-3}s$
- human lifespan $10^{9}s$

Introduction II

Hence, both in time and space we need to cover many orders of magnitude. To deal with this, we need to describe systems on the relevant scales on which they exhibit a particular behaviour. In the same spirit, methods need to be developed to investigate the model on that particular scale. Thus this separation of scales needs to be bridged.

A challenge is the coupling between the models and methods (multi-scale models / methods). Figure 1 illustrates the general idea. The system is modeled on three different scales. Couplings describe the interaction between the scales such that for example the temperature remains invariant.

Introduction III

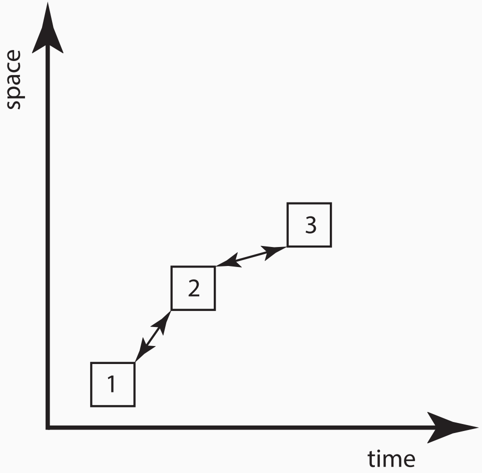


Figure 1: Scales in a multi scale model and the interaction between the scales

Introduction IV

An example multi-scale of models / methods will be put forward in a lecture later on. 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]

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 $T = 2.53$.

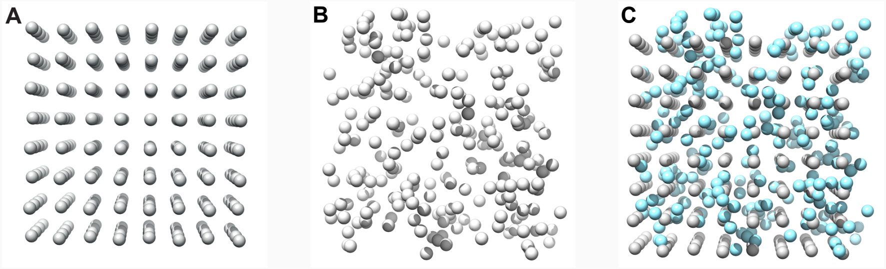


Figure 2: Panel A shows crystal structure, B shows a liquid structure and C an overlay of the two for comparison. Note that the radius with which the particles are depicted does not reflect the size of the particle.

General Remarks II

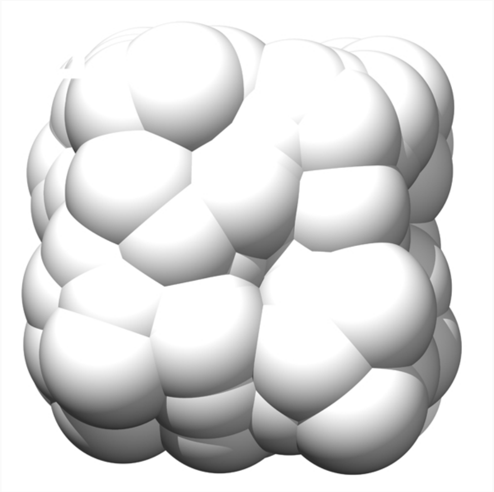


Figure 3: Lennard-Jones liquid where the particles are depicted with a radius where the Lennard-Jones potential is zero (see later in the section on force fields).

General Remarks III

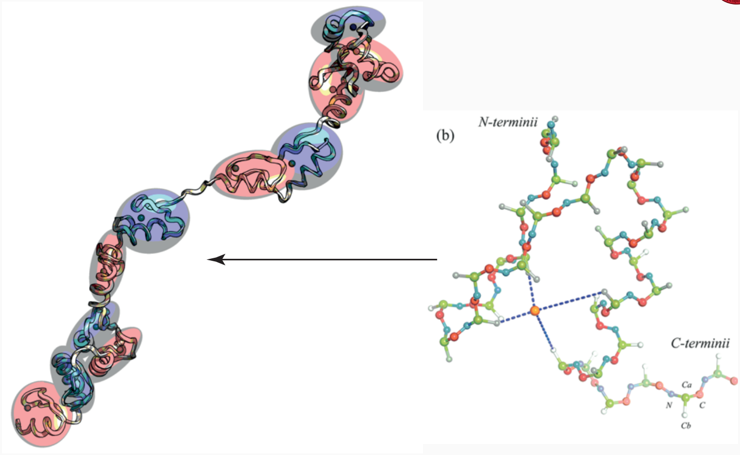


Figure 4: Shown is the model of a CTCF protein. Image taken from [13].

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) $p = (p^1, p^2, p^3)^T$.

General Remarks V

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

- 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

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

General Remarks VI

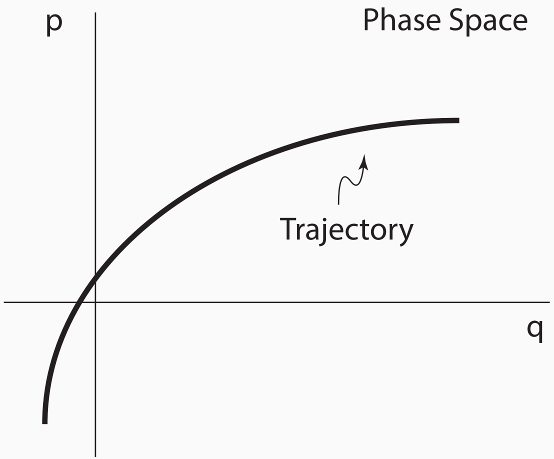


Figure 5: Phase space with generalized coordinate q and generalized momentum p.

General Remarks VII

More generally, we consider a state space **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 **S** or **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

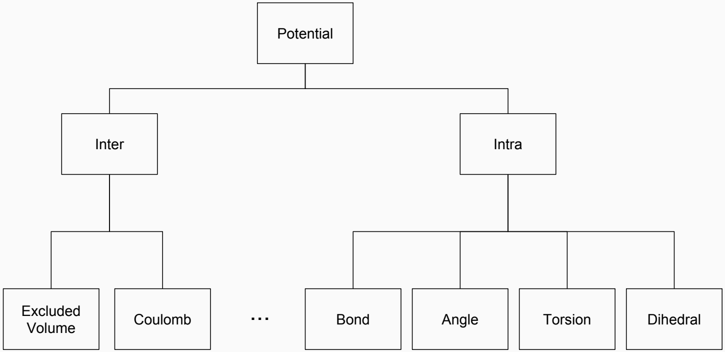


Figure 6: Shown is a possible classification scheme for potentials.

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

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

  is a two-body potential, 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

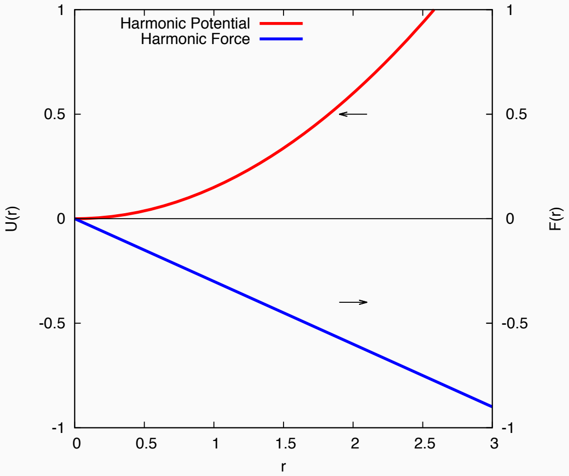


Figure 7: Harmonic potential as described in Equation 5 and the corresponding force.

Force Fields V

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 $r_{\mathbf{ub}}$ between the atoms (i, k) (Urey-Bradley term)

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

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

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

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.

Force Fields VI

- Non-bonded potential energy terms (Inter): Here we find potentials like 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

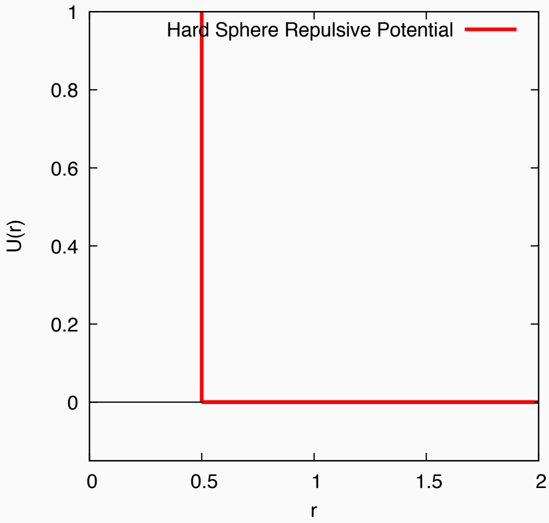


Figure 8: Excluded volume potential as described in Equation. 9.

Force Fields VIII

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

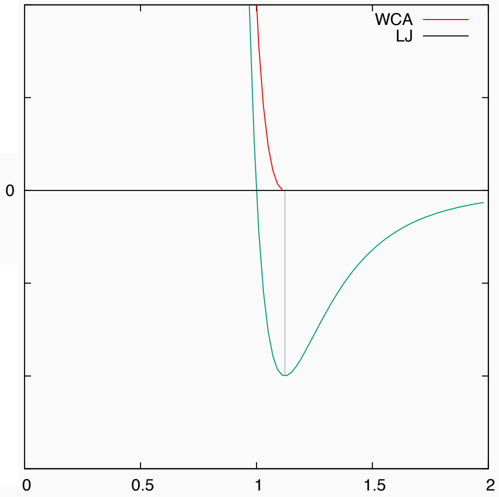


Figure 9: Weeks-Chandler-Andersen potential as well as the Lennard-Jones potential

Force Fields X

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, i.e. 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 1: Energy ϵ given in $1.38066 \times 10^{\mathbf{23}} J$ and length σ given in $10^{-\mathbf{1}} nm$

Force Fields XI

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

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

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

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

This removes the discontinuity at r_c. T5he truncation though does effect the some properties like the location of phase transition points.

Force Fields XII

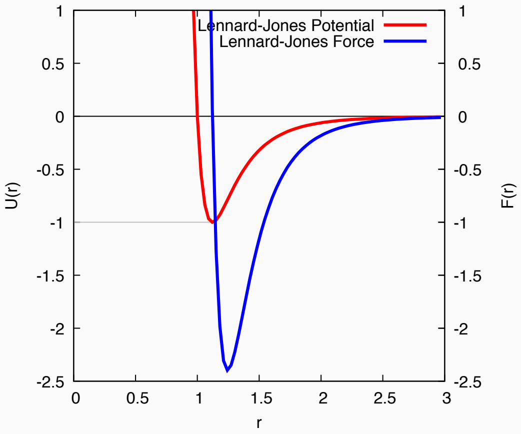


Figure 10: Lennard-Jones potential as described in Equation 12.

Force Fields XIII

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:

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

Force Fields XIV

The electrostatic potential or Coulomb 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

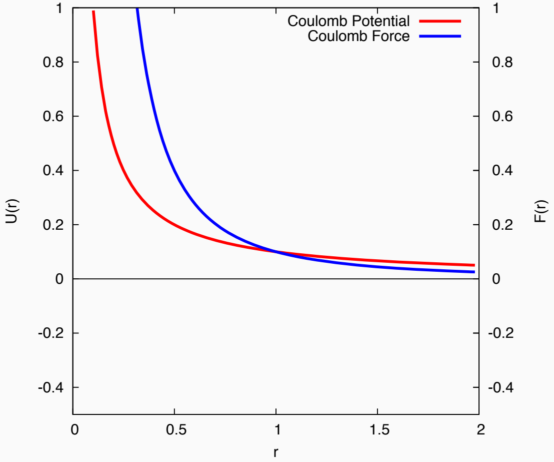


Figure 11: Lennard-Jones potential as described in Equation 16.

Force Fields XVI

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

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

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

Force Fields: Empirical Potential II

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

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 Equation 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

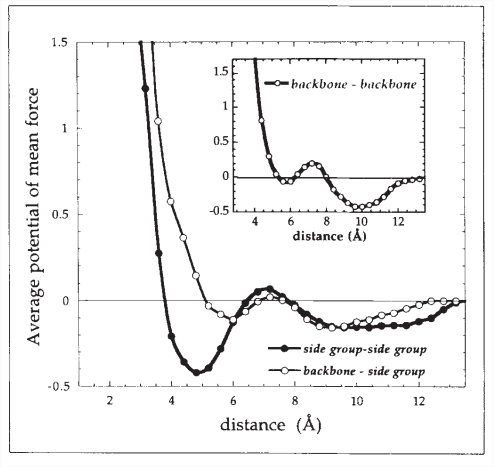


Figure 12: Empirical potential taken from *Structure-derived potentials and protein simulations* by Jernigan and Bahar [18].

Force Fields: Empirical Potential IV

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

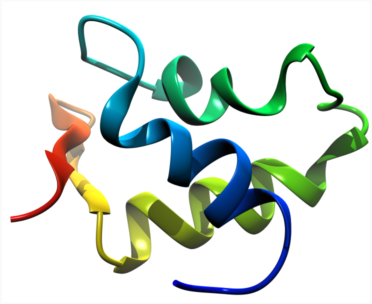
Example

The H2AX protein (shown in Figure 13, 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 Equation 21 using X-ray crystallographic data [19] on a large number of protein structures from the protein data bank.

Force Fields: Empirical Potential V



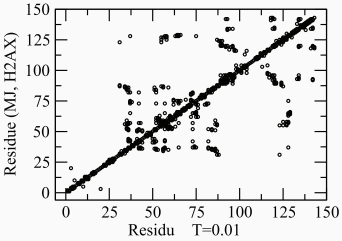


Figure 13: The left panel shows the ribbon images of H2AX. The right panel shows the contact probability between the residues at a specific temperature. Right panel image taken from Fritsche et. al. [20].

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.

```
/**
 * Force calculation
 */
void forceCalculation(float x[], float y[], float z[],
                  float fx[],float fy[],float fz[],
                  float rcoffs,float *potential,
                  float *virial, float side) {
  int i,j;
  float xi,yi,zi;
  float rd, rd2, rd3, rd4, rd6, rd7;
  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.

Nearest-Neighbor Search II

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)

Nearest-Neighbor Search III

- 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 IV

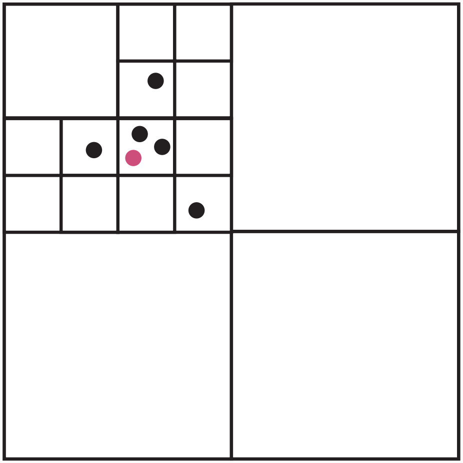


Figure 14: Example of a quad-tree space partitioning data structure. Shown is the space partitioning for the red particle, if only the black ones exist and that there is a finite cut-off of the potential consistent with the length scale of the last partitioning.

Nearest-Neighbor Search V

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

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Molecular Dynamics

Molecular Dynamics I

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

Molecular Dynamics II

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

$$\Gamma(t) = \binom{r}{p} \quad . \tag{26}$$

Then this trajectory obeys the equation of motion

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

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

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

Molecular Dynamics III

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

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

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

Molecular Dynamics IV

and with this

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

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

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

then the phase space volume needs to be conserved

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

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

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

where R_n gives the error involved in the approximation.

Using the forward $t+\Delta t$ and the backward difference $t-\Delta t$

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

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

If we add the two equations, we obtain

Molecular Dynamics: Discretization II

$$r(t+\Delta t) = 2r(t) - r(t-\Delta t) + \frac{F(t)}{2m}\Delta^2 t + R_4^* \ . \tag{40}$$

If we subtract the two equations we obtain

$$r(t+\Delta t) + r(t-\Delta t) = 2v(t)\Delta t + R_3^* \tag{41}$$

and hence an estimator for the velocity

$$v(t) = \frac{r(t+\Delta t) + r(t-\Delta t)}{2\Delta t} + R_2^* \ . \tag{42}$$

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

Algorithm 2 Molecular Dynamics Algorithm

```
1: $n = 0$
2: Specify positions $r_i^{-1}$ and $r_i^0$
3: Set $\Delta t$
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

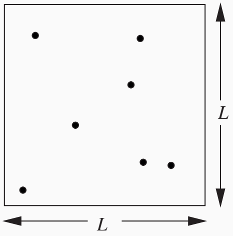
The computational volume is given by

$$\Omega = (a_1, b_1) \times \ldots \times (a_d, b_d) \subset \mathbb{R}^d \qquad (43)$$

where here for simplicity we assume

$$b_i - a_i = L, \quad a_i, b_i \in \mathbb{R} \qquad (44)$$

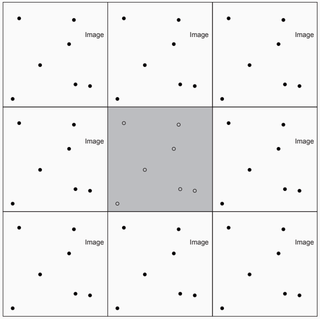
Of course, more complicated situations may be considered.



Molecular Dynamics: Boundary Conditions

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

$$\forall r = (x_1, \dots, x_d) \in \Omega, i = 1, \dots, d :$$
$$F(x_1, \dots, x_{i-1}, a_i, x_{i+1}, \dots, x_d) = F(x_1, \dots, x_{i-1}, b_i, x_{i+1}, \dots, x_d) \quad (45)$$



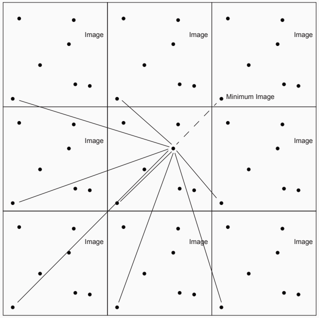
Molecular Dynamics: Boundary Conditions

```c
/**
 * 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

```c
/**
 * 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: Verlet Algorithm

The Verlet algorithm can be reformulated in such a way as to give a numerically more stable method [35, 36]. Define

$$z_i^n = \frac{r_i^{n+1} - r_i^n}{h} \tag{46}$$

The equations

$$\begin{aligned} r_i^n &= r_i^{n-1} + h z_i^{n-1} \\ z_i^n &= z_i^{n-1} + h F_i^n / m \end{aligned} \tag{47}$$

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

Molecular Dynamics

Algorithm 4 Molecular Dynamics Algorithm: NVE Velocity Form

```
1: $n = 1$
2: Set $\Delta t$
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:     $r_i^{n+1} = 2r_i^n - r_i^{n-1} + F_i^n h^2/m$
8:     Compute $F_i^{n+1}$
9:     $v_i^{n+1} = v_i^n + h(F_i^{n+1} + F_i^n)/m$
10:    $n = n + 1$
11: end while
```

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) [37–42]. It can be shown [43] 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 [43]

$$\Lambda = \frac{1}{2} \sum_i m v_i^2 = const \tag{48}$$

(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) [44]

$$\frac{1}{2} \sum_i m v_i^2 = \alpha t \tag{49}$$

Molecular Dynamics: Constant Temperature II

$$\beta = \left[(3N-4)k_B T_{\text{ref}} / \sum_i m v_i^2\right]^{1/2} \tag{50}$$

so that after the scaling step we have

$$\frac{1}{2}\sum_i m v_i^2 = \frac{1}{2}(3N-4)k_B T_{\text{ref}} \,. \tag{51}$$

Molecular Dynamics: Constant Temperature

Algorithm 5 Molecular Dynamics Algorithm: NVT

```
1: $n = 1$
2: Set $\Delta t$
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:     $r_i^{n+1} = r_i^n + hv_i^n + h^2 F_i^n / 2m$
8:     $v_i^{n+1} = v_i^n + h(F_i^{n+1} + F_i^n)/2m$
9:     Compute $\sum_i (v_i^{n+1})^2$ and the scaling factor $\beta$.
10:    Scale all velocities $v_i^{n+1} \rightarrow v_i^{n+1}\beta$
11:    $n = n + 1$
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

$$H = E + PV \ . \tag{52}$$

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 [45–47]. The Lagrangian now looks like

$$\mathcal{L}(r, \dot{r}, V, \dot{V}) = \frac{1}{2}\sum_i m\dot{r}_i^2 - U(r) + \frac{1}{2}M\dot{V} + P_E V \ . \tag{53}$$

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.,

Molecular Dynamics: Constant Pressure II

$$\rho_i = \frac{r_i}{V^{1/3}} = \frac{r_i}{L} \ . \tag{54}$$

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 (54) 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 (54) couples the dynamical variables $\mathbf{r}$ to the volume. Taking the first time derivative we obtain

$$\dot{r}_i = L\dot{\rho}_i + \rho\dot{L} \ . \tag{55}$$

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

$$\frac{p_i}{m} = L\dot{\rho}_i \tag{56}$$

as the momentum conjugate to ρ_i and the Lagrangian becomes

Molecular Dynamics: Constant Pressure III

$$\mathcal{L}(\rho, \dot{\rho}, V, \dot{V}) = \frac{1}{2} L^2 \sum_i m \dot{\rho}_i^2 - U(L\rho) + \frac{1}{2} M \dot{V}^2 - P_E V \ . \tag{57}$$

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 [48, 49]

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

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

Molecular Dynamics: Constant Pressure IV

$$\begin{aligned} \frac{d^2\dot{\rho}_i}{dt^2} &= \frac{F_i}{mL} - \frac{2}{3}\dot{\rho}_i\left(\frac{\dot{V}}{V}\right), \\ \frac{d^2 V}{dt^2} &= \frac{P - P_E}{M} \end{aligned} \tag{59}$$

with the pressure P computed from the virial

$$P = \frac{1}{3L}\left[\sum_i m\dot{\rho}_i^2 + \sum_{i<j} r_{ij}F_{ij}\right] . \tag{60}$$

Molecular Dynamics: Constant Pressure I

Algorithm 6 Molecular Dynamics Algorithm: NVT

```
1: $n = 1$
2: Set $\Delta t$
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

Langevin Dynamics I

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.

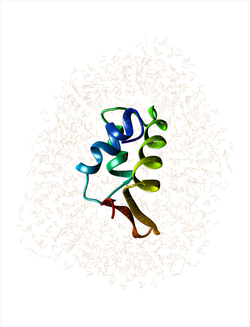


Figure 15: H2AX protein with solvants

Langevin Dynamics I

Thus we start off with a Langevin equation [50]

$$M\ddot{r} + \eta\dot{r} + \nabla V = \xi(t) \,, \tag{61}$$

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 (61) 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)

$$DD^T = \langle \xi\xi^T dt \rangle \,. \tag{62}$$

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

$$DD^T = 2k_B T\eta \,. \tag{63}$$

Langevin Dynamics II

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

$$\eta = \gamma M \tag{64}$$

with a damping constant γ. We then get

$$D = (2k_B T\gamma M)^{1/2}\ . \tag{65}$$

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

$$f = \frac{k_B T}{h} \exp\left(-\frac{\Delta E}{k_B T}\right) \tag{66}$$

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

Langevin Dynamics III

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

$$V(r) \approx V(r_{min}) + \frac{1}{2}(r - r_{min})^T \bar{V}(r - r_{min}) \tag{67}$$

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

$$M\ddot{r} + \bar{V}(r - r_{min}) = 0 \tag{68}$$

with the general solution

$$r = r_{min} + \sum_l u_l \exp(i\omega_l t) \; . \tag{69}$$

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

Monte Carlo Method

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
  - ...

Monte Carlo Method: Typical Parts I

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

$$\begin{aligned} P(A|B) &= \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\bar{A})P(\bar{A})} \quad (70) \\ &= \frac{P(B|A)P(A)}{P(B)} \,. \quad (71) \end{aligned}$$

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,

Random Numbers II

- fast algorithm.

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

$$x_{i+1} = G(x_i), \qquad x_0 = \text{initial value} \ , \tag{72}$$

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.

Random Numbers III

Linear Congruential Generators

A very simple generator is constructed using the modulo function.

$$G(x) = (xa + b) \bmod M \; . \tag{73}$$

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.

$$x_{n+1} = (x_n^{-1} a + b) \bmod M \; , \tag{74}$$

Random Numbers IV

where x_n^{-1} is the multiplicative inverse of x_n in the integers $\bmod\, m$ 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.
- 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

```c
/*------------------------------------------------------------------------- */
/*                         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 16: C source for a modulo random number generator

Random Numbers VI

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

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

Figure 17: 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

$$x_{n+1} = (x_{n-s} + x_{n-r} + c) \bmod M \tag{75}$$

- Subtract-with-carry

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

- Problems:

Random Numbers VII

- 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

$$x_n = x_{n-p} \otimes x_{n-q} \ , \quad p > q \ . \tag{77}$$

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.

Random Numbers VIII

```
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;
}
```

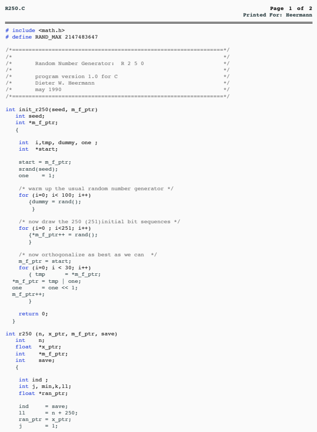
Figure 18: Fibonicci

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 $R250$. (Follow this link to access the code for the R250.c)

Random Numbers IX



R250.C

Page 1 of 2
Printed For: Heermann

```c
# 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 19: The shift bit register generator R250

Non Uniform Distributions I

- Let us turn 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 $x =< x > +\sigma s_n \sqrt{3.0/n}$.

Non Uniform Distributions II

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
```

Non Uniform Distributions III

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:     $Y = X$
5: else
6:     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: $x = (x^1, \dots, x^p), p > 2$

Algorithm 11 Generalized Gibbs Sampler Algorithm

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

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

Sampling from an Empirical Distribution

```python
from scipy.interpolate import interp1d
from statsmodels.distributions.empirical_distribution import ECDF
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages

# assign parameter values
seed = 4711
mu1, sigma1 = 100, 25
mu2, sigma2 = 150, 10
sample_size = 10000

np.random.seed(seed)
x1 = mu1 + sigma1 * np.random.randn(sample_size)
x2 = mu2 + sigma2 * np.random.randn(sample_size)
x = np.concatenate((x1,x2))

# create empirical distribution
ecdf = ECDF(x)
inv_cdf = interp1d(ecdf.y, ecdf.x, bounds_error=False, assume_sorted=False)

# Sample the empirical distribution
r  = np.random.uniform(0, 1, 1000)
y_sample = inv_cdf(r)
```

Sampling from an Empirical Distribution I

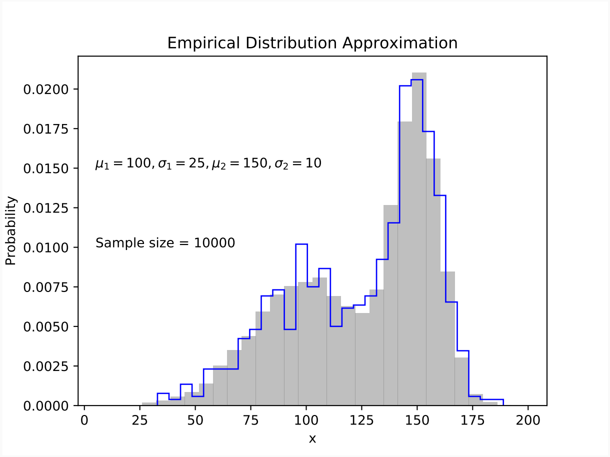


Figure 20: Sampling of an empirical distribution

Monte Carlo Sampling I

Consider a state space S with states $s \in S$. We can generate a trajectory

$$\tau = (s_1, \ldots, s_n) \tag{78}$$

of states from the state space S by, for example, uniformly drawing states from state space. Consider further a property $R(\tau)$ that depends on the trajectory of length n (see for example later section on machine learning where R is the return that we want to optimize. Of course, $n = 1$ is possible.

What we are interested is to calculate the expectation of R under the probability $s \sim \pi$

$$< R >= \mathbf{E}_{s\sim\pi}[R(s)] \; . \tag{79}$$

Markov Chain Monte Carlo I

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

$$P(x_n|x_0, ...x_{n-1}) = P(x_n|x_{n-1}) \, . \tag{80}$$

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

$$\frac{1}{n}\sum_{i=1}^{n} \phi(x_i) \to \int \phi(x)\pi(x)dx \tag{81}$$

and

$$X_i \sim \pi \, . \tag{82}$$

Examples:

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

Markov Chain Monte Carlo II

- 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 $P(x_{i+1}|x_i) =: P_{x_i, x_{i+1}}$ then

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

- $P_{x,y} \geq 0, \sum_y P_{xy} = 1$

Markov Chain Monte Carlo III

- 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!

To develop Monte Carlo methods we want:

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

$$\int P(x,y)\pi dx = \pi(y)\ . \tag{83}$$

- 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.

Markov Chain Monte Carlo IV

We require:

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

$$\forall x, y \in S \quad \exists n \in \mathbb{N}, n > 0 : (P^n)_{xy} > 0 \, . \tag{84}$$

- 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

$$\sum_{y \in S} \pi_y = 1, \quad \sum_{y \in S} \pi_y P_{yx} = \pi_x \, . \tag{85}$$

- 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. $\sum_{y \in S} \pi_y P_{yx} = \pi_x$

Markov Chain Monte Carlo V

2. P is irreducible and aperiodic

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

$$\pi_y P_{yx} = \pi_x P_{xy} \ . \tag{86}$$

which implies (1).

Metropolis-Hastings Monte Carlo I

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

$$\pi(x) = Z^{-1} \exp\{-\beta H(x)\} \; . \tag{87}$$

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:

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

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

$$a_{xy} := F\left(\frac{\pi_y \rho_{yx}^{(0)}}{\pi_x \rho_{xy}^{(0)}}\right) \; \forall x, y \in \mathcal{S} : x \neq y \tag{88}$$

then

$$\frac{a_{xy}}{a_{yx}} = \frac{F\left(\frac{\pi_y p_{yx}^{(\mathbf{0})}}{\pi_x p_{xy}^{(\mathbf{0})}}\right)}{F\left(\frac{\pi_x p_{xy}^{(\mathbf{0})}}{\pi_y p_{yx}^{(\mathbf{0})}}\right)} = \frac{F(z)}{F(1/z)} \overset{!}{=} z \tag{89}$$

with

$$z := \frac{\pi_x p_{xy}^{(0)}}{\pi_y p_{yx}^{(0)}} \; . \tag{90}$$

- The condition of detailed balance is satisfied if

$$\forall z : \frac{F(z)}{F(1/z)} \overset{!}{=} z \; . \tag{91}$$

Metropolis-Hastings Monte Carlo IV

- Often used choices are

$$F(z) = \min(z, 1) \tag{92}$$

  and

$$F(z) = \frac{z}{1+z} \; . \tag{93}$$

- Note that the proposals of states need not to be symmetric. As a further point we note that is was proven[51] 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

Algorithm 12 Metropolis-Hastings Monte Carlo Algorithm [1, 52]

```
for i=0; i < mcsmax do
    sample a new state $x$;
    set $y = x_i$;
    sample a uniform random number $r$ from $(0, 1)$;
    if $r \leq \min\{\frac{p^{\mathbf{0}}_{xy}\pi_x}{p^{\mathbf{0}}_{x}y\pi_y}, 1\}$ then
        $x_{i+1} = x$;
    else
        $x_{i+1} = y$;
    end if
end for
```

Error Analysis I

- 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 $A_i, i = 1, ..., n$:

$$\left\langle (\delta A)^2 \right\rangle = \left\langle \left[n^{-1} \sum_{i=1}^{n} (A_i - \langle A \rangle)^2 \right] \right\rangle \; . \qquad (94)$$

- In terms of the autocorrelation function for the observable A

$$\phi_A(t) = \frac{\langle A(0)A(t) \rangle - \langle A \rangle^2}{\langle A^2 \rangle - \langle A \rangle^2} \; . \qquad (95)$$

  We define two characteristic correlation times.

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

$$\Phi_A(t) \propto \exp\left(-\frac{t}{\tau_{A,exp}} \right) \; . \qquad (96)$$

  - 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.

Error Analysis II

- **Integrated autocorrelation time**

$$\tau_A^{int} = \int_0^\infty \phi_A(t)dt \ . \tag{97}$$

- We can rewrite the statistical error as

$$\left\langle (\delta A)^2 \right\rangle \cong \frac{2\tau_A}{n\delta t}\left[\left\langle A^2 \right\rangle - \langle A \rangle^2\right] \ , \tag{98}$$

  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.

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.

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.

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

$$\begin{aligned} \frac{dx}{dt} &= \frac{\partial \mathcal{H}}{\partial p} \\ \frac{dp}{dt} &= -\frac{\partial \mathcal{H}}{\partial x} \,. \end{aligned} \tag{99}$$

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

$$p_C(x \to x')dx' = p_C(p)dp \,. \tag{100}$$

Hamiltonian Monte Carlo IV

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

$$p_C(p) \propto e^{-\beta \sum_{j=1}^{N} \frac{p_j^2}{2m}} . \tag{101}$$

- Thus

$$P_A((x,p) \to g^{\delta t}(x,p)) = \min\{1, e^{-\beta\delta\mathcal{H}}\} , \tag{102}$$

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

$$e^{-\mathcal{H}(x,p)} \min\{1, e^{-\delta\mathcal{H}}\} = e^{-\mathcal{H}(g^{\delta t}(x,p))} \min\{e^{\delta\mathcal{H}}, 1\} \tag{103}$$

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

$$g^{-\delta t} \circ g^{\delta t} = id \tag{104}$$

Hamiltonian Monte Carlo V

- and **area-preserving**

$$\det \frac{\partial g^{\delta t}(x,p)}{\partial(x,p)} = 1 \,, \tag{105}$$

  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.

Hamiltonian Monte Carlo VI

- Therefore neither $p(x)$ nor any ensemble averages depend on the step size δt chosen.
- However, the average acceptance probability $\langle P_A \rangle$, because of (102), depends on the average discretization error $\langle \delta\mathcal{H} \rangle$ and hence does depend on δt.
- It can be shown that for $(\varrho, T) \neq (\varrho_c, T_c)$

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

  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

$$\langle e^{-\beta\delta\mathcal{H}} \rangle = 1 \ . \tag{106}$$

- Equation (106) can be expanded into cumulants

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

- 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.

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

$$\langle \delta\mathcal{H} \rangle = \frac{\beta}{2} \langle (\delta\mathcal{H} - \langle \delta\mathcal{H} \rangle)^2 \rangle \, . \tag{107}$$

- From (102) and (107) one has in this case

$$\begin{aligned} \langle P_A \rangle &= \frac{1}{\sqrt{2\pi \langle (\delta\mathcal{H} - \langle \delta\mathcal{H} \rangle)^2 \rangle}} \int_{-\infty}^{\infty} dt \, \min\{1, e^{-\beta t}\} e^{-\frac{(t - \langle \delta\mathcal{H} \rangle)^2}{2\langle (\delta\mathcal{H} - \langle \delta\mathcal{H} \rangle)^2 \rangle}} \\ &= \mathrm{erfc}(\frac{1}{2}\sqrt{\beta \langle \delta\mathcal{H} \rangle}) \, . \end{aligned} \tag{108}$$

- The square root in (108) is always well defined since (106) implies

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

- Equality holds in the limit $\delta t \to 0$, where energy is conserved exactly and $\langle P_A \rangle = 1$.

Hamiltonian Monte Carlo VIII

- 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.
- 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 (99) any time-reversible and area-preserving discrete mapping can be used to propagate the system through phase space.

Rejection-Free Monte Carlo I

So far, we have been using the rejection Monte Carlo algorithms. To remind us, the algorithms proceeds from state x to possible state x' as outlined in Algorithm 13.

Algorithm 13 Accept/Reject Monte Carlo Algorithm

```
1: Choose initial state $x$
2: for n-of-samples do
3:     Select a new state $x'$
4:     With probability $p$ accept, i.e. set $x = x'$
5:     With probability $(1 - p)$, $x'$ is rejected
6: end for
```

- The probability will depend on some change induced by the state change
- Construct algorithm that does not involve accept/reject, i.e. always accept
- Thus, the methods will be (synchonously or asynchronously) event-driven [53] (see Algorithm 14).

Rejection-Free Monte Carlo II

Algorithm 14 Event-Driven Algorithm

```
1: for n-of-samples do
2:     Identify all possible events
3:     Identify the event with the smallest time stamp $\Delta t$
4:     Set time $t = t + \Delta t$
5: end for
```

- Methods that rely on rates between states thus the sequence that ultimately will be generated evolves in time (see Figure 21).
- However, not as in the previous chapters systolically, driven by a constant increment in time, but by leaps of various length in time.
- This also opens up the possibility to make rigorous the notion of time in Monte Carlo methods.

Rejection-Free Monte Carlo III

.

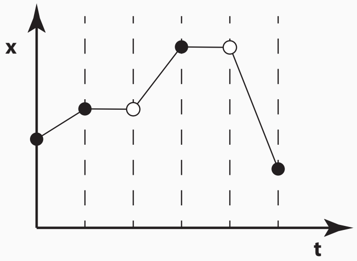


Figure 21: The lhs panel shows the typical systolic propagation of time for example in the Metropolis Monte Carlo. Sometimes new state proposals are rejected (circles) and the previous state is the new state. The rhs panel depicts the leaps in time that are made to achieve a rejection free algorithm.

Rejection-Free Monte Carlo IV

- Consider a state space Ω and a sequence $\{x_{t_k} \in \Omega\}$ of states from the state space. Often we simply write i or j etc. to label the states.
- Here we assume $t_0 < t_1 < \cdots < t_k < \cdots$.
- So far we have had $\Delta t = t_k - t_{k-1}$ constant, i.e. the system was moved forward in time by a constant stride.
- Furthermore, for two states (x_{k-1}, x_k) we have the Markov property so that the sequence $\{x_{t_k} \in \Omega\}$ is a Markov chain.
- Let us now look at continuous-time Markov chains $\{x_t \in \Omega | t \in \mathbb{R}, t \geq 0\}$. For the chain to be a continuous-time Markov chain the following condition needs to apply

Rejection-Free Monte Carlo V

$$\mathbb{P}(x(t+\tau)=j|x(\tau)=i, x(u)=k, 0\leq u\leq \tau) = \mathbb{P}(x(t+\tau)=j|x(\tau)=i)\ . \quad (109)$$

Define

$$p_{ij}(t) := \mathbb{P}(x(t+\tau)=j|x(\tau)=i) = \mathbb{P}(x(t)=j|x(0)=i) \quad (110)$$

and for any state i we have (for N possible states)

$$\sum_{j=1}^{N} p_{ij}(t) = 1\ . \quad (111)$$

Let $P(0) = \lim_{t\searrow 0} P(t) = I$ be the initial condition. Then the matrix R defined by

$$\lim_{h\searrow 0} \frac{P(h)-I}{h} = P'(0) = R \quad (112)$$

is the infinitesimal generator of the continuous-time Markov process with rate r_{ij}

Rejection-Free Monte Carlo VI

$$\sum_{j=1, j\neq i} r_{ij} = -r_{ii} \tag{113}$$

and

$$r_{ij} = \lim_{h\searrow 0} \frac{p_{ij}(h)}{h} \geq 0 \text{ and } r_{ii} \leq 0\ . \tag{114}$$

Define $r_i := -r_{ii} > 0$ to be the rate corresponding to state i. Given R, then for all $t \geq 0$

$$P'(t) = RP(t)\ . \tag{115}$$

and

$$P(t) = Re^{-Rt} \tag{116}$$

as the first-passage-time distribution and further

Rejection-Free Monte Carlo VII

$$p_{ij} = r_{ij} e^{-r_{ij} t} \ . \tag{117}$$

Since we are talking about first-passage only, only one of the possibilities can happen.

Thus, rather than focusing on the transition probabilities (c.f. Figure 22) as we have in the previous chapters, we can focus on the rates between states opening up to models where there is no Hamiltonian. Even more so, the rates themselves may depend on time. If they do not then the Markov process is stationary.

Rejection-Free Monte Carlo VIII

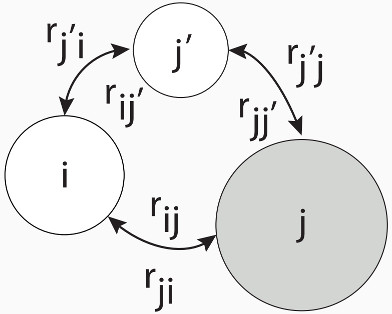


Figure 22: The figure shows the general situation where the circles denote states in state space that belong to the same state *i*.

Rejection-Free Monte Carlo IX

- Let n_i denote the population of state i.
- Given that we are dealing with a thermal system then n_i must be proportional to $\exp\{-F(i)/k_B T\}$.
- In equilibrium if we have detailed balance then

$$n_i r_{ij} = n_j r_{ji} \ . \tag{118}$$

- Thus, what is needed for a model is a state space Ω and a set of rates R, i.e. (Ω, R).
- This can for example be a set of chemical reactions with the corresponding rates.
- We envisage that at any given time for a state i not all states j are accessible.
- Thus it is convenient to relabel the currently accessible states with a new label.
- We arrive at a list of N possible events and a list with corresponding rates

$$\{E_n \in \Omega\} \qquad \text{with } n = 1, \ldots, N \tag{119}$$

$$\{r_n\} \qquad \text{with } n = 1, \ldots, N \ . \tag{120}$$

Rejection-Free Monte Carlo X

- From a computational point of view, it is immediately apparent that what is needed is a well performing bookkeeping algorithm for the events and the rates as they may change after an event has been chosen.
- Let us consider this for the Ising model.
- It was pointed out by Bortz, Kalos and Lebowitz [54] that the probability of accepting new configurations in the Ising model is very low in same cases.
- Consider the case when the temperature is low.
- Then two spins will have likely the same orientation and thus a reversal has very low probability.
- Thus, out of the N attempts only a very low fraction will result in changes. Suppose only attempts are made that are successful.
- For this the rates r_{ij} from state i to j need to be known a priori.

Rejection-Free Monte Carlo XI

- In the Ising case we know transition rates among states a priori. For the two-dimensional Ising model

$$\mathcal{H} = -J \sum_{<i,j>} S_i S_j \quad S_i = \pm 1 \tag{121}$$

with its spin-up spin-down symmetry we have the situations as shown in Table 2.

Spin	$\uparrow (+1)$					$\downarrow (-1)$				
Neighbours	4	3	2	1	0	0	1	2	3	4
Class	1	2	3	4	5	6	7	8	9	10

Table 2: Classes for the kinetic Monte Carlo (n-fold way) algorithm proposed by Bortz et. al. [54]. Corresponding to each class i there is a probability p_i.

- Altogether we have ten possible states, depending on the number of neighbors the central spin is surround by.
- Each of these states we assign a class.

Rejection-Free Monte Carlo XII

- Assume further that the transition probability between states is given by

$$p = \frac{x}{1+x} \quad \text{with} \quad x = \exp\{-\Delta\mathcal{H}/k_B T\} \ , \tag{122}$$

  then all possible transitions $r_{i,}$ are given.
- One possibility is to use Equation 117 to draw time increments for the event to happen.
- This algorithm is known as the first-reaction method [55].
- For this we generate a random number $\rho \in (0,1)$ and compute

$$t_{ij} = -r_{ij}^{-1} \ln(\rho) \ . \tag{123}$$

- Thus, for every state change we know the probability and the first passage times.
- What remains to do is to identify the state change $i \to j$. For this we select the reaction coordinate that comes first in time

$$\Delta t = \min_{ij} t_{ij} \ . \tag{124}$$

- Then this state change is performed and time advances (see Algorithm 15)

$$t = t + \Delta t \ . \tag{125}$$

Rejection-Free Monte Carlo XIII

.

Algorithm 15 First Reaction Monte Carlo Algorithm

```
1: Initial time $t = 0$
2: Choose initial state $i$
3: for n-of-samples do
4:     Set up list of transition rates $r_{ij}$ (size $N$)
5:     Generate $N$ random numbers $\rho_j$ from a uniform distribution on $(0, 1]$
6:     $t_{ij} = r_{ij}^{-1} \ln(\rho_j^{-1})$
7:     $\Delta t = \min_{ij} t_{ij}$
8:     Carry out event $i \rightarrow j$ that is minimum
9:     Update $t = t + \Delta t$
10:    $i \leftarrow j$
11: end for
```

Rejection-Free Monte Carlo XIV

- Hence, we only perform those state changes that actually occur.
- This is in contrast to the procedure that we have developed so far.
- Note that this algorithm uses $\mathcal{O}(N)$ to build the list of transition rates, $\mathcal{O}(N)$ for the number of random numbers and $\mathcal{O}(N)$ to determine the minimum time.
- The obvious difference to the Metropolis Monte Carlo algorithm is that time does not advance in fixed increments but rather leaps in non-constant strides.
- It must further be pointed out that the transition probabilities change at every step.
- Indeed, one of the key features is that the distribution of rates is coupled to the state space [56] and can change.
- For the Ising case there is no such problem.
- This can be seen when we consider the two-dimensional case shown in Diagram 109.

$\uparrow$	$\uparrow$	$\downarrow$
$\uparrow$	$\downarrow$	$\uparrow$
$\uparrow$	$\uparrow$	$\uparrow$

- This translates into the class scheme from Table 2.

Rejection-Free Monte Carlo XV

2	3	10
2	4	3
1	2	2

- A spin flip can change the transition probability and with it the class.
- The starting point is a choice of a state the system is started in.
- This determines the possible states that the system can transition into and the corresponding rates r_{ij}.
- The next step is to compute the sum over all the possible rates from i to j, i.e. all possible reaction paths.
- The next step then is to pick one of the possible reaction paths with equal probability followed by advancing the time as shown in Algorithm 16.

Rejection-Free Monte Carlo XVI

.

Algorithm 16 Kinetic Monte Carlo Algorithm

```
1: Initial time $t = 0$
2: Choose initial state $i$ at random
3: for n-of-samples do
4:     Set up list of transition rates $r_{ij}$ (size $N$)
5:     Compute $R_{i,j} = \sum_{k=1}^{i} r_{ik}$ for $j = 1, ..., N$
6:     Compute $R_i = R_{i,N}$
7:     Generate $\rho$ from a uniform distribution on $(0, 1]$
8:     Choose $i$ such that $R_{i,j-1} < \rho R_i \leq R_{ij}$
9:     Carry out event $j$
10:    Update $i \to j$
11:    Generate $\rho$ from a uniform distribution on $(0, 1]$
12:    $\Delta t = R_i^{-1} \ln(\rho^{-1})$
13:    $t = t + \Delta t$
14: end for
```

Rejection-Free Monte Carlo XVII

- The beauty of the kinetic Monte Carlo Method is that it easily generalizes to arbitrary states and reactions.
- This is why has been used for many condensed matter systems [57–60] with certain refinements [61–65] and coupled to molecular dynamics [66].
- Further developments are the coarse-grained kinetic Monte Carlo [67, 68] and the first-passage kinetic Monte Carlo algorithm [69].
- Let us return to the initial example of the Ising Model.
- Let n_i be the number of spins in class i (see Table 2), then we need to choose the relative weights $n_i p_i$ according to Algorithm 16 and once a class has been chosen a spin in that class is chosen with probability $1/n_i$.
- Fichthorn and Weinberg [70] showed that under the condition of detailed balance and the effective independence of the events, the Algorithm 16 yields a Poisson process and that static and dynamic properties are consistent with the Hamiltonian dynamics [71].
- However, detailed balance is not necessary!
- As we will see later, the kinetic Monte Carlo method is used for non-equilibrium situation and where detailed balance is not fulfilled but global balance is achieved.

Rejection-Free Monte Carlo XVIII

- Note that number of operation, i.e. the complexity is $O(N)$. Makysm [72] showed that using a binning method and recursive search trees, the complexity can be brought down to $O(\log_2 N)$ [62].
- For completeness, even though we are in the chapter on rejection-free Monte Carlo, here is a rejection algorithm for the model pair (Ω, Q).

Rejection-Free Monte Carlo XIX

.

Algorithm 17 Rejection Kinetic Monte Carlo Algorithm

```
1: for n-of-samples do
2:     Set up list of transition rates $r_n$ (size $N$)
3:     Compute an estimator for the sum of rates $\bar{r}$
4:     while state not selected do
5:         Generate $\rho$ from a uniform distribution on $[0, N)$
6:         Compute $n = (Int)(\rho) + 1$
7:         Select $n$ if $n - \rho < r_n/\bar{r}$
8:     end while
9:     $n$ is new state
10: end for
```

Rejection-Free Monte Carlo XX

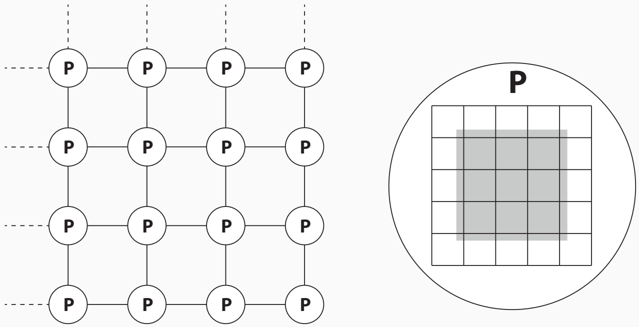


Figure 23: For the simplest parallel kinetic Monte Carlo algorithm, we assume that the topology for the processors is that of a lattice (for simplicity here a simple square lattice with the processors (P) at the nodes of the lattice) with possible periodic boundary condition (dashed lines). The solid lines represent bi-directional communiation channels (lhs). The rhs panel shows the possibility that a processor has been assigned more than site, say for the 2-D Ising model, L/l lattice sites. The gray shaded area is the part where no communication between the processors is needed for a decision to flip a spin.

Rejection-Free Monte Carlo XXI

- For the Ising model, Lubachevsky [73] has succeeded to parallelize the Monte Carlo algorithm based on the ideas put forward in the more general context by Chandy and J. Misra [74, 75].
- He formulated the algorithm as a distributed discrete-event system.
- Various methods have been designed specifically with lattice models at focus [76–78].
- Also the scaling properties of these type of algorithms have been investigated [79, 80] associating the development of the individual time increments at the individual processors with time increments corresponding to depositions and thus identifying this with surface growth (Kardar-Parisi-Zhang equation [81]).
- The parallelization of the τ-leap has been done by Xu et.al. [82] and for the presence of long-range interactions see [83].
- Also much effort has gone into parallelization of the Gillespie ansatz, for example, Komarov [84].
- The key problem in the parallelization is to avoid event time incompatibilities with communications.

Rejection-Free Monte Carlo XXII

- The solution that Lubachevsky [73] has put forward is the strict synchronization (c.f. Figure 23 and Algorithm 18).
- The problem is solved in this algorithm using a global synchronization at the slight expense of efficiency.
- The algorithm presented here is aiming at the above outlined Ising situation.
- We assume two functions $\text{nextState}(i, t_i, \text{neighbours}(i))$ which calls upon the neighbor processors for the corresponding states s_j and $\text{nextTime}(t_i)$ delivers the next time.

Rejection-Free Monte Carlo XXIII

.

Algorithm 18 Lubachevsky Parallel Monte Carlo Algorithm

```
1:  $s' = s_i$
2:  $t' = t_i$
3:  for n-of-samples do
4:      if $t_i \leq \min_{j \in \text{neighbours}(i)} t_{ij}$ then
5:          $s' = \text{nextState}(i, t_i, \text{neighbours}(i))$
6:          $t' = \text{nextTime}(t_i)$
7:          Global synchronize
8:          $t = t'$
9:          $s' = s$
10:         Global synchronize
11:     else
12:         Global synchronize
13:         Global synchronize
14:     end if
15: end for
```

Rejection-Free Monte Carlo XXIV

- Assume that in the Ising case the lattice is much larger that the number of processor and that there are L/l lattice sites per processors (i.e. $L \times L$ lattice with $l \times l$ blocks).
- There are now interior and boundary sites to be handled by the Algorithm 18. Korniss et. al. [85] have argued that the synchronization steps in the algorithm are not necessary.
- If the same random number generator runs on each of the processors with the same initial seed, they argue that the probability of equal-time nearest-neighbor updates is of measure zero.
- Thus they suggest to treat the interior spins (gray shaded region in Figure 23) like regular spins and use

$$p = \min\{1, \exp(-\Delta H/kT)\} \qquad (126)$$

  with $\Delta t = -\ln(\rho)$ (ρ the random number) advancement in time.
- For the boundary spins the criterion in Algorithm 18 is applied.
- To ensure freedom of a deadlock a barrier is used for the boundary spins with a *wait until* the local time t becomes less than or equal to the same quantity for the neighbours.

Rejection-Free Monte Carlo XXV

- For the kinetic Monte Carlo algorithm for the Ising model Lubachevsky [73] introduced an additional class N_b on top of the 10 classes for the boundary spins. Assume as above that the linear system size is L and that there are $4l$ boundary spins per processor.
- Then $N_b = 4(l-1)$.
- The basic idea is to use the original Monte Carlo, for example Metropolis Monte Carlo, for the boundary spins and for the interior spins the kinetic Monte Carlo.
- Thus the algorithms proceeds as outlined in Algorithm 19.
- For this we augment the 10 classes with the additional class N_b.

Rejection-Free Monte Carlo XXVI

.

Algorithm 19 Lubachevsky Parallel Kinetic Monte Carlo Algorithm

```
1:  Initial time $t = 0$
2:  for n-of-samples do
3:      Set up list of transition rates $r_i = n_i p_i$ plus $N_b$
4:      Compute $R_k = \sum_{i=1}^{i} r_i$
5:      Generate $\rho$ from a uniform distribution on $(0, 1]$
6:      Choose $i$ such that $R_i < \rho R_i \leq R_i$
7:      Choose a spin with equal probability within the class $i$
8:      if  spin is within the interior then
9:          Flip the spin
10:     else
11:         Wait until the local simulated time $\leq$ neighbour processor
12:         Apply Metropolis Monte Carlo to the spin
13:     end if
14:     Update time
15: end for
```

Rejection-Free Monte Carlo XXVII

- A slightly different approach has been taken by Martinez [86] by a synchronous time decomposition of the master equation (synchronous parallel kMC method (spkMC)).
- The basic idea is to create so called null events advancing the internal clock of each processor. This is done without altering the stochastic trajectory of the system.
- Further developments have been done specifically for the reaction-diffusion problems (see [87] and references therein).
- Due to the success of other parallelization algorithms on GPUs, an algorithm was proposed by Jimenez and Ortiz [88], Klingbeil [89] and Agostino et. al. [90].
- Also discrete-event approaches have been developed [91] specifically for the Gillespie ansatz.

Lifting I

- As the name of this section suggests we augment the state space Ω with one or more additional variables.
- Let us first examine this idea for the Ising model in the case of conserved energy.
- Assume that we add the extra variable or degree of freedom to the Hamiltonian [92] (121)

$$\mathcal{H}' = e - \sum_{<i,j>} S_i S_j \quad S_i = \pm 1 \tag{127}$$

with $\Omega' = \mathbb{N} \times \Omega$.

- The extra variable e allows to lift the system out of the otherwise constraint hyperspace of constant energy.
- Set e to an appropriate value according to the initial energy.
- We can construct a Markov chain by choosing a spin at site ν at random.
- We change the spin direction at site ν to obtain $\Delta\mathcal{H}$ for the energy change in the Ising Hamiltonian.
- If we loose energy, then we transfer the energy to e and accept the change.

Lifting II

- If we would gain energy, then we accept the change under the condition that e has enough energy.
- Let us now look at the more general case. Chen et. al. and others [93–97] constructed a non-reversible Markov chain Monte Carlo Method (Lifted Metropolis-Hastings) as for example also in the (Hamiltonian) Hybrid Monte [98] (see also for the Bouncy Particle Sampler method [99]).
- The effect of this lifting is a reduced mixing time of the Markov chain (at best reduced by the square root of the original time).
- So far we almost always used the detailed balance condition for the transition probability W and the invariant distribution p which we want to obtain from a Markov chain

$$p(x)W(x,x') = p(x')W(x',x) \text{ for all } x,x' \in \Omega \, . \tag{128}$$

- This is not a necessary but sufficient a condition for the transition probability.

Lifting III

- One of possible solutions to Eq. 128 is the Metropolis Hastings transition probability

$$W(x,x') = q(x|x') \min\left\{1, \frac{p(x')q(x|x')}{q(x'|x)p(x)}\right\} \tag{129}$$

  with the propositional probability q. The Hybrid Monte Carlo Method [98] has made use of this propositional probability.
- Consider the global balance condition for the transition probability W [100]

$$\int p(x)W(x,x')dx = \int p(x')W(x',x)dx' \tag{130}$$

  which we need to really to fulfill and the constraint

$$W(x,x')W(x',x) = 0 \text{ for all } x, x' \in \Omega\,. \tag{131}$$

- W's that fulfill criterion 130 and criterion 131 are said to check a maximal global balance condition [101, 102].

Lifting IV

- Following the idea of adding additional degrees of freedom, we augment the system by an auxiliary variable e. Thus for the distribution p this results in

$$p(x, e) = p(x)p(e) \tag{132}$$

  and for the above example (127) this would be

$$p(e) \propto \exp\{-\beta e\} \; . \tag{133}$$

  and fix the propositional probability as

$$q(x', e|x, e) = \begin{cases} 1, & \text{if } x' = x + e\Delta s \\ 0, & \text{otherwise} \end{cases} \tag{134}$$

  where the statement $x' = x + e\Delta s$ is meant to express that x' and x should not differ too much.
- Thus we updated the state in the direction given by e. This is continued until rejection occurs.
- Then we choose a new e' and continue with (x', e') which lifts the rejection into the lifting space rendering the entire method rejection-free.
- The probability for the choice of e' is based on the condition 130.

Event-Chain Monte Carlo I

- We will extend the rejection-free Monte Carlo simulation methods by considering irreversible Markov chains drawing on idea by Peters [103] and the concept of lifting [94].
- These methods have been successfully developed for the problem of melting in two dimensions [104–107].
- Extensions have been derived for discrete-variable models [108], classical continuous spin models [109, 110] and further generalized to rejection-free global-balance algorithms [111] and the forward event-chain Monte Carlo algorithm [100].
- Here we follow [100] in the exposition of the algorithm.
- The goal is to use the ideas of lifting developed in the previous section to develop a rejection-free Monte Carlo algorithm.
- We use the extra variable e to suggest a new state.
- Rather than using an except/reject on this, we choose a time for the new event to happen and sample all the state in a chain along the way, until we have reached the transition time. We then choose a new e value and continue.
- To sample the time Δs we go about as in (117) and (123).

Event-Chain Monte Carlo II

- For ease presentation we follow the mechanistic language and assume an energy function $E(x)$ and consider e a velocity (see also Hybrid Monte Carlo [98]).
- Thus in Eq. 134 we are looking for displacements in space controlled by the time Δs and the velocity e.
- In Eq (134) we have made a choice for the propositional probability.
- With the notation $[a]^+ = \max\{0, a\}$ and Metropolis choice of transition probability (129) we have

$$W(x, x') = \min\{1, \exp\{-\Delta E(x)e\}\} = \exp\{-[\Delta E(x)e]^+\} \ . \tag{135}$$

- To determine the transition time we add up all the moves until we have reached the event time

$$\Delta E^*(\Delta s) = \int_0^{\Delta s} [\nabla E(x + se)e]^+ ds \tag{136}$$

  and find the time Δs by solving the equation

$$\Delta E^*(\Delta s) = \log(\rho) \tag{137}$$

  where $\rho \in (0, 1]$ is a uniform random number. It rests to choose the transition probability for e. Here Michel and Senecal [100] suggest

Event-Chain Monte Carlo III

$$p(e' \to e) = \delta(e' + e)\ . \qquad (138)$$

- In Algorithm 20 the full algorithm is exposed (for parallelization for example for dense hard sphere and polymer systems see [112]).

Event-Chain Monte Carlo IV

Algorithm 20 Event Chain Monte Carlo Algorithm [100]

```
1: Initial state $x' = x_0$
2: for n-of-samples do
3:     Set current event chain length $l_c = l$
4:     Set random direction $e$
5:     while True do
6:         Set initial sample $x = x'$
7:         Compute $\Delta E^* = -\log(\rho)$, $\rho$ from a uniform distribution on $(0, 1]$
8:         Compute $\Delta s$
9:         if $l_c < \Delta s$ then
10:            Compute $x' = x + l_c e$
11:            Set sample $x^k = x'$
12:            Break
13:        else
14:            Compute $x' = x + \Delta s e$
15:            Update chain length $l_c = l_c - \Delta s$
16:            Update direction $-e$
17:        end if
18:    end while
19: end for
```

Excercises

Excercises I

Exercise 1: **Hard Disk Potential**
Implement the potential:

$$u_{ij}(r) = \begin{cases} 0 & \text{if} \quad \mathbf{r} > r_c \\ \infty & \text{otherwise} \end{cases} \tag{139}$$

Exercise 2: **Lorentz-Berelot combining rule**
Implement the potential:

$$\sigma_{\alpha\beta} = \frac{\sigma_{\alpha\alpha} + \sigma\beta\beta}{2} \tag{140}$$

$$\epsilon_{\alpha\beta} = \sqrt{\epsilon_{\alpha\alpha} + \epsilon_{\beta\beta}} \tag{141}$$

Exercise 3: **Morse Potenial**
Implement the potential:

$$u_{ij}(r) = k\left(1 - e^{a(r - r_0)}\right)^2 \tag{142}$$

Excercises II

Exercise 4: **Mie-Potential**

Implement the Mie Potential [114]:

$$U(r) = A/r^m - B/r^n \tag{143}$$

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 [115] with the modification of Forsythe [116].

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.

Excercises III

Algorithm 21 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 numbers 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 favorite random number generator and perform the permutation test.

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

Exercise 8: **Weibul distribution**

Construct an algorithm generating the Weibull distribution having the cumulative distribution function with $\lambda > 0$ and $\alpha > 0$

$$F(x) = 1 - e^{-\lambda x^{\alpha}} \quad , x \geq 0 \tag{144}$$

and probability density

$$f(x) = \lambda \alpha x^{\alpha - 1} e^{-(\lambda x)^{\alpha}} \ . \tag{145}$$

Bibliography

Bibliography I

References

[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.

Bibliography II

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

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

[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.

Bibliography III

[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.

[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.

Bibliography IV

[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.

[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.

Bibliography V

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

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

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

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

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

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

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

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

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

Bibliography VI

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

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

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

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

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

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

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

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

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

[53] David G. Kendall. Random fluctuations in the age-distribution of a population whose development is controlled by the simple "birth-and-death" process. 12(2):278–285, 1950.

[54] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz. A new algorithm for monte carlo simulation of ising spin systems. *Journal of Computational Physics*, 17(1):10–18, 1975.

Bibliography VII

[55] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[56] Tim P. Schulze. Efficient kinetic monte carlo simulation. *J. Comput. Phys. 0021-9991*, 227(4):2455–2462, 2008.

[57] W M Young and E W Elcock. Monte carlo studies of vacancy migration in binary ordered alloys: I. *Proceedings of the Physical Society*, 89(3):735, 1966.

[58] B. Meng and W. H. Weinberg. Dynamical monte carlo studies of molecular beam epitaxial growth models: interfacial scaling and morphology. *Surface Science*, 364(2):151–163, 1996.

[59] Vasily V. Bulatov and Wei Cai. *Computer Simulations of Dislocations (Oxford Series on Materials Modelling*. Oxford Univ Press, 2006.

[60] Stephan A. Baeurle, Takao Usami, and Andrei A. Gusev. A new multiscale modeling approach for the prediction of mechanical properties of polymer-based nanomaterials. *Polymer*, 47(26):8604–8617, 12 2006.

[61] K. Binder and M.H. Kalos. In K. Binder, editor, *Monte Carlo Methods in Statistical Physics*, volume 7 of *Springer Topics in Current Physics*, page 225. Springer-Verlag Berlin Heidelberg, 1979.

Bibliography VIII

[62] James L. Blue, Isabel Beichl, and Francis Sullivan. Faster monte carlo simulations. *Physical Review E*, 51(2):R867–R868, 02 1995.

[63] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, 03 2000.

[64] T. P. Schulze. Kinetic monte carlo simulations with minimal searching. *Physical Review E*, 65(3):036704–, 02 2002.

[65] James M. McCollum, Gregory D. Peterson, Chris D. Cox, Michael L. Simpson, and Nagiza F. Samatova. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Computational Biology and Chemistry*, 30(1):39–49, 2 2006.

[66] Angela Violi, Adel F. Sarofim, and Gregory A. Voth. Kinetic monte carlo–molecular dynamics approach to model soot inception. *Combustion Science and Technology*, 176(5-6):991–1005, 05 2004.

[67] M.A. Katsoulakis 3 (2005) A. Chatterjee, D.G. Vlachos. *International Journal for Multiscale Computational Engineering*, 3(135), 2005.

[68] A. Chatterjee and D.G. Vlachos. An overview of spatial microscopic and accelerated kinetic monte carlo methods. *Journal of Computer-Aided Materials Design*, 14:253, 2007.

Bibliography IX

[69] Aleksandar Donev, Vasily V. Bulatov, Tomas Oppelstrup, George H. Gilmer, Babak Sadigh, and Malvin H. Kalos. A first-passage kinetic monte carlo algorithm for complex diffusion–reaction systems. *Journal of Computational Physics*, 229(9):3214–3236, 2010.

[70] Kristen A. Fichthorn and W. H. Weinberg. Theoretical foundations of dynamical monte carlo simulations. *The Journal of Chemical Physics*, 95(2):1090–1096, 2017/01/03 1991.

[71] Santiago A. Serebrinsky. Physical time scale in kinetic monte carlo simulations of continuous-time markov chains. *Physical Review E*, 83(3):037701–, 03 2011.

[72] P A Maksym. Fast monte carlo simulation of mbe growth. *Semiconductor Science and Technology*, 3(6):594, 1988.

[73] Boris D Lubachevsky. Efficient parallel simulations of dynamic ising spin systems. *Journal of Computational Physics*, 75(1):103–122, 1988.

[74] K. M. Chandy and J. Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452, 1979.

[75] J. Misra. Distributed discrete-event simulation. *ACM Comput. Surv.*, 18:39, (986.

Bibliography X

[76] Yunsic Shim and Jacques G. Amar. Semirigorous synchronous sublattice algorithm for parallel kinetic monte carlo simulations of thin film growth. *Physical Review B*, 71(12):125432–, 03 2005.

[77] Giorgos Arampatzis, Markos A. Katsoulakis, Petr Plecháč, Michela Taufer, and Lifan Xu. Hierarchical fractional-step approximations and parallel kinetic monte carlo algorithms. *Journal of Computational Physics*, 231(23):7795–7814, 2012.

[78] Ignacio Martin-Bragado, J. Abujas, P. L. Galindo, and J. Pizarro. Synchronous parallel kinetic monte carlo: Implementation and results for object and lattice approaches. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 352:27–30, 2015.

[79] G. Korniss, Z. Toroczkai, M. A. Novotny, and P. A. Rikvold. From massively parallel algorithms and fluctuating time horizons to nonequilibrium surface growth. *Physical Review Letters*, 84(6):1351–1354, 02 2000.

[80] G. Korniss, M. A. Novotny, H. Guclu, Z. Toroczkai, and P. A. Rikvold. Suppressing roughness of virtual times in parallel discrete-event simulations. *Science*, 299(5607):677, 01 2003.

[81] Mehran Kardar, Giorgio Parisi, and Yi-Cheng Zhang. Dynamic scaling of growing interfaces. *Physical Review Letters*, 56(9):889–892, 03 1986.

Bibliography XI

[82] L. Xu, M. Taufer, S. Collins, and D. G. Vlachos. Parallelization of tau-leap coarse-grained monte carlo simulations on gpus. In *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, pages 1–9, 2010.

[83] Jens Nielsen, Mayeul d'Avezac, James Hetherington, and Michail Stamatakis. Parallel kinetic monte carlo simulation framework incorporating accurate models of adsorbate lateral interactions. *The Journal of Chemical Physics*, 139(22):224706, 2018/08/20 2013.

[84] Ivan Komarov and Roshan M. D'Souza. Accelerating the gillespie exact stochastic simulation algorithm using hybrid parallel execution on graphics processing units. *PLOS ONE*, 7(11):e46693–, 11 2012.

[85] G Korniss, M. A Novotny, and P. A Rikvold. Parallelization of a dynamic monte carlo algorithm: A partially rejection-free conservative approach. *Journal of Computational Physics*, 153(2):488–508, 1999.

[86] E. Martínez, P. R. Monasterio, and J. Marian. Billion-atom synchronous parallel kinetic monte carlo simulations of critical 3d ising systems. *Journal of Computational Physics*, 230(4):1359–1369, 2011.

[87] Weiliang Chen and Erik De Schutter. Parallel steps: Large scale stochastic spatial reaction-diffusion simulation with high performance computers. *Frontiers in Neuroinformatics*, 11:13, 2017.

Bibliography XII

[88] F. Jiménez and C. J. Ortiz. A gpu-based parallel object kinetic monte carlo algorithm for the evolution of defects in irradiated materials. *Computational Materials Science*, 113:178–186, 2016.

[89] Guido Klingbeil, Radek Erban, Mike Giles, and Philip K. Maini. Stochsimgpu: parallel stochastic simulation for the systems biology toolbox 2 for matlab. *Bioinformatics*, 27(8):1170–1171, 04 2011.

[90] Daniele D. Agostino, Giulia Pasquale, Andrea Clematis, Carlo Maj, Ettore Mosca, Luciano Milanesi, and Ivan Merelli. Parallel solutions for voxel-based simulations of reaction-diffusion systems. *BioMed Research International*, 2014:10, 2014.

[91] Lorenzo Dematté and Tommaso Mazza. On parallel stochastic simulation of diffusive systems. In Monika Heiner and Adelinde M. Uhrmacher, editors, *Computational Methods in Systems Biology*, pages 191–210, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[92] Michael Creutz. Microcanonical monte carlo simulation. *Physical Review Letters*, 50(19):1411–1414, 05 1983.

[93] Fang Chen, László Lovász, and Igor Pak. Lifting markov chains to speed up mixing, 1999.

Bibliography XIII

[94] Persi Diaconis, Susan Holmes, and Radford M. Neal. Analysis of a nonreversible markov chain sampler. pages 726–752, 2000.

[95] Thomas P. Hayes and Alistair Sinclair. Liftings of tree-structured markov chains. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 6302 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, 2010.

[96] Konstantin S. Turitsyn, Michael Chertkov, and Marija Vucelja. Irreversible monte carlo algorithms for efficient sampling. *Physica D: Nonlinear Phenomena*, 240(4–5):410–414, 2 2011.

[97] Marija Vucelja. Lifting—a nonreversible markov chain monte carlo algorithm. *American Journal of Physics*, 84(12):958–968, 2017/01/06 2016.

[98] Simon Duane, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987.

[99] Alexandre Bouchard-Côté, Sebastian J. Vollmer, and Arnaud Doucet. The bouncy particle sampler: A nonreversible rejection-free markov chain monte carlo method. *Journal of the American Statistical Association*, 113(522):855–867, 04 2018.

[100] Manon Michel and Stéphane Sénécal. *Forward Event-Chain Monte Carlo: a general rejection-free and irreversible Markov chain simulation method*. 02 2017.

Bibliography XIV

[101] Joris Bierkens, Paul Fearnhead, and Gareth Roberts. *The Zig-Zag Process and Super-Efficient Sampling for Bayesian Analysis of Big Data*. 07 2016.

[102] Joris Bierkens and Gareth Roberts. A piecewise deterministic scaling limit of lifted metropolis-hastings in the curie-weiss model. *Ann. Appl. Probab.*, 27(2):846–882, 2017.

[103] E. A. J. F. Peters and G. de With. Rejection-free monte carlo sampling for general potentials. *Physical Review E*, 85(2):026703–, 02 2012.

[104] Etienne P. Bernard, Werner Krauth, and David B. Wilson. Event-chain monte carlo algorithms for hard-sphere systems. *Physical Review E*, 80(5):056704–, 11 2009.

[105] Etienne P. Bernard and Werner Krauth. Two-step melting in two dimensions: First-order liquid-hexatic transition. *Physical Review Letters*, 107(15):155704–, 10 2011.

[106] Sebastian C Kapfer and Werner Krauth. Sampling from a polytope and hard-disk monte carlo. *Journal of Physics: Conference Series*, 454(1):012031, 2013.

[107] Sebastian C. Kapfer and Werner Krauth. Two-dimensional melting: From liquid-hexatic coexistence to continuous transitions. *Physical Review Letters*, 114(3):035702–, 01 2015.

Bibliography XV

[108] Alejandro Mendoza-Coto, Rogelio Díaz-Méndez, and Guido Pupillo. Event-driven monte carlo: Exact dynamics at all time scales for discrete-variable models. *EPL (Europhysics Letters)*, 114(5):50003, 2016.

[109] Manon Michel, Johannes Mayer, and Werner Krauth. Event-chain monte carlo for classical continuous spin models. *EPL (Europhysics Letters)*, 112(2):20003, 2015.

[110] Yoshihiko Nishikawa, Manon Michel, Werner Krauth, and Koji Hukushima. Event-chain algorithm for the heisenberg model: Evidence for \$z\ensuremath{\simeq}1\$ dynamic scaling. *Physical Review E*, 92(6):063306–, 12 2015.

[111] Manon Michel, Sebastian C. Kapfer, and Werner Krauth. Generalized event-chain monte carlo: Constructing rejection-free global-balance algorithms from infinitesimal steps. *The Journal of Chemical Physics*, 140(5):054116, 2018/08/13 2014.

[112] Tobias A. Kampmann, Horst-Holger Boltz, and Jan Kierfeld. Parallelized event chain algorithm for dense hard sphere and polymer systems. *J. Comput. Phys. 0021-9991*, 281(C):864–875, 2015.

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

Bibliography XVI

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

[115] 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).

[116] 817 G. E. Forsythe: Math. Comput. **26**. 1972. 1972.

Index

Index I

Index II

Index III

Index IV

Index V

Index VI