Monte Carlo Calculations

HW 9: Friday, Nov. 15, 2019
DUE: Friday, Nov. 22, 2019
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
chap. 14 parallel processing

1. **FINAL PROJECT OUTLINE DUE ON MON. NOV. 25, 2019**: 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 MON. DEC 16, 2019** (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 0.68 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[5] (called the same way as Ran.doub() on page 342). Do not use the integer valued member functions to get small integers (by masking out the low order bits) 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 Protein Folding in 2D

Proteins perform many important biological functions. A protein is formed by linking a sequence of amino acids together in a chain. Each amino acid is a medium sized organic molecule. A simplified model of a protein is to imagine a string of beads. Each bead is an amino acid and the string connecting the beads is the covalent bond that attaches adjacent amino acids. Each amino acid (bead) is strongly bonded to only two other amino acid (the one directly in front and back). In some types of chemical environment the protein may be stretched out in a straight line. In most biological situations (usually in a water solution) it might be curled up into a compact structure. The process of going from an extended straight line to a final curled up form is called the protein-folding problem and is still an active area of research. This homework will focus on the physics of the folding process (in a simplified form) and not the detailed chemistry or biology of the protein.

There are a total of 20 different types of amino acids, and a typical protein can have several hundred amino acids. The order of the amino acids and the final folded configuration determine the function of the protein. A detailed accounting of all of the atoms and electrons in a protein and its chemistry is a considerable task. However a very simplified model (as follows) might give some clue as to how some proteins go from an extended state to a final folded state. What follows is modeled after the discussion given in section 11.1 of Giordano[1] on two dimensional folding. Šali[6] et al have discussed a similar problem in three dimensions if you are interested.

For simplicity assume that the protein is restricted to only two dimensions (coord. x and y) and that it is not allowed to cross over itself. Also assume that the covalent bond between adjacent amino acids (beads on the string) is fairly strong and cannot be broken but has some flexibility in angle (remember that each amino acid has two covalent bonds to adjacent amino acids). The angle between adjacent amino acids will be restricted to only 0, 180 or ± 90 degrees, and the distance between all adjacent amino acids is the same. This means that the protein must lie on a square grid (as in figure 1). Both the x and y coordinates of each amino acid must be integer multiples of some value. For simplicity work in units of this value, so that the coordinates of each amino acid are integer values. Each amino acid must fall on a grid point and the bond between adjacent

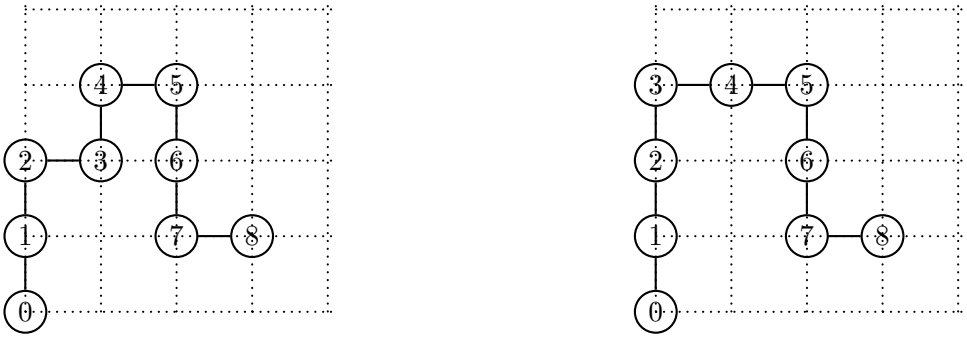

Figure 1: Two states of a folded protein on a 2D square grid. Each circle is one amino acid and each solid line is a covalent bond. Left is before and right is after an allowed move of one amino acid subunit (number 3).

amino acid is symbolized as a horizontal or vertical line between grid points. This means that a given protein can take on many different possible shapes. The preferred shape may be influenced by the temperature and the chemical environment.

When the protein bends around there is an additional interaction between the amino acids as they approach other amino acids that they are not covalently bonded to. If they get close enough they may feel an additional van der Waals type force (attractive) between each other. For simplicity, neglect all but the nearest neighbor interactions. The net change in energy due to folding is:

$$E_{Total} = \sum_{\text{all pairs}} E_{t(i),t(j)}\delta_{ij} = \frac{1}{2}\sum_{j\neq i} E_{t(i),t(j)}\delta_{ij} \tag{1}$$

where $E_{t(i),t(j)}$ is the interaction energy between amino acid of type $t(i)$ and type $t(j)$ (usually negative), and δ_{ij} is 1 if amino acids i and j are separated by one grid square in a horizonal or vertical direction (but not diagonally) and are not covalently bonded (i.e. not next to each other on the chain) but $\delta_{ij} = 0$ for all other conditions. For example see the short protein in figure 1 with nine amino acids (circles). The horizontal and vertical solid lines are the covalent bonds. In the left hand form of this protein only amino acids labeled 3 and 6 are not covalently bonded (not next to each other in the chain) and are separated by exactly one grid spacing, so only this pair contribute to the sum. When doing this sum be careful to count each pair only once. Note that E_{ij} is a 20×20 symmetrical matrix $E_{ij} = E_{ji}$. Also, the covalent bond energy between adjacent amino acids is assumed to remain constant on folding so this component of the energy will be ignored (i.e. its the same for all configurations).

This nearest neighbor interaction energy is similar in magnitude to the thermal energy, so this process will depend on temperature. The Monte Carlo nature of this solution is to generate random changes to the structure and evaluate the energy of the new configuration and search for the lowest energy configuration. The procedure for this calculation is as follows.

1. Initialize the interaction matrix E_{ij} to random values between E_{min} and E_{max} (both are negative). This is a symmetric matrix of 20×20 elements ($E_{ij} = E_{ji}$).
2. Initialize the protein structure to be be a straight line (can be horizontal or vertical) with N_L amino acids. Calculate the energy of this protein $E_0 = E_{Total}$ as per equation 1. Each type of amino acid is chosen at random (20 types total).

3. Generate a random change in the protein structure. First choose one amino acid at random. Then chose one of four possible moves for this amino acid at random (corresponding to ± 1 for the x and y coordinate, excluding the current position). If this new position is allowed (i.e. its not currently occupied and its separated from its two bonding partners by one in either x or y but not both), then keep this move and go to the next step. Repeat this step until an allowed move is found.

4. Calculate the energy of this new structure $E_{new} = E_{Total}$ using equation 1.

5. If $\Delta E = E_{new} - E_0 \leq 0$ keep this new protein structure (and skip next step).

6. If $\Delta E = E_{new} - E_0 > 0$ then accept this step with probability $p(\Delta E) = \exp(-\Delta E/(k_B T))$. To do this, calculate a random number R between 0 and 1. If $p(\Delta E) > R$ then accept this move otherwise reject it. (This is the Metropolis algorithm to calculate this probability distribution, which is the Boltzmann distribution for temperature T and k_B is Boltzmann's constant.)

7. If this new structure is accepted replace the current protein structure with the new one and set $E_0 \leftarrow E_{new}$. Repeat steps 3 through 6 for N_{step} steps.

You should write a program to implement this Monte-Carlo calculation for a protein of length N_L= 45 amino acids, a range of interaction energies E_{min}= -7 to E_{max}= -2 and a temperature corresponding to $k_B T$= 1 (in units of the bonding energy). Run this program for 10 million Monte-Carlo time steps (one step is one pass through the above procedure). Each step probably corresponds to a typical atomic or molecular relaxation time on the order of a picosec or so, which gives us some indication of how long the protein might take to fold up to its final (lowest energy) form. Plot the energy and the end-to-end distance versus time steps. You only need to print out about 500 points in this range not all 10 million values. (Hint the energy should decay exponentially and then level off at some value with random thermal fluctuations about this value.) Also plot the actual structure at a time step of 10^4, 10^5, 10^6, and 10^7. A Monte-Carlo time step is defined as one pass through the algorithm above. You can draw this structure by simply plotting a line through all coordinates of the amino acids (in order) and also a large symbol (circle) at each coordinate.

If you start with a large temperature and gradually reduce the temperature then this is called the simulated annealing method (see section 10.12 of Numerical Recipes[5]). Simulated annealing is a general method of finding an optimal solution to a system with only discrete internal position (i.e. not continuously variable).

Although this problem has been stated in a bio-physics context, the general method can be adapted for a variety of other physics and engineering problems. For example optimizing the arrangement of electronics components on a printed circuit board can also be solved using simulated annealing.

OPTIONAL: Plot the average energy and/or size versus temperature for $k_B T$= 10 to about 0. This value should be the average over several thousand Monte-Carlo time steps after it has reached its equilibrium value.

OPTIONAL: Try different starting point RN seeds to see how the final structure changes.

OPTIONAL: Try different lengths and total times (longer lengths need longer times to compress).

Programming Hints You are not required to use the following programming strategy, but you may find some of the following ideas helpful. Store the protein structure as an array of 3 by N_L elements. Adjacent elements in the array are covalently bonded. For each amino acid store its X and Y position and its type (3 integer values for each amino acid). Make three separate subroutine, one to calculate the distance between amino acids, one to calculate the total energy and one to make a random change in the structure and check whether it is geometrically allowed. Then loop over each Monte-Carlo time step and test the energy conditions as above.

Note, you may calculate a separation as $|x_i - x_j| + |y_i - y_j|$ using all integer arithmetic, which is faster and allows for an equality check for exactly 1 for nearest neighbors. (The function abs() is the integer absolute value function.)

References

[1] N. J. Giodano and H. Nakanishi, *Computational Physics, 2nd edit.*, (Prentice-Hall 1997,2006).

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

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

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

[5] 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.

[6] A. Šali, E. Shakhnovich and M. Karplus, "How does a protein fold?", Nature, Vol. 369 (1994) p.248.