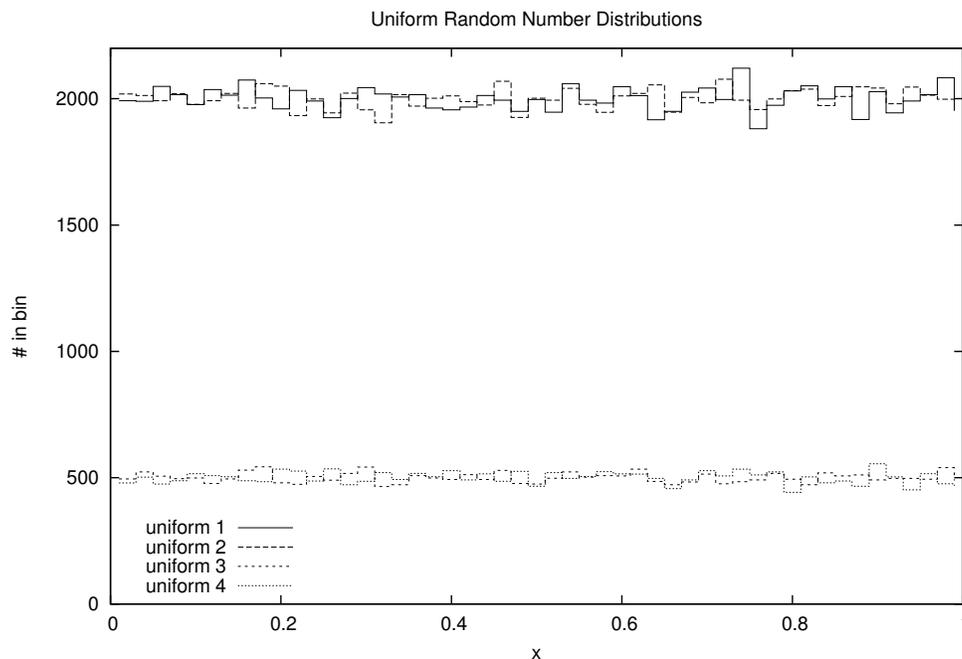


13. 6810 Session 13

a. Follow-ups to Session 12

- **Histograms of Uniform Random Number Distributions.** Here is a typical figure you might get when histogramming uniform distributions using gnuplot (using `plot ...with steps` to get the histogram look): Two sets of numbers have an average of 2000 counts per

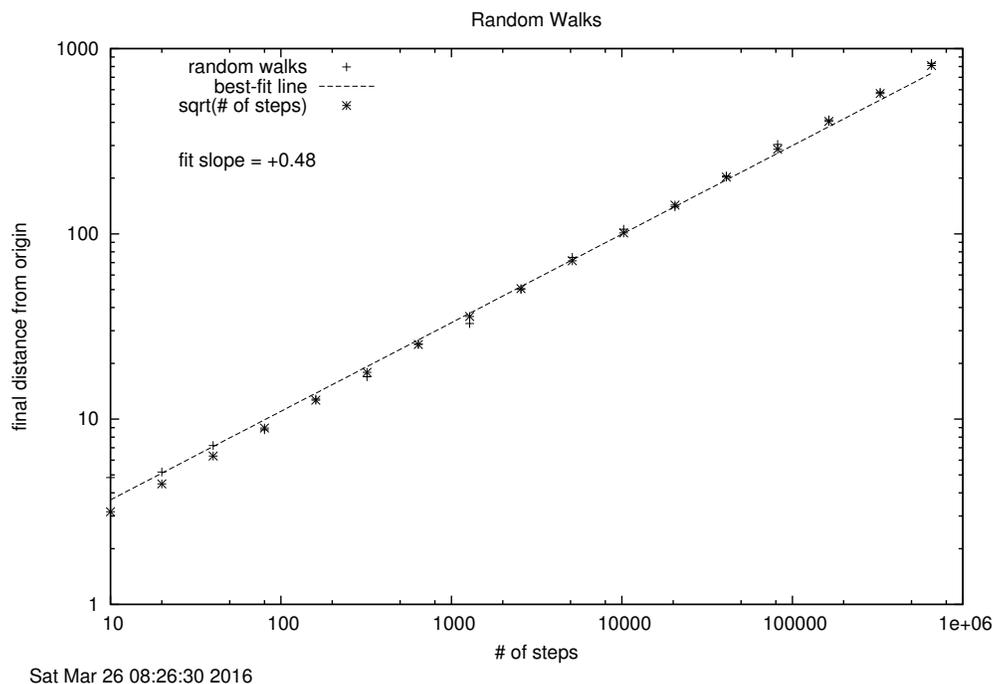


Sat Mar 26 08:33:30 2016

bin and two sets have an average of 500 per bin. The size of the *fluctuations* should scale as $\sqrt{\#}$ of counts in a bin, which is evident in the figure. That is, the fluctuations of the top sets look roughly twice as large as those of the bottom set and a standard deviation of 40–50 looks about right. How would you check those numbers more carefully?

- **The Length of Random Walks.** In Session 12 you explored how the final distance R from the origin of a random walk scales with the number of steps N . Remember that there is no definite relationship for any *individual* walk, but only a statistical relationship that arises by averaging R for a given N over many walks. (So any given walk could end up very close or very far from the origin.) On the next page is a graph that you might obtain if you average the result at each N over a sufficiently large number of walks (say 100). It shows a good fit to the expected \sqrt{N} dependence!
- **The story of RANDU** [from the Wikipedia article]

RANDU is the name of a random number generator that was commonly used on mainframe computers in the 1960's. Technically it is a linear congruential pseudorandom number generator of the ParkMiller type (see the Wikipedia article for more details) that is famous for being



not-so-random! In Donald Knuth’s famous book on *The Art of Computer Programming*, he writes: “its very name RANDU is enough to bring dismay into the eyes and stomachs of many computer scientists!” You would need a good eye to see that it would fail the simple “eye” test we used in Session 12 (where you were supposed to plot every two random numbers as x - y coordinates). Figure 1 compares a good RNG (left) to RANDU (right) using something like the test we did (see the webpage for details). If you look carefully (or maybe from a distance), you can see that the supposedly random pixels from RANDU tend to lie on parallel lines slanting up and to the right at about 20 degrees. However, there is no doubt that RANDU is a disaster if you generalize our test to three dimensions, plotting every three consecutive values as an x - y - z point. An example is shown in Fig. 2, where it is clearly seen that the points lie on planes rather than filling the cube uniformly.

b. Recap of Session 12: Monte Carlo Integration and Lead-in to Session 13

We explored in Session 12 how we could approximate an integral $\int_a^b f(x) dx$ by picking a random sequence of N values of x , which we label $\{x_i\}$, distributed uniformly on $[a, b]$.

- That is, the probability of choosing a number between x and $x + dx$ is

$$P_{\text{uniform}}(x) dx = \frac{1}{b-a} dx . \quad (13.1)$$

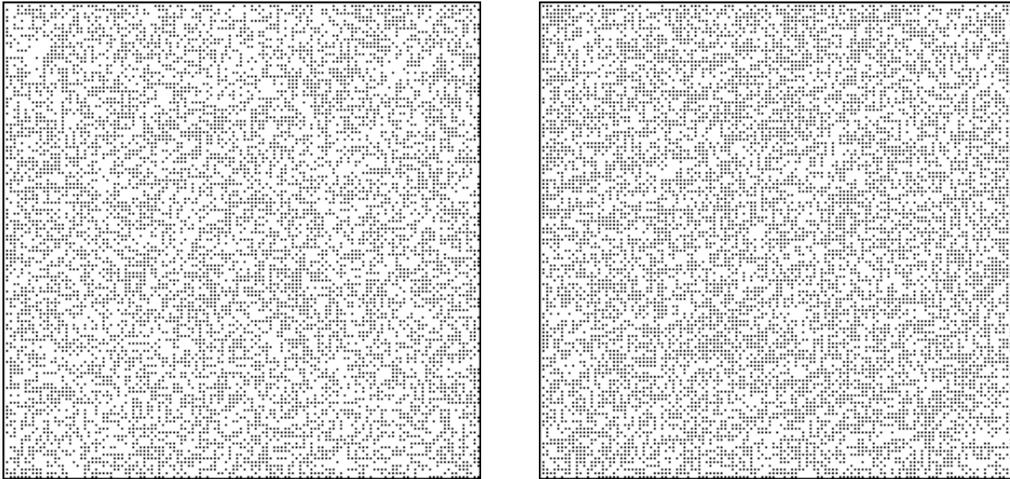


Figure 1: Figures from <http://www.causascientia.org/software/URandomLib.html> advertising the URandomLib random number generator. These are 256×256 pixel squares filled in using random numbers from URandomLib (left) and RANDU (right).

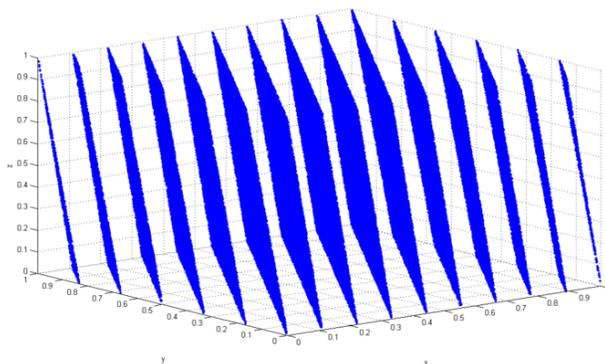


Figure 2: A three-dimensional plot of the points from 100,000 total pseudorandom numbers, with each three values used as an (x, y, z) point. Figure from Wikipedia (<http://en.wikipedia.org/wiki/RANDU>).

- Note that this PDF (probability distribution function) is *normalized*, i.e.,

$$\int_a^b P_{\text{uniform}}(x) dx = \frac{1}{b-a} \int_a^b dx = 1, \quad (13.2)$$

as you'd expect since the total probability must be unity.

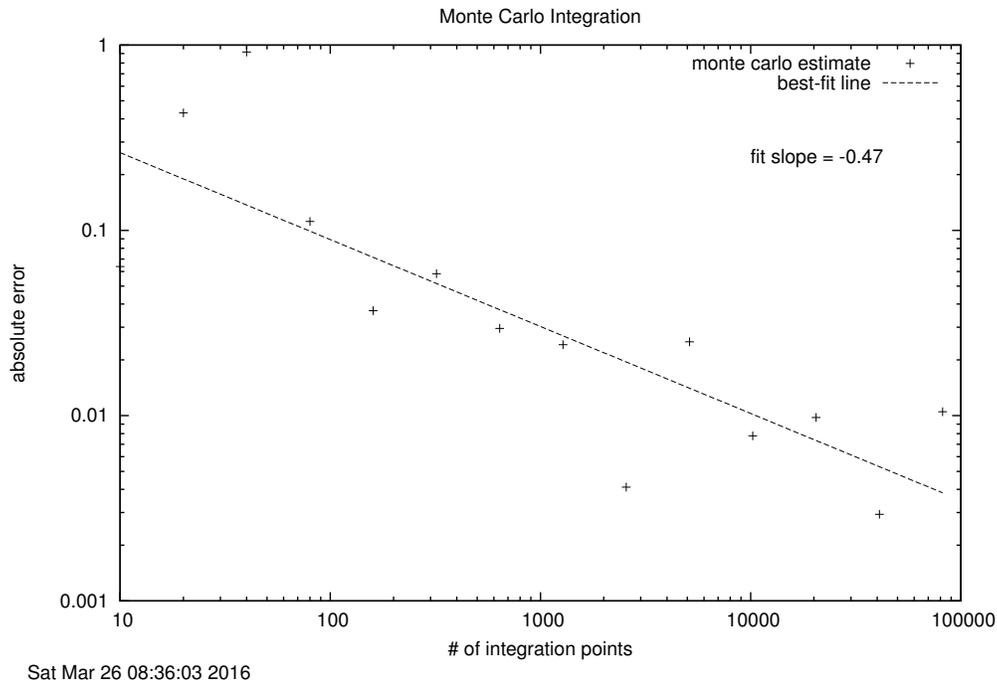
The Monte Carlo expression for the integral arises by approximating the *average* of $f(x)$ in $[a, b]$:

$$\langle f \rangle = \frac{1}{b-a} \int_a^b f(x) dx \approx \frac{1}{N} \sum_i f(x_i). \quad (13.3)$$

We can write this as

$$\int_a^b f(x) P_{\text{uniform}}(x) dx \approx \frac{1}{N} \sum_i f(x_i) . \quad (13.4)$$

You should have found that the accuracy of the integral improved statistically with the number of points. Here is a graph showing the results of 16 trials for each value of N :



The expected behavior is revealed by the fit (error improves like $1/\sqrt{N}$), but with large fluctuations. We should average over many more trials if we want a cleaner demonstration!

Now what if we used a different probability distribution $P(x)$? Then if our N values of x , which we'll label $\{\tilde{x}_i\}$, are distributed according to $P(x)$, the average of the $f(\tilde{x}_i)$'s yields an approximation to the integral over $f(x)P(x)$:

$$\frac{1}{N} \sum_i f(\tilde{x}_i) \approx \int_a^b f(x) P(x) dx . \quad (13.5)$$

This means that we can always divide a given integrand into a part that we treat as a probability distribution (called $P(x)$ here) and the rest (which is called $f(x)$ here).

- If we choose $P(x)$ so that $f(x)$ varies only relatively slowly with x , we will get an accurate estimate of the integral.
- This is called “importance sampling”.
- The generalization to many dimensions is immediate: we simply replace one-dimensional x by a multi-dimensional vector \mathbf{x} everywhere.

In Session 12 we saw that GSL can generate random numbers according to various one-dimensional PDF's, such as uniform and gaussian distributions. But the multi-dimensional integrals we'll want to do for physical problems will generally have a nonstandard $P(\mathbf{x})$. We will use the *Metropolis algorithm* to generate an appropriate distribution of $\tilde{\mathbf{x}}_i$'s.

c. Thermodynamic Properties of the Ising Model

If we want to calculate the thermodynamic properties of a system at temperature T (that is, in equilibrium) that has a Hamiltonian $\mathcal{H}(\mathbf{x})$, we have the problem of calculating a high dimensional integral with a rapidly varying integrand.

- Here “ $\mathcal{H}(\mathbf{x})$ ” is the energy of the system for a given *configuration* \mathbf{x} . A configuration is a specification of the state of the system in some relevant variables.
- For a quantum mechanical system of N spinless particles, a configuration could be the N position vectors.
- For a system of spins on a lattice, a configuration could be a specification of the spin projection at each lattice point.

We'll use the Ising model in one and two dimensions as our first example. The system consists of a set of lattice sites with spins, which take on only two possible values, $+1$ or -1 . The spins are arranged in a chain (one-dimensional Ising model) or a square lattice (two-dimensional Ising model). The interaction between spins is short-ranged, which is represented in our model by interactions only between *nearest neighbors*. That is, there is potential energy only from adjacent spins. Each spin has two nearest neighbors in one dimension and four nearest neighbors in two dimensions (diagonals don't count here!).

- We have to specify what happens at the edge of the system. E.g., do the ends of the one-dimensional chain have only one nearest neighbor, or do we imagine the chain wrapped into a continuous band? In the latter case, we have *periodic boundary conditions*, because it is equivalent to imagining an infinite chain that repeats itself periodically.

We also allow for an external constant magnetic field, H , which interacts with each of the spins individually to contribute a Zeeman energy. (Note: the notation H for the magnetic field is somewhat unfortunate because of the confusion with the Hamiltonian. In other references such as Landau and Paez, $H \rightarrow \mu B$ is used instead.)

We choose units so that the Hamiltonian takes the simple form:

$$\mathcal{H}(\mathbf{x}) = -J \sum_{\langle i,j \rangle} S_i S_j - \sum_i H S_i , \quad (13.6)$$

where $S_i = \pm 1$ is the spin.

- Here \mathbf{x} is just a shorthand for a specification of the spin S_i on each site. For a one-dimensional Ising model with N sites, you could store this as an array of length N with each element either $+1$ or -1 .

- The notation $\langle i, j \rangle$ stands for “nearest neighbor” pairs of spins. Obviously the product $S_i S_j$ can be either +1 or -1.
- In many applications we’ll set the external field H to zero.
- The constant J (called the “exchange energy”) specifies the strength of the spin-spin interaction. For a ferromagnetic interaction, $J > 0$, while for an “anti-ferromagnetic interaction” $J < 0$. *What do you think is the physical origin of J for a ferromagnetic interaction?*
- The *magnetization* of the system is the average value of the spin, $\langle S_i \rangle$, where the average is taken over the entire lattice. If the spins are equally likely to be up (+1) as down (-1), then the net magnetization is zero. In an external magnetic field, up or down will be favored energetically (if $H > 0$ in $\mathcal{H}(\mathbf{x})$ above, which is favored?), and there will generally be a net magnetization. In a ferromagnet at sufficiently low temperature, there will be *spontaneous* magnetization even in the absence of an external field. This may occur over regions (“domains”) or over the entire lattice.

A configuration in the Ising model is a *microstate* of the system. *If we have N sites in a one-dimensional Ising model, how many possible configurations (or microstates) are there? How many different values of the energy (with $H = 0$)?*

The thermal average of a quantity A that depends on the configuration \mathbf{x} (examples of A are the energy per degree of freedom $E = \langle \mathcal{H} \rangle_T / N$ or the magnetization per degree of freedom $M = \langle \sum_i S_i \rangle_T / N$) is given by the *canonical ensemble* (the denominator is the *partition function* Z)

$$\langle A(\mathbf{x}) \rangle_T = \frac{\int d\mathbf{x} A(\mathbf{x}) e^{-\mathcal{H}(\mathbf{x})/kT}}{\int d\mathbf{x}' e^{-\mathcal{H}(\mathbf{x}')/kT}} = \int d\mathbf{x} A(\mathbf{x}) P_{\text{eq}}(\mathcal{H}(\mathbf{x})), \quad (13.7)$$

where we have pulled out the Boltzmann factors to identify the probability distribution function $P_{\text{eq}}(\mathcal{H}(\mathbf{x}))$:

$$P_{\text{eq}}(\mathcal{H}(\mathbf{x})) = \frac{e^{-\mathcal{H}(\mathbf{x})/kT}}{\int d\mathbf{x}' e^{-\mathcal{H}(\mathbf{x}')/kT}}. \quad (13.8)$$

(*Is this PDF normalized?*) In the Ising model example, the integration over \mathbf{x} is just a sum over the spin configurations. If our configurations are chosen as eigenstates of energy (which they are in the Ising model example), then we can just sum over energies rather than every configuration:

$$\langle A \rangle_T = \frac{\sum_E (\# \text{ of states with energy } E) A(E) e^{-E/kT}}{\sum_{E'} (\# \text{ of states with energy } E') e^{-E'/kT}} = \sum_E A(E) P(E). \quad (13.9)$$

If we can construct a set of N configurations $\{\tilde{\mathbf{x}}_i\}$ (a statistical sample) that are distributed according to P , then we can apply our Monte Carlo method (i.e., we will be doing importance sampling) and $\langle A \rangle_T = (1/N) \sum_i A(\tilde{\mathbf{x}}_i)$. This is what the Metropolis algorithm does for us!

d. Metropolis Algorithm

The first thing to get straight is that the Metropolis algorithm has nothing to do with Superman or Fritz Lang :). It is named for the first author on the paper that described the algorithm. (A bit of trivia: another author on the paper is Edward Teller.) The Metropolis algorithm generates a *Markov chain*; this is a sequence of configurations \mathbf{x}_i that will be distributed according to the canonical distribution (i.e., the Boltzmann factor will tell us the relative probability of any configuration).

The Markov process constructs state \mathbf{x}_{l+1} from the previous state \mathbf{x}_l according to a *transition probability* $W(\mathbf{x}_l \rightarrow \mathbf{x}_{l+1})$. The idea is to construct W such that in the limit of a large number of configurations, the distribution of states \mathbf{x}_i approaches the equilibrium Boltzmann distribution P_{eq} from Eq. (13.8).

A sufficient condition for this to happen is that the principle of *detailed balance* should hold. In equilibrium, the *rate* of $\mathbf{x} \rightarrow \mathbf{x}'$ should equal the rate of $\mathbf{x}' \rightarrow \mathbf{x}$, or else it is not equilibrium! In calculating these rates, there are two terms to multiply:

$$\left(\frac{\text{probability}}{\text{time}} \text{ of } \mathbf{x} \rightarrow \mathbf{x}' \right) \times (\# \text{ of } \mathbf{x}'\text{'s}) = \left(\frac{\text{probability}}{\text{time}} \text{ of } \mathbf{x}' \rightarrow \mathbf{x} \right) \times (\# \text{ of } \mathbf{x}'\text{'s}) \quad (13.10)$$

or

$$W(\mathbf{x} \rightarrow \mathbf{x}') \times N P_{\text{eq}}(\mathbf{x}) = W(\mathbf{x}' \rightarrow \mathbf{x}) \times N P_{\text{eq}}(\mathbf{x}') , \quad (13.11)$$

which is detailed balance. This means that

$$\frac{W(\mathbf{x} \rightarrow \mathbf{x}')}{W(\mathbf{x}' \rightarrow \mathbf{x})} = \frac{P_{\text{eq}}(\mathbf{x})}{P_{\text{eq}}(\mathbf{x}')}. \quad (13.12)$$

But we know that the probability of configuration \mathbf{x} is

$$P_{\text{eq}}(\mathbf{x}) = \frac{1}{Z} e^{-\mathcal{H}(\mathbf{x})/kT}. \quad (13.13)$$

Any choice of W satisfying Eq. (13.12) should work, in principle.

For example, we can take

$$W(\mathbf{x}_a \rightarrow \mathbf{x}_b) = \begin{cases} \frac{1}{\tau_s} e^{-\delta E/kT} & \text{if } \delta E = E_{\mathbf{x}_b} - E_{\mathbf{x}_a} > 0 \\ \frac{1}{\tau_s} & \text{if } \delta E = E_{\mathbf{x}_b} - E_{\mathbf{x}_a} \leq 0 \end{cases} \quad (13.14)$$

with τ_s arbitrary for now (set it equal to unity for convenience). We can check that it works by considering all possible outcomes, as shown in this chart:

	$\delta E_{\mathbf{x} \rightarrow \mathbf{x}'}$	$W(\mathbf{x} \rightarrow \mathbf{x}')$	$e^{-E/kT}$	δE	$W(\mathbf{x}' \rightarrow \mathbf{x})$	$e^{-E/kT}$
$E_{\mathbf{x}'} > E_{\mathbf{x}}$	> 0	$\frac{1}{\tau_s} e^{-(E_{\mathbf{x}'} - E_{\mathbf{x}})/kT}$	$e^{-E_{\mathbf{x}}/kT}$	< 0	$\frac{1}{\tau_s}$	$e^{-E_{\mathbf{x}'}/kT}$
$E_{\mathbf{x}'} < E_{\mathbf{x}}$	< 0	$\frac{1}{\tau_s}$	$e^{-E_{\mathbf{x}}/kT}$	> 0	$\frac{1}{\tau_s} e^{-(E_{\mathbf{x}} - E_{\mathbf{x}'})/kT}$	$e^{-E_{\mathbf{x}'}/kT}$

If you do the multiplications, you’ll see that detailed balance is satisfied in every case.

Section 2.2 of the Binder/Heerman excerpt describes an implementation of Metropolis for the Ising model. The basic idea is that from a starting configuration, one generates a candidate for a new configuration somehow (for example, by flipping one randomly selected spin). After calculating the energy change between new and old, one has the transition probability W from Eq. (13.14). Generate a random number between 0 and 1; if the number is less than $\tau_s W$ keep the new configuration but otherwise flip the spin back and keep this as the “new” configuration.

Let’s be extremely explicit about how we can implement this. Take $\tau_s = 1$. We have a starting configuration with energy E_a and a new candidate configuration with energy E_b . If $E_b < E_a$, then we accept the new candidate always (i.e., with probability 1). That is, if the energy is lowered, we always keep the new one. This agrees with Eq. (13.14). If the energy is raised, namely $E_b > E_a$, we keep it *some of the time*. [You might think we should always just go “downhill” in energy, but we know this is wrong: if so, we would not have a chance to follow any path through the configuration space (because we would never go uphill), so we would not have ergodicity.] Following Eq. (13.14), we should keep it with probability $e^{-\delta E/kT} = e^{-(E_b - E_a)/kT}$. This is a number between 0 and 1. Say it is 0.3. Then we generate a random number uniformly distributed between 0 and 1. If it is less than 0.3 we keep the new configuration; if not, we reject it. (To see that this is the correct procedure, imagine doing it 1000 times. You would expect roughly 300 times to get a number less than 0.3, so keeping those would give the desired probability.)

We’ll go over various implementation issues in this Session and the next. These include:

- **Equilibration.** If we are going to use configurations generated by Metropolis to calculate thermal averages, we want to make sure we have reached “equilibrium.” How long does this take and what is the signature?
- **Efficiency.** The initial two-dimensional Ising model code has several inefficiencies that are generic to many types of Monte Carlo simulations. In Session 14 we’ll look at how to optimize to gain a factor of six in speed.
- **Autocorrelation.** When evaluating an average over Monte Carlo configurations, we want to skip the first n_1 steps and then use data taken every n_0 Monte Carlo steps. How do we determine how large to take n_0 and n_1 ? The *autocorrelation function* can be used to determine a reasonable choice for n_0 .

e. References

- [1] R.H. Landau and M.J. Paez, *Computational Physics: Problem Solving with Computers* (Wiley-Interscience, 1997). [See the 6810 info webpage for details on a new version.]
- [2] M. Hjorth-Jensen, *Lecture Notes on Computational Physics* (2009). These are notes from a course offered at the University of Oslo. See the 6810 webpage for links to excerpts.
- [3] W. Press *et al.*, *Numerical Recipes in C++*, 3rd ed. (Cambridge, 2007). Chapters from the 2nd edition are available online from <http://www.nrbook.com/a/>. There are also Fortran versions.