**Monte Carlo Calculations**

**HW 9:** Wednesday, Nov. 7, 2018
**DUE:** Wednesday, Nov. 14, 2018
**READ:** Numerical Recipes in C++, Section 7.0-7.3 pages 340-378, random number generators
Section 7.7 pages 397-403 M.C. integration
Section 10.12 pages 549-551 Metropolis algorithm
OPTIONAL: Landau, Paez, and Bordeianu: chap. 5 Monte Carlo,
section 6.5-6.9 M.C. integration

1. **FINAL PROJECT OUTLINE DUE ON FRI. NOV. 16, 2018**: Please write a one page outline of your final project. This should include a title and a few paragraphs and equations outlining your project goals and how you expect to accomplish them. You should also include at least one reference. Your outline must be approved before turning in your final project report. You may start to work on the project before your outline is approved but you may need to change your project if the outline is not accepted. Also, if this is a joint project (with another course) you should state this in your outline and on your final report.

2. **FINAL PROJECT DUE ON WED. DEC 12, 2018** (during exams): Please read the previous handout. This will be graded out of 20 points total. Because of the tight deadlines for senior grades etc. there will be a 2 point/day penalty for late reports. If you have another exam on this day your project is due the next day you do not have an exam (please see me in advance). Please plan to work on your project well in advance. If you have registered S/U for this course, you do not have to turn in a final project or outline if you have turned in all homework and have an average of 8/10 or better.

3. For your information the homework mean and standard deviation are 9.1 and 0.64 for homework number 1 through 7 not including some late ones (which is very good).

**PROBLEM (10 points):**

# 1 Random Number Generator

Although some compilers come equipped with a random number generator, for this homework use the Ranq1.doub() function on page 351 of Numerical Recipes[7] (called the same way as Ran.doub() on page 342). Do not use the integer valued member functions to get small integers because the low order bits may be correlated. This is one case where it is probably best to use the code verbatim. You can include n3r.h or add the following to the top of the Num. Rec. RNG file.

```
//  define Num Rec. data types
typedef double Doub;
typedef int Int;
```

```
typedef unsigned int Uint;
typedef unsigned long long Ullong;
```

First test your random number generator by producing a histogram of the probability distribution of 100,000 random numbers (generate the histogram data in your C/C++ program with about 100 bins), and a plot of about 10,000 pairs of random numbers $(x_i, x_{i+1})$ with each point displayed as a small dot. There should be enough points so that the graph is neither filled nor empty (it should look a uniform shade of gray from a distance). The histogram should be flat and the plot should uniformly fill all the space between 0 and 1. Optionally it is interesting to calculate the total number of points with $x_i^2 + x_{i+1}^2 < 1$ divided by the total number of points (the answer should converge to $\pi/4$ as the number of points is increased).

## 2 Monte-Carlo Simulation of Crystal Growth in 3D

Solid crystals can be grown by deposition from a vapor phase onto an existing crystal surface. Also when the surface of an object is coated with a metal or other material by evaporation in vacuum the surface grows by a more or less random deposition of atoms onto the surface. This problem will use the random number generator (above) to model crystal surface growth in a simplified random process. This simulation is a way to observe the dynamics of crystal surface growth which is diffifult to observe experimentally ( on a time scale of nano sec. etc.). Electron microscopes can observe the final crystal structure and X-ray observe a macroscopic average.

Consider a cross section of a 2D crystal surface of total size $N_x \times N_y$ that is initially flat (see Fig. 1). New atoms will be deposited on the surface and the crystal will grow in the $z$ direction. At each time step choose a point on the perimeter of the surface at random and either add an atom to it or let an adjacent atom diffuse into this location (using the rules described below). Repeat this process for many time steps. This method without diffusion is described in the 1D surface version by H. Gould and J. Tobochnik[4] and is called the Eden[2] model.

A crystal is modeled as a simple cubic lattice of points (square grid) in three dimensions (see figure 1). Only one type of atom is considered (single element). Each point in the 3D lattice is either occupied or vacant, and the state of each lattice position in the crystal should be stored in a single 3D array (may be a type char to save memory) as:

$$\begin{aligned} L(x,y,z) &= 0 \quad \text{vacant} \\ &= 1 \quad \text{occupied} \end{aligned} \tag{1}$$

The coordinates $(x, y, z)$ take on only integer values. Periodic boundary conditions are used for $x$ and $y$ but not for $z$ because $z$ should not be filled completely. This means that $x$= -1 is the same as $x = (N_x - 1)$ and $y$ = -1 is the same as $y = (N_y - 1)$. The crystal is initially a single uniform layer of atoms at $z = 0$:

$$\begin{aligned} L(x,y,z)_{INIT} &= 1 \quad \text{if} \quad z = 0 \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{2}$$

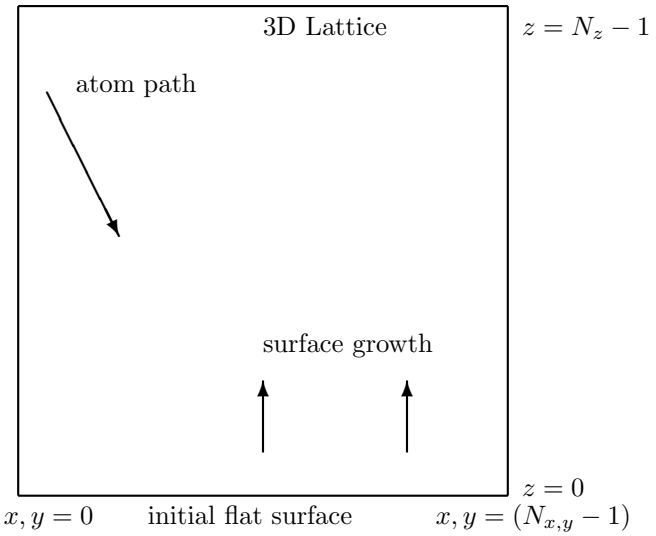


Figure 1: Cross section of 3D crystal growth geometry.

The surface is built up by randomly alternating between deposition and diffusion. Each deposition phase is essentially a simple Eden model[2, 4, 5] using a random selection of an unoccupied perimeter site, (defined as a site that is currently unoccupied but at least one of its nearest neighbors is occupied) which is then occupied with sticking probability $p_0$. Each diffusion phase is a simple random walk to an unoccupied nearest neighbor. Deposition and diffusion use separate sticking probabilities $p_0$ and $p_1$ respectively which may or may not be equal. The total probability of an atom sticking in a new site is summed over all occupied nearest neighbors:

$$P_{TOTAL} = \sum_{i,j,k=-1}^{+1} L(x+i, y+j, z+k) \frac{p_x}{P_{MAX}\sqrt{i^2+j^2+k^2}} \tag{3}$$

where $p_x = p_0$ for deposition or $p_1$ for diffusion and the i=j=k=0 position is excluded. $P_{MAX} = 6 + \frac{12}{\sqrt{2}} + \frac{8}{\sqrt{3}} = 19.1041$ is the value when all neighbors are occupied and normalizes the probability. It might be helpful to precalculate these probilities (for each position) and store in an array (to run a little faster).

The procedure is summarized as:

1. Initialize a large 3D array of lattice points as unoccupied (=0) except for the $z$=0 points (=1 on initial crystal surface).
2. Generate a list of all perimeter sites. A point is defined as a perimeter point if it is currently unoccupied and one or more of its 26 nearest neighbors is occupied. Store a set of $(x, y, z)$ coordinates for all of these perimeter points (at this time step) in three (large) 1D arrays.
3. Choose whether to deposit or diffuse with probability $P_{DEP}$. (If $P_{DEP} > R$ where $R$ is a uniformly distributed random number between 0 and 1 then deposit a new atom otherwise diffuse an existing atom.)

4. If deposit, choose one perimeter site at random and occupied it with probability $p_0$ summed over all occupied nearest neighbors. (If $P_{TOTAL} > R$ where $R$ is a uniformly distributed random number between 0 and 1 then deposit a new atom.)
5. If diffusion, choose one perimeter site at random and also one of its nearest neighbors at random. If this nearest neighbor is occupied move it to the perimeter site if the new site has a higher sticking probability (larger $P_{TOTAL}$) or with probability $p_1$ summed over all occupied nearest neighbors. (If $P_{TOTAL} > R$ where $R$ is a uniformly distributed random number between 0 and 1 then diffuse.) Don't move if the new position makes a zero sticking probability in its nearest neighbors. The initial z=0 surface is held fixed.
6. Repeat all steps except the first until enough new atoms have been deposited.

This procedure uses a single Monte-Carlo cycle and randomly selects whether it is a deposit or diffusion phase[1]. This procedure will generate both voids and overhangs. It is a little unphysical in that the voids appear in the perimeter list and can be populated in the deposition phase even if they are completely surrounded (would require atomic rearrangement or tunneling). Also the diffusion process may create isolated groups of atoms (but no isolated single atoms). This should be rare however.

At each step the average height $h(t)$ and roughness or width $w(t)$ of the surface should be calculated. If the height of each column $h_{x,y}$ is defined as the maximum $z$ value that is occupied at each $x, y$ coordinate, then the average height and square of the roughness are as defined as:

$$h = \frac{1}{N_x N_y} \sum_{xy} h_{x,y} \tag{4}$$

$$w^2 = \frac{1}{N_x N_y} \sum_{xy} (h_{x,y} - h)^2 = \frac{1}{N_x N_y} \sum_{xy} h_{x,y}^2 - \left( \frac{1}{N_x N_y} \sum_{xy} h_{x,y} \right)^2 \tag{5}$$

Note that $w$ is just the standard deviation of the height.

Use the above procedure to calculate the growth of a small crystal of size $N_x \times N_y \times N_z = 50 \times 25 \times 25$. You may use the 3D array class available on the course web site (similar to the 1D and 2D class already discussed) if you wish. Run your program until $N_A$ =10 monolayers have been deposited (i.e. until 25*50*$N_A$ atoms have been deposited). Use sticking probabilities of $p_0 = p_1$=0.01, a probability of deposition vs. diffusion of $P_{DEP}$=0.5 and the random number generator you tested above. Make a graph of the average height $h(t)$ and the roughness $w(t)$ as a function of time $t$ (= number of Monte Carlo steps). Also plot a cross section of the crystal with the $y$ coordinate in the middle of the crystal. For this value of $y$ plot circles for each $(x, z)$ position in the crystal that is occupied (do not connect the circles, which represent atoms). It is best to plot with equal aspect axes (so each cell looks square). Although there may be a better way, in python, try something like:

```
subplot(111, aspect='equal')
plot( x, y, 'ko' )
```

Does this Monte-Carlo simulation produce a believable random surface?

HINT: You can make the coordinates obey periodic boundary conditions using

```
inline int periodic( int i, int n ) { return( (i+n)%n ); }
```

OPTIONAL-1: Try different values of the sticking coefficients and branching probability for deposition vs. diffusion (smaller sticking probabilities should produce a denser packing).

OPTIONAL-2: Try the random number generator in the STL called, default_random_engine generator in <random>.

OPTIONAL-3: If the height in each (x,y) position (max. z occupied) is assigned an intensity then a 2D image of these intensities is another way to visualize the surface (a 2D image in x,y).

OPTIONAL-4: Comparison to Experiment

Fleet[3] et al have used pulsed laser deposit (PLD) to grow crystals of strontium titonate ($SrTiO_3$). A high powered laser beam evaporates a target material which is then deposited onto a clean crystal surface to grow a new material. This was done at the Cornell CHESS X-ray facility, and the surface growth was monitored using X-ray diffraction.

X-ray diffraction is usually used to determine the crystal structure of solid materials. A beam of X-rays illuminates the specimen and is scattered (or diffracted). The scattered X-rays interfere with each other to form a diffraction pattern. If the material is crystalline (a regular array of atoms) then a pattern of sharp spots (Bragg peaks) is formed. The position and spacing of these spots is related to the spacing in the crystal and can be used to determine the crystal structure. However, in this experiment, a position in the diffraction pattern in between these diffraction spots for this material was observed (anti-Bragg peaks). This position should normally yield no X-ray intensity for a perfect crystal but was used to experimentally observe the imperfections in the crystal during formation formation.

The X-ray diffraction intensity at a single point in the diffraction pattern (reciprocal space) $\vec{q} = (q_x, q_y, q_z)$ is:

$$I \propto \sum_{xyz} L(x,y,z) \exp\left[2\pi i(q_x x + q_y y + q_z z)\right] \tag{6}$$

Fleet[3] et al used position $(q_x, q_y, q_z)$=(0,0,0.5) in units of the crystal cell size. $x, y, z$ take on only integer values in this set of units, so the expression in the exponent takes on values of $0, \pi, 2\pi, \ldots$, and the exp() factor is either +1 or -1. If layers of the crystal occur in pairs then this intensity sums to zero, so this intensity tracks the mismatch between pairs of atomic layers, and gives some information about the dynamics of crystal formation.

The simulation you performed above can easily yield a comparison to this experiment. At each time step calculate the above X-ray intensity at the (0,0,0.5) position and plot it versus time (Monte-Carlo steps).

## References

[1] Joaquin Cortes and Paulo Araya, "Monte Carlo Simulations of Adsorption on Heterogeneous Surfaces with a Random Topography of Surface Sites", J. Chem Phys. 95 (1991) , p. 7741-7744.

[2] M. Eden, in: *Proc. 4th Berkeley Symposium on Mathematical Statistics and Probability, Vol. IV*, ed. J. Neyman (Univ. of Calif., Berkeley) p.223.

[3] Aaron Fleet, Darren Dale, Y. Suzuki, and J. D. Brock, "Observed Effects of a Changing Step-Edge Density on Thin-Film Growth Dynamics" Phys. Rev. Lett. 94 (2005) 036102.

[4] H. Gould and J. Tobochnik, *An Introduction to Computer Simulation Methods, 2nd edit*, Addison-Wesley 1996, Section 14.3, page 487.

[5] David P. Landau and Kurt Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, Cambridge Uiv. Press 2000

[6] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, J. Chem. Physics, 21 (1953) p.1087.

[7] W. H. Press, S. A, Teukolsky, W. T. Vetterling and B. P. Flannery *Numerical Recipes, The Art of Scientific Computing, 3rd edit.*, Camb. Univ. Press 2007.