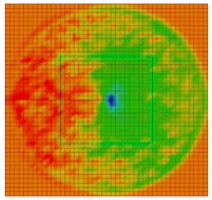
SAND2013-10043C

October 2013-January 2014

ESP900: Atomistic/Molecular Simulation:

Lecture 11: kinetic Monte Carlo methods

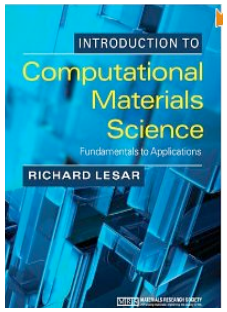


Instructor: Reese Jones

rjones@sandia.gov

(925) 294-4744

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



Topics

- Introduction
- Spatial KMC and the N-fold way
- KMC for chemical reactions
- Implementation issues
- The SPPARKS code
- Application examples
- On-the-fly KMC

Review of Metropolis Monte Carlo

- Last lecture, we looked at the Metropolis method.
- For each step:
  - Pick a move at random: the **trial move**
  - Calculate the change in energy for the trial move
  - Compute the probability of accepting the trial move, typically:

$$P_{accept} = \begin{cases} 1, & \text{if } \Delta E \le 0 \\ \exp\left(-\Delta E / k_B T\right), & \text{if } \Delta E > 0 \end{cases}$$

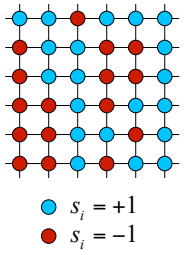
  - Generate a random number r, and accept the move if $r < P_{accept}$
- Eventually, people started using this method to look at *dynamics* of systems, not just equilibrium states
  - E.g. Stoll et al., *Phys. Rev. B* **8**:3266 (1973)
- These could probably be considered to be Kinetic Monte Carlo simulations… but usually aren't
- KMC in this context usually refers to a change in the algorithm itself, not just the introduction of dynamics

The N-Fold Way

- In the Metropolis method, for systems close to equilibrium the acceptance probability for most events is often very low
  - Especially true near equilibrium, when $\Delta E >> 0$ for many events
  - Leads to slow evolution and convergence
- The **n-fold** method was introduced to overcome this problem
  - Bortz, Kalos and Lebowitz, *J. Comp. Phys* **17**:10 (1975)
    - "**BKL**" for short
- The new method is *equivalent* to the Metropolis method, but changes the order of operations
  - Instead of choosing a trial move at random and testing for acceptance, compute the probability of *all* possible events
    - Obviously, requires that the number of events be finite, which limits the types of problems this can be used on
  - At each step, ask the question: Which event happens *next*?

The Ising Model

- BKL worked with the **Ising model**, a very important model in materials science
  - Lattice of sites, where each site has one of 2 values ("spin"), e.g. ±1
  - Model used for phase changes, magnetic spin systems, critical phenomena, etc…

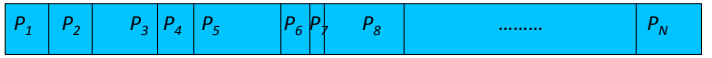


$$E(\mathbf{s}) = -J\sum_{i,j} s_i s_j - H\sum_i s_i$$

- First sum is over nearest neighbor pairs
  - If J>0, favors like neighbors
- Second term biases system toward +1 or -1 (depending on sign of H)
- Metropolis Monte Carlo trial move: pick a site at random and flip its spin to the opposite value

The N-Fold Method

- In the Ising model, each site has just one possible move (or "event"): flip spin from one value to the other
- Goal: choose an event at random to execute next
- Intuitively, this random choice for each event *i* should be weighted by its acceptance probability P_i
  - Imagine arranging all of the events in a line, weighted by their probabilities:



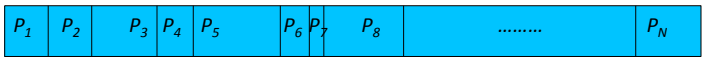
- Choose a point along this line at random, and execute that event

Define: $Q_i \equiv \sum_{j=1}^{i} P_j$

Choose random number $r \in [0,1]$

Find and execute event with: $\frac{Q_{i-1}}{Q_N} < r < \frac{Q_i}{Q_N}$

The N-Fold Method



$$Q_i \equiv \sum_{j=1}^{i} P_j$$

- After the chosen event gets executed, we have to recompute this list of events and probabilities
- For many systems, most of the possible events and their probabilities remain unchanged
  - E.g. for the Ising model, only the flipped site and its nearest neighbors are affected
- However, we still have a couple of inefficiencies here…
  - We need to compute the cumulative sum of all of the events, Q_i, any time any one of them changes
  - To choose an event, it appears we have to loop over *all* events to find the one where:

$$\frac{Q_{i-1}}{Q_N} < r \le \frac{Q_i}{Q_N}$$

  - These are both order N operations (not good)

The 10-Fold Way

- The BKL paper authors found a nice solution for these inefficiencies
- Note that there are really only two "events" for the 2D Ising model:

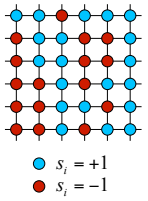


TABLE I

Classifications of Spins in the Ten-Fold Way[a]

Class	Spin	Number of spin up nearest neighbors
1	Up	4
2	Up	3
3	Up	2
4	Up	1
5	Up	0
6	Down	4
7	Down	3
8	Down	2
9	Down	1
10	Down	0

From BKL, *J Comp Phys* (1975)

- All of the events within the same class have *equal* probability
- So we can first select a *class* at random with weighted probability, then select an event from that class with uniform probability
  - Details on next slide…

The 10-Fold Way

- Define the probability P^{class} and cumulative probability Q^{class} for each of the 10 classes:

$$Q_i^{class} = \sum_{i=1}^{10} n_i P_i^{class}$$

$$n_i = \#\text{ of events in class } i \ (1 < i < 10)$$

$$P_i^{class} = \text{constant probability for each event in class } i$$

- Generate random number r between 0 and 1, and find class such that:

$$\frac{Q_{i-1}^{class}}{Q_{10}^{class}} < r \le \frac{Q_i^{class}}{Q_{10}^{class}}$$

  - This operation has cost of order 10, no matter how large the system is
- Then choose a random event from this class, with uniform probability
  - This operation has cost of order 1
- BKL called this the "10-fold way", and the general method the "n-fold way"

Gillespie's Stochastic Simulation Algorithm (SSA)

- A similar but independent method was developed at around the same time as the BKL paper
- D.T. Gillespie, "Exact stochastic simulations of coupled chemical reactions." *J. Phys. Chem.* **81**(25):2340
- Developed for systems of chemical reactions
- Gillespie's work is more obviously related to dynamics (and kinetics)
- Even though the methods are similar in the basic idea, the two communities don't seem to have much overlap
  - Physicists and mathematicians reference BKL
  - Chemists reference Gillespie
- It's worth understanding the Stochastic Simulation Algorithm (SSA) in detail (even for non-chemists!)

Stochastic Simulation Algorithm

- Suppose we have a system of chemical reactions taking place in a fixed volume:

$$X + Y \xrightarrow{c_1} 2Y$$

$$2Y \xrightarrow{c_2} Z$$

$$etc...$$

  - Rate c_i for each reaction will be defined more formally later…
- One way to follow changes in the amount of each chemical species is to write coupled, deterministic ODE's for them:

$$\frac{dN_X}{dt} = f_1(X,Y,Z)$$

$$\frac{dN_Y}{dt} = f_2(X,Y,Z)$$

$$\frac{dN_Z}{dt} = f_3(X,Y,Z)$$

- But in small systems, with small numbers of molecules, this approach has some flaws:
  - Real system is discrete, not continuous (numbers of molecules are integers)
  - Real system is stochastic, not deterministic

Stochastic Simulation Algorithm

- Gillespie's SSA proceeds stochastically, one reaction at a time
- Have to answer two questions:
  - **When** does the next reaction occur?
  - **What** is the next reaction that occurs?
- Obviously, these questions can only be answered in some "probabilistic" way
- First, note that reaction rates (the ci constants) can be thought of as probabilities
  - E.g. for the reaction:

$$X + Y \xrightarrow{c_1} 2Y$$

- The rate is defined such that:

$c_1 dt$ = average probability that a particular $X + Y$ pair in the system will undergo reaction 1 in the next infinitesimal time interval dt

- The total number of X+Y pairs is $N_X N_Y$, so:

$N_X N_Y c_1 dt$ = average probability that *some* $X + Y$ pair in the system will undergo reaction 1 in the next infinitesimal time interval dt

Stochastic Simulation Algorithm

- More generally:

  $c_i dt$ = average probability that a particular set of reactants for reaction i will undergo reaction i in the next infinitesimal time interval dt

  $a_i dt$ = average probability that *some* set of reactants for reaction i will undergo reaction i in the next infinitesimal time interval dt

- E.g. for our original set of 2 reactions:

$$X + Y \xrightarrow{c_1} 2Y$$

$$2Y \xrightarrow{c_2} Z$$

$$a_1 = N_X N_Y c_1$$

$$a_2 = \frac{1}{2} N_Y (N_Y - 1)$$

  - Note that a_2 has a different form since it involves two atoms of the same species

Stochastic Simulation Algorithm

- So we can compute the set of probabilities a_i for each reaction i
- Note that a_i has units of 1/time, and can be thought of as the expected rate for each reaction
- Now define the **reaction probability density function** $P(\tau,i)$:

  $P(\tau,i)d\tau$ ≡ probability that, starting from time t, the next reaction will occur in the infinitesimal time interval $(t+\tau, t+\tau+d\tau)$ *and* will be a reaction of type i

- We can break this down into the product of two separate probabilities:

$$P(\tau,i)d\tau = P_1(\tau)P_2(i|\tau)d\tau$$

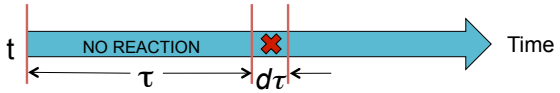
  where

  $P_1(\tau)d\tau$ ≡ probability that the next reaction occurs in the interval $(t+\tau, t+\tau+d\tau)$

  $P_2(i|\tau)$ ≡ probability that the next reaction will be of type i, given that the next reaction occurs at time $t+\tau$

Stochastic Simulation Algorithm

- First, let's compute $P_1(\tau)d\tau$, the probability that the next reaction of *any* type occurs in the interval $(t+\tau, t+\tau+d\tau)$



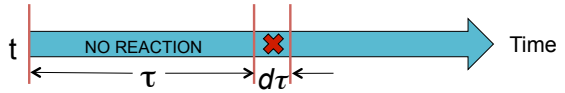
- To compute this, note that the individual reaction probabilities can be summed over all reactions (assume there are M of them), so that:

  $a_0 d\tau$ = average probability that any reaction will take place in the next infinitesimal time interval $d\tau$

  where

$$a_0 = \sum_{i=1}^{M} a_i$$

Stochastic Simulation Algorithm



- Now the total probability P_1 can be broken down further:

$$P_1(\tau)d\tau = P_0(\tau)a_0 d\tau$$

  where

  $P_0(\tau)$ = probability that NO reaction will occur within time τ

  $a_0 d\tau$ = probability that ANY reaction will occur in the subsequent interval $d\tau$

- Almost there…

  $(1 - a_0 d\tau)$ = probability that NO reaction will occur within an interval $d\tau$

$$P_0(\tau + d\tau) = (1 - a_0 d\tau) P_0(\tau)$$

$$\frac{P_0(\tau + d\tau) - P_0(\tau)}{d\tau} = -a_0 P_0(\tau)$$

$$P_0(\tau) = \exp(-a_0 \tau)$$

Stochastic Simulation Algorithm

- So:

$$P_1(\tau)d\tau = P_0(\tau)a_0 d\tau = a_0 \exp(-a_0\tau)d\tau$$

- $P_2(i|\tau)$ is easier; the probability that the next reaction is of type i is simply proportional to a_i, normalized by the sum over all types:

$$P_2(i|\tau) = \frac{a_i}{a_0}$$

- So the full reaction probability density function is:

$$P(\tau,i)d\tau = \left(a_0 \exp(-a_0\tau)d\tau\right)\left(\frac{a_i}{a_0}\right)$$

$$= a_i \exp(-a_0\tau)d\tau$$

- In practice, this gets simulated in two steps:
  - Use $P_1(\tau)$ to generate the time τ to the next reaction
  - Use $P_2(i|\tau)$ to select the next reaction i

Computing the Time to Next Reaction

- We need to select a time τ from the probability distribution $P_1(\tau)$
  - But in general, random number generators select from a uniform distribution, usually on the interval (0,1)
  - How do we map from one distribution to another?
- Let $x(\tau)$ be a random number with uniform distribution on the interval (0,1); we can compute the mapping by integrating:

$$\int_0^\tau P_1(\tau')d\tau' = \int_0^x P(x(\tau'))dx'$$

$$\int_0^\tau a_0 \exp(-a_0\tau')d\tau' = \int_0^x dx'$$

$$1 - \exp(-a_0\tau) = x$$

$$\tau = \frac{1}{a_0}\ln\left(\frac{1}{1-x}\right)$$

- Note that if x is uniformly distributed on (0,1), then so is r_1=1-x:

$$\tau(r_1) = \frac{1}{a_0}\ln\left(\frac{1}{r_1}\right)$$

  - Note that τ will be on the interval (0,∞), as expected

Computing the Type of the Next Reaction

- Now select the next reaction, exactly as in the n-fold method
  - Generate a random number r_2 on the interval (0,1), and find the type i such that:

$$\frac{\sum_{j=1}^{i-1} a_j}{a_0} < r_2 \le \frac{\sum_{j=1}^{i} a_j}{a_0}$$

- After the reaction is selected, the numbers of each species involved in the reaction get updated, e.g.:

$$X + Y \xrightarrow{c_1} 2Y$$

$$N_X \rightarrow N_X - 1$$

$$N_Y \rightarrow N_Y + 1$$

- Since the reaction probabilities a_i depend on the numbers of species (e.g. $a_1=N_X N_Y c_1$), those must be updated for all reactions involving the modified species (not just the latest reaction)

SSA: Summary of Algorithm

- Given: initial numbers of all species, list of M reactions with rates c_i
- Set system clock to zero: $T = 0$

1. Compute all reaction probabilities a_i
2. Compute total reaction rate a_0:

$$a_0 = \sum_{i=1}^{M} a_i$$

3. Generate random number r_1 on (0,1) and compute time to next reaction:

$$\tau = \frac{1}{a_0}\ln\left(\frac{1}{r_1}\right)$$

4. Generate random number r_2 on (0,1) and select next reaction i such that:

$$\frac{\sum_{j=1}^{i-1} a_j}{a_0} < r_2 \le \frac{\sum_{j=1}^{i} a_j}{a_0}$$

5. "Execute" reaction by updating numbers for all products and reactions of reaction i
6. Update system clock: $T \rightarrow T + \tau$
7. End if total simulation time has been reached
8. Go to step 1

SSA: Example

- Look at an example from Gillespie's original paper:
  - Solve the 2-reaction example case:

$$X + Y \xrightarrow{c_1} 2Y$$

$$2Y \xrightarrow{c_2} Z$$

where:

$$N_X \text{ is assumed constant}$$

$$c_1 N_X = 5$$

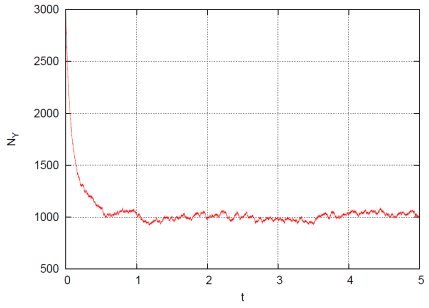
$$c_2 = 0.005$$

$$N_Y(t=0) = 3000$$

  - Equivalent deterministic system has a steady state of:

$$N_Y = \frac{c_1 N_X}{c_2} = 1000$$

  - SSA solution fluctuates around this value



Kinetic Monte Carlo

- The SSA and BKL methods are essentially equivalent
  - The original BKL paper even includes time dependence and the selection of a timestep, equal to that in the SSA
- This basic algorithm is the method that is usually called **Kinetic Monte Carlo**
- The algorithm can be used for any system that can be described by a set of discrete events with known rates/ probabilities
- Especially useful for atomic systems where:
  - All atoms remain on a set of pre-defined lattice points, and
  - Dynamics is driven by **infrequent events** (relative to atomic vibrational timescale)
- E.g. crystal growth, surface diffusion

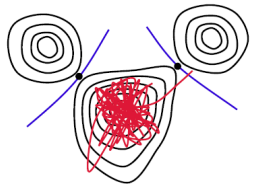
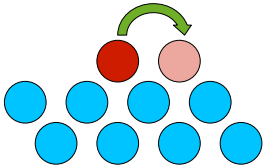
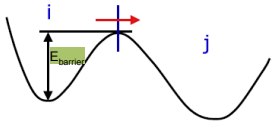


Illustration of potential energy surface in an infrequent event system (from Voter, 2005)

KMC Events



A surface diffusion atom hop event



- For many atomistic systems, we can use **transition state theory** to estimate the rate of a given event
- We compute the barrier energy $E_{barrier}$, the amount of energy needed to reach the saddle point separating two local energy minima
- The rate is computed from an Arrhenius equation:

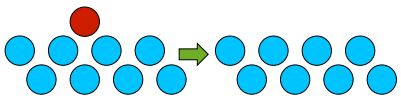
$$k = k_0 \exp\left(-\frac{E_{barrier}}{k_B T}\right)$$

where k_0 is the "attempt frequency" (related to the vibrational frequency of the system)

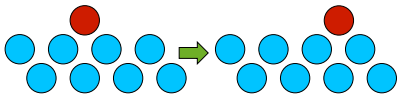
KMC Events

Events in atomistic systems can be broadly classified as **Glauber** or **Kawasaki dynamics**

- Glauber dynamics
  - Atoms/particles are inserted or removed from the lattice
  - Number of particles is not conserved
  - Can be thought of as exchanging particles with an external reservoir (e.g. surrounding gas)
  - Used for simulations of evaporation/ condensation, deposition
- Kawasaki dynamics
  - Atoms/particles moved from one site to another on the lattice
  - Number of particles is conserved
  - Used for simulations of diffusion
- Both types can be combined in the same simulation



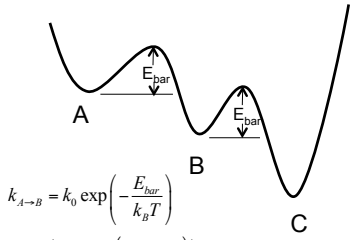
Glauber Dynamics



Kawasaki Dynamics

Connection Between Transition State Theory and the Metropolis Monte Carlo Method

- Sometimes the Metropolis MC algorithm (or at least, the MMC acceptance criterion) is used to simulate dynamics
- How is this related to "real" dynamics, e.g. transition state theory?
- Make two assumptions about our set of events:
  1. All events have the same attempt frequency k_0
  2. All energetically *downhill* events have the same barrier energy E_{bar}
- Then for all downhill events (like A→B):



$$k_{A\to B} = k_0 \exp\left(-\frac{E_{bar}}{k_B T}\right)$$

- For uphill events (like B → A):

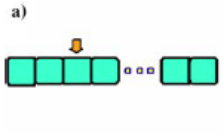
$$k_{B\to A} = k_0 \exp\left(-\frac{E_{bar} + (E_A - E_B)}{k_B T}\right)$$

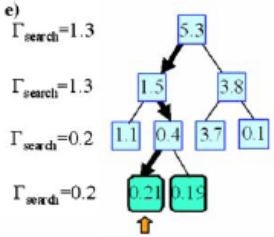
- The time variable can then be rescaled (note that the scale factor depends on temperature!) to give in general:

$$k_{A\to B} = \begin{cases} 1, & \text{if } E_B \le E_A \\ \exp\left(-\frac{E_B - E_A}{k_B T}\right), & \text{if } E_B > E_A \end{cases}$$

Implementation: Search & Update Algorithms

- For a general simulation with many events, the selection of the event can become expensive
  - This is the problem that BKL solved using their n-fold way, but this may not be generally applicable
- A simple linear search to find *i* such that $\frac{Q_{i-1}}{Q_N} < r \le \frac{Q_i}{Q_N}$ scales as *N*
- More efficient algorithms are possible, e.g. binary search
  - Tree structure allows faster search; scales as $\log_2 N$
- See Chatterjee & Vlachos (2007) reference for a nice review of various algorithms

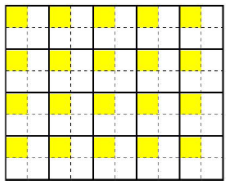




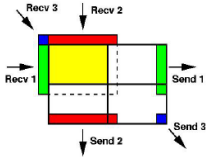
Linear vs. binary search algorithm (from Chatterjee and Vlachos, 2007)

KMC in Parallel: SPPARKS

- Sandia's SPPARKS code is a freely available, general, parallelized KMC code:
  - www.cs.sandia.gov/~sjplimp/spparks.html
- True KMC is an inherently serial algorithm
  - One event at a time
  - Each event affects possible events and rates around it, so not strictly legal to execute multiple events in parallel (e.g. on multiple processors)
- SPPARKS (and other parallelized KMC codes) make an approximation that allows independent events to be executed in parallel
  - Processor domains sub-divided into sectors to perform events that will not conflict with neighbors
  - Balance must be achieved between accuracy and efficiency



2D domain partitioned among 20 processors (4x5); each sub-domain further divided into 4 sectors



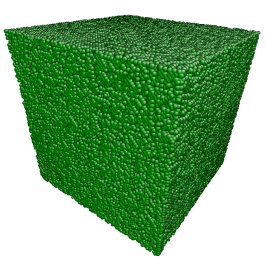
Each processor can perform events on a given sector while other sectors are "frozen"; inter-processor communication updates boundaries

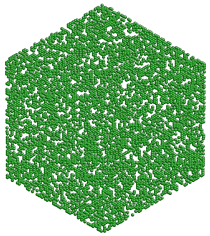
SPPARKS

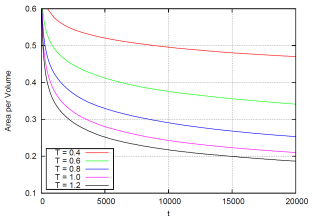
- SPPARKS provides the main computational "engine" to do event search and selection
- Handles lattice creation, parallelization of the domain, I/O, and other general functions
- Problem-specific information, like the definition of events and their probabilities, is specified in a user-supplied **application** class
- Application class is an inherited C++ class that must supply at least the following methods for a KMC simulation:
  - `site_energy`(): compute the energy associated with the given lattice site in the current state
  - `site_propensity`(): compute the total rate (or probability) at which events associate with the given site may take place
  - `site_event`(): perform an event at the given site and update the list of events and probabilities
- Some basic applications are included with the SPPARKS code, including the Ising model, Gillespie's SSA model, and some simple models for solid diffusion

KMC Example: Nanoporous Material Evolution

- The “diffusion” application of the SPPARKS code was used to compute the time evolution of a nanoporous material
  - Initially, atoms occupy random sites in a lattice with 50% volume fraction
  - Atom hop events on the surface model material diffusion
  - Surface area decreases as the pore structure coarsens over time

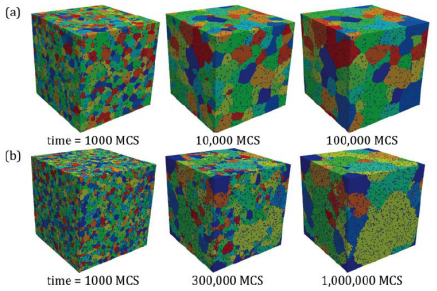






KMC Example: Grain Growth

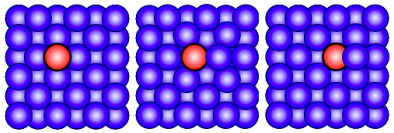
- A Potts model is often used to model growth of grains in a material
  - Similar to an Ising model, except instead of two possible values each site can have one of *N* integer values (so, Ising is a special case with *N*=2)
  - Energy function penalizes unlike neighbors, so large “grains” (volumes with the same site value everywhere) are energetically favorable
  - Each event flips the value of a site to that of one of its neighbors



[Plimpton et al., 2009]

“On-the-Fly” KMC

- The biggest source of inaccuracy in most KMC simulations (when compared with reality) is our lack of knowledge about the possible events that might take place
- Classic example: adatom diffusion on a (100) metal surface
  - It had been assumed that the relevant events in diffusion involve hops of an atom from one surface site to another
  - Peter Feibelman and coworkers at Sandia discovered using DFT calculations in 1990 that the lowest energy event is actually an exchange of atoms with the first layer
  - Other possible events might include multiple atom motion, or other non-intuitive processes

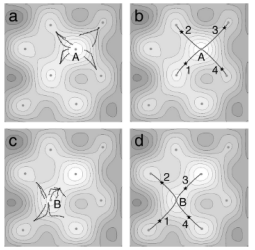


Adatom exchange process on a (100) surface (illustration from Voter, 2007)

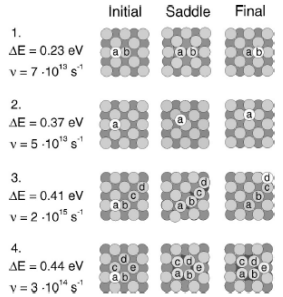
“On the Fly” KMC

Henkelman and Jonsson (J. Chem. Phys. **115**, 2001) introduced a KMC method that attempts to automatically find events and their rates by exploring the potential energy surface

- Called “On the Fly” KMC since it doesn’t require a pre-defined list of events



Exploration of potential energy surface to find nearby local energy minima (from Henkelman and Jonsson, 2001)



The four lowest energy events found for an adatom on a (100) surface, including the exchange event (from Henkelman and Jonsson, 2001)

Further Reading

- A. Chatterjee and D.G. Vlachos, “An overview of spatial microscopic and accelerated kinetic Monte Carlo methods,” *J. Computer-Aided Mater. Des.* **14**:253 (2007)
  - A nice tutorial and review article on KMC
- A.F. Voter, “Introduction to the Kinetic Monte Carlo Method,” in *Radiation Effects in Solids*, edited by K.E. Sickafus, E.A. Kotomin, and B.P. Uberuaga (Springer, Berlin, 2007), p. 1-24
  - Another good introduction to some of these topics
- S.J. Plimpton et al., “Crossing the Mesoscale No-Man’s Land via Parallel Kinetic Monte Carlo”, SAND Report #SAND2009-6226
  - SAND report summarizing the SPPARKS code, with several example applications
- A.B. Bortz, M.H. Kalos, and J.L. Lebowitz, “A New Algorithm for Monte Carlo Simulation of Ising Spin Systems”, *J. Comp. Phys.*, **17**:10 (1975)
  - The “BKL” paper on the n-fold way
- D.T. Gillespie, “Exact Stochastic Simulation of Coupled Chemical Reactions,” *J. Phys. Chem.* **81**(25):2340 (1977)
  - Development of the SSA

Homework

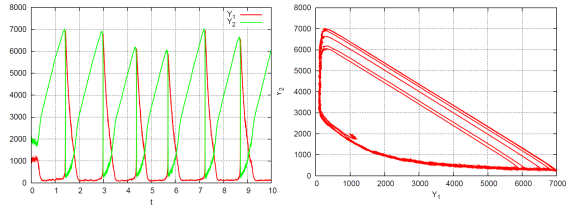
- Another example of an SSA problem from Gillespie’s *J Phys Chem* paper:
- The Brusselator:

$$\bar{X}_1 \xrightarrow{c_1} Y_1$$
$$\bar{X}_2 + Y_1 \xrightarrow{c_2} Y_2 + Z_1$$
$$2Y_1 + Y_2 \xrightarrow{c_3} 3Y_1$$
$$Y_1 \xrightarrow{c_4} Z_2$$

Your assignment: Download the brusselatorSSA.py script, fill in the missing parts of the code, and reproduce the plots of Y1 and Y2 vs. time from Gillespie’s paper (below);

with:

X_1 and X_2 constant
$c_1\bar{X}_1 = 5000$
$c_2\bar{X}_2 = 50$
$c_3 = 5\times10^{-5}$
$c_4 = 5$
$Y_1(0) = 1000$
$Y_2(0) = 2000$



INPUT

```
clock = 0
while clock < totalTime:
   a1 = NX * NY * c1
   a2 = 0.5 * NY * (NY - 1) * c2
   a0 = a1 + a2
   r1 = random.uniform(0,1)
   tau = 1.0/a0 * math.log(1.0/r1)
   r2 = random.uniform(0,1)
   iType = 1
   if (r2 > a1/a0):
      iType = 2
    if (iType == 1):
      NY = NY + 1
   else:
      NY = NY – 2
  clock = clock + tau
```

BONUS

- Change the neutral species in the fluid system for a water model like TIP4P and observe the differences in the solvation structure
- Make a fluid confined by two walls. Will the RDF change? Will it be uniform?

- Observe the differences in point defect structures obtained using a Stillinger-Weber potential vs Tersoff
- Reverse the loading of the nanobeam , will the response change?
- Add stress as an output does it tell you anything?

Lecture 12

Week 7: **Analyzing Inhomogeneous Systems**

- Identification and visualization of defects and structures
- Metrics, e.g. radial distribution function, common neighbor analysis, centrosymmetry
- Available tools
- Homework: Calculation of centrosymmetry and slip vector around a defect

Week 8 : **Molecular Dynamics**

- Newton's 2nd Law
- Time integration algorithms (Verlet, SHAKE, Gear)
- Conserved quantities
- Ensembles (NVE, NVT, NPT, NPH) & equations of motion
- Thermostats, e.g. Nose-Hoover
- Initial conditions and velocity distributions
- *Homework*: NVT average of pressure.

Reading Suggestions for Lec. 12

- Chapter 6 of LeSar
- Chapter 4 of Frenkel & Smit
- Chapter 3 & 6 of Evans & Morriss
- http://en.wikipedia.org/wiki/Molecular_dynamics
- http://lammps.sandia.gov/