Foundations of Chemical Kinetics Lecture 15: The kinetic Monte Carlo method

The single-molecule simulation method

Marc R. Roussel

November 16, 2021

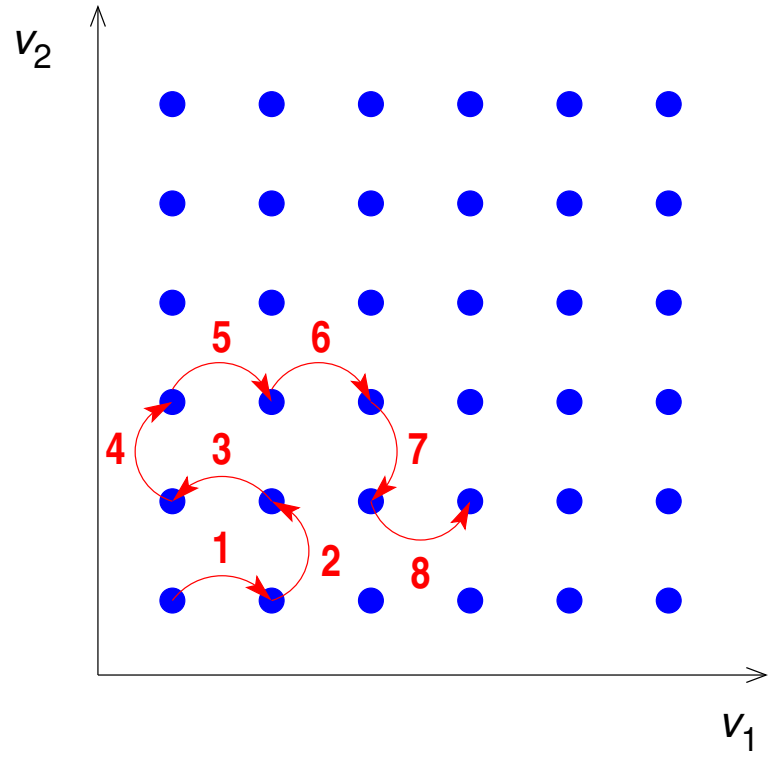
Markov chains

Moving up and down a set of vibrational energy levels or reallocating energy between vibrational modes in IVR are examples of Markov chains.

- These are random processes, which may be driven by collisions or simply by the quantum mechanics of energy transfer between vibrational modes.
- They are memoryless, i.e. what is likely to happen next only depends on the current state of the system, and not on how this state was reached.
- They are Poisson processes, i.e. the waiting time until the next event is independent of how long we have been waiting so far.
- The state space is discrete, consisting of the countable (if possibly infinite) number of quantum states of a molecule.

Simulating Markov chains: a picture

In the Landau-Teller approximation



Both the sequence of jumps and the times at which the jumps occur are random variables.

Simulating Markov chains: statistical properties

Suppose that a molecule is in state s at time t. Then

- The next jump is chosen randomly, with probabilities directly proportional to the transition rates w_{sj}.
  - For a Landau-Teller process, only jumps to states corresponding to a change of one unit in one of the vibrational quantum numbers need to be considered.
  - For IVR, only jumps to states with the same total energy need to be considered.
- The time between jumps is exponentially distributed, with a characteristic decay time $\tau = \left(\sum_j w_{sj}\right)^{-1}$, where the sum is over states that can be reached in one jump from s.

The Kinetic Monte Carlo (KMC) Algorithm

Simulating single molecules

At each step:

1. Calculate $w_{\text{tot}} = \sum_j w_{sj}$ for the current state s and for all j reachable in one step from s.
2. Generate two random numbers $r_{1,2} \in (0, 1)$.
3. Use r_1 to generate the time to next jump:

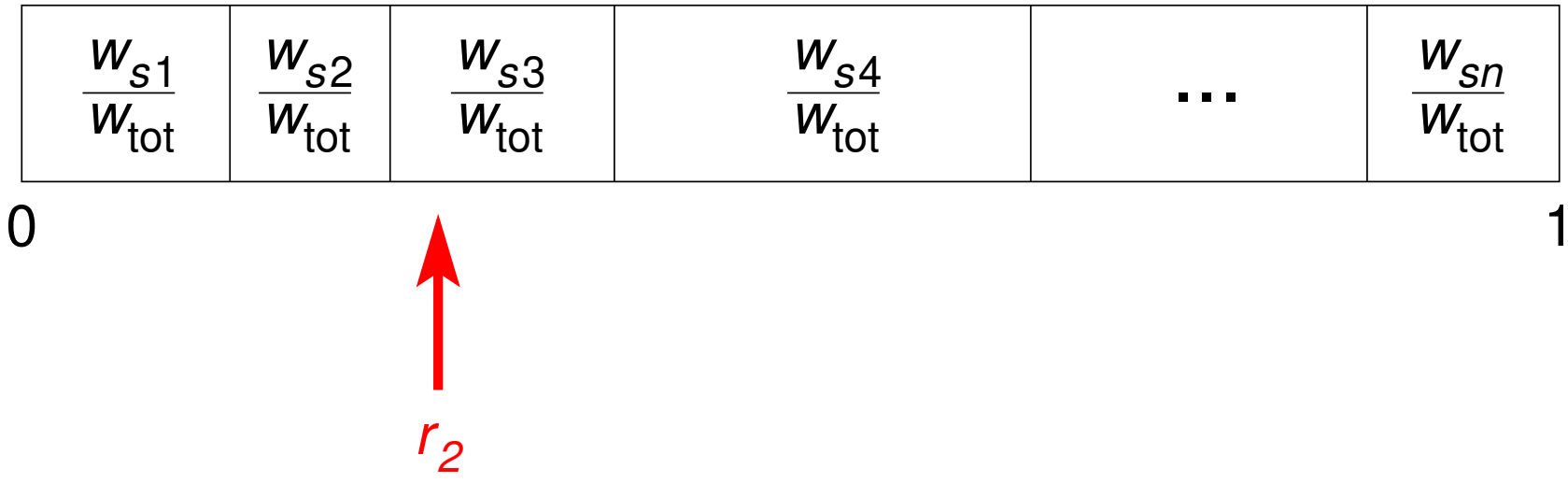
$$\Delta t = -\ln r_1 / w_{\text{tot}}$$

4. Use r_2 to pick the destination of the jump as follows: Find the smallest k such that

$$\sum_{j=1}^{k} w_{sj} > r_2 w_{\text{tot}}$$

5. Add Δt to the simulation time.
6. Change the state of the molecule according to the jump chosen.

The logic of the jump selection method



Practical issues

- A complete program includes some setup (storing the w_{ij} values, setting $t = 0$, choosing an initial state for the molecule, etc.).
- Typically, we would either store the result of every jump and the corresponding time, or we would store the state of the molecule at fixed time intervals.
- We can only get so much information from a single simulation of one molecule, so we would usually repeat the simulation many times and compute some averages from the pool of simulation results.

A (relatively) simple example of IVR with reaction

- Suppose that a molecule has 2 vibrational modes with $\omega_0^{(1)} = 2\omega_0^{(2)}$.
- Suppose that the reaction requires at least $v_2 = 10$.
  In other words, if we reach $v_2 = 10$, the molecule is assumed to react and the simulation stops.
- We want to calculate k_{2K} from the RRK mechanism. This is

$$k_{2K} = 1/\bar{t}_{\text{react}}$$

- RRK theory uses a microcanonical treatment for k_{2K}, i.e. it calculates k_{2K} as a function of the energy of A^*, the energized molecule.

A (relatively) simple example of IVR with reaction
(continued)

- Define $n_2 = 2v_1 + v_2$, the amount of energy stored in the two vibrational modes expressed in units of $\hbar\omega_0^{(2)}$.
- All of the states with equal energy are equal in probability, so we can skip the jump selection part of the KMC algorithm, and just generate a random next state.
  - Generate a new random v_1 between 0 and $\lfloor n_2/2 \rfloor$. ($\lfloor \cdot \rfloor$ is the 'floor' operation, i.e. rounding down.)
  - Calculate $v_2 = n_2 - 2v_1$

A (relatively) simple example of IVR with reaction
(continued)

- Because all of the jumps are to states of equal energy, $w_{ij} = w_{ji}$.
- For simplicity, assume that all of the w_{ij} are equal to w.
- The total number of possible states is $\lfloor n_2/2 \rfloor + 1$ $(0, 1, 2, \ldots \lfloor n_2/2 \rfloor)$.
- From any given v_1, there are therefore $\lfloor n_2/2 \rfloor$ possible jump destinations.
- Therefore, $w_{\text{tot}} = w \lfloor n_2/2 \rfloor$.

Our simulation code to-do list

- For several values of n_2:
  1. Obtain many realizations of single-molecule IVR:
     1. Simulate jumps until we have $v_2 \geq 10$.
     2. Keep track of the simulated time it takes for this to happen
  2. Average the IVR time and calculate $k_{2K}(n_2)$.

A side note about IVR

- The master equation treatment of IVR is a classical cartoon.
- A better picture is that, at any given time, the molecule exists in a superposition of states of approximately equal energy (subject to the time-energy uncertainty principle).
- The superposition evolves with time towards a uniform distribution over all states within the energy set.
- The probability of reaction per unit time is proportional to the weight of states that carry sufficient energy in the reaction coordinate among all the states in the superposition.

Building a computer program

- The smart thing is to build the program in steps, starting from the inside and working outwards.
- Put lots of comments in your code (lines starting with %) to remind you how it works.
  It won't be obvious tomorrow!
- In this case:
  1. Write code for the Boltzmann sampling.
  2. Write a single simulation step.
  3. Wrap the step in a loop to repeat until reaction occurs.
  4. Wrap the initial Boltzmann sampling and 'stepper' in a loop to collect statistics over many simulations.

Matlab commands and functions we will need

Variables:	Names of variables always start with a letter and can include underscores and digits. Names are case-sensitive.
Matrices	can be stored in a variable `M(i,j)` addresses the (i, j) element of the matrix M. An $n \times 1$ or $1 \times n$ matrix is a vector. Elements of a vector can be addressed with a single argument, e.g. `v(i)`.
rand(m,n)	returns a matrix of random numbers in the interval $(0, 1)$ of dimensions $m \times n$
randi([a,b])	returns a random integer in the interval $[a, b]$
ceil(x), floor(x)	return the ceiling and floor of x, respectively
log(x)	returns the natural logarithm of x
mean(v)	returns the average of the values in the vector v

Matlab commands and functions we will need
(continued)

a:b is a range, specifically the list of numbers $a, a+1, \ldots, b$.

while loops repeat until the condition following the keyword `while` is met.
Loops are terminated by the keyword `end`.

for loops repeat over a range of values, e.g. `for i=1:20` would start with $i = 1$, iterate through all of the values of i including 20, then stop.

Simulation parameters

- $\hbar\omega_0^{(2)}/k_B T = 0.1$
- $w_{ij} = 10^{15}\,\mathrm{s}^{-1}$ for all i, j