Due date: Thursday, September 26

Starting with this second homework assignment you will get a few problems that expand on topics from the lectures, and at least one problem that builds on previous work on the Hadamard model.

1. Estimate the effective hard-sphere radius[1] of the Helium atom knowing only that (i) the solid and gas[2] phases coexist[3] at 297 K and 115 kbar , and (ii) the solid/gas transition in the hard-sphere model occurs at dimensionless pressure $p^* = 11.56$.

2. Let u and v be two elements of an $n \times n$ orthogonal matrix U. Find the joint probability distribution $f(u, v)$ for the uniform measure on U for three cases of u and v: (i) they are the same element, (ii) they are distinct but lie on the same row or column, (iii) they lie on distinct rows and columns. Only work out the limiting form for $n \to \infty$ and ignore normalization.

3. Analytically calculate the $\alpha = 1$ Hadamard-model partition function Z to order β^2 (leading terms of the "high temperature expansion"). You will need to use the three results above, for the joint distribution of matrix elements. By taking derivatives of $\log Z$ obtain the mean energy to order β and heat capacity to order β^2. As a check, these should both scale as n^2 (the naive "volume" of the system). With n^2 divided out we will denote these $e(\beta)$ and $c(\beta)$. In a future assignment you will use these results to check your Hadamard simulation at high temperatures.

[1] Radius is half the diameter.

[2] I prefer to refer to the non-solid phase as a gas, not a liquid, because it is not self-bound (prefers a particular density). However, at these high pressures this gas of Helium atoms is very far from ideal.

[3] Pinceaux, J-P., J-P. Maury, and J-M. Besson. "Solidification of helium, at room temperature under high pressure." *Journal de Physique Lettres* 40.13 (1979): 307-308.

4. Learn how to wrap the efficient Givens-rotation function you worked out in the last assignment in a working piece of executable code in the language of you choice. Estimate the energy $e(\beta)$ and heat capacity $c(\beta)$ at $\beta = 5.5$ for $n = 12$. Demonstrate that your sampling time T for these averages is sufficient (greater than the mixing time) by checking that the statistical error for sampling time NT decreases as $1/\sqrt{N}$.

   Here are some general remarks meant to make both your work and that of the grader easier!

   - Keep your code simple! At this stage we are just seeking consensus on two numbers. Worry about the fancy user-interface in the next assignment.
   - We are open to all reasonable options for optimizing the Markov-chain "engine." If you like Python, you should consider some of the tips found here:

     `https://wiki.python.org/moin/PythonSpeed/PerformanceTips`

     It's even possible that some combination of Numpy methods can be hacked to build an efficient engine. If so, it will be interesting to see how that performs compared to C. If you've gone to the trouble of learning enough C to write the engine, then it's actually not that much additional work to write the wrapper in C as well (but beware that declaring, allocating, printing, and reading input are much more cumbersome than in Python). Finally there are now things such as Cython, Pyrex, etc. that offer a way to integrate compiled low-level code with Python. If anyone follows that route, and the overhead is low, please let the rest of us know!
   - Don't compute $c(\beta)$ as a finite difference, but work out analytically something the Markov chain can average directly.
   - You might want to include the acceptance/rejection step in your compiled function, as in the solution for the first assignment, as that avoids passing the row/column pair as arguments (another source of Python overhead).
   - Here's a way to combine (i) eliminating initialization transients and (ii) convergence testing. Fix the number of sampling blocks N, something not too small or large, like $N = 20$. Try a series of sample

block sizes T, starting with something small, like $T = 10^3$ (for immediate feedback). In each block compute your averages and also their errors. Disregard these in the first block ($N = 0$) because results will be skewed by the initialization. You will know that T is large enough when the averages in the remaining blocks look like they are independently distributed and not showing drift. Report the average of the blocks and the standard error when that is achieved. Increasing T (or N) further will then systematically reduce the error.

- The only optimization knob at your disposal is selecting the range of the Givens angle in each proposed transition — use it! When set poorly the acceptance probability a will be either very small or nearly 1. In my code I adjust the range dynamically: when a is greater than 50% I make slightly more aggressive proposals by multiplying the range by a factor slightly greater than 1, when below 50% I decrease the range by a factor slightly less than 1. All this is done in block $N = 0$ (another reason not to take the results of this block seriously) and the range is fixed in the subsequent blocks. Print out the range parameter that was selected. Later, when we vary β, it will be interesting to see how the optimized range varies.
- It is common practice to apply elementary transitions to the entire system — a "sweep" — between computing averages. For the Hadamard model a sweep comprises proposed Givens rotations to all pairs of rows and all pairs of columns. The number of sweeps will be our Markov chain "time" rather than the number of Givens rotations.