Due date: Thursday, September 26

1. Find a computer on which you can compile and run the hard matrix model simulation program `hmm.c`. You only need the standard C programming libraries. If you have the GNU C compiler on your system, the program is compiled with this command

   ```
   gcc -O3 hmm.c -lm -o hmm
   ```

   The flag `-O3` optimizes the compiled machine instructions, `-lm` calls the math library (for `fabs( )` and `sqrt( )`), and `-o hmm` is the way to give your compiled program the name `hmm` (but you may call it whatever you wish).

   Next, try running the program to simulate the HMM for $n = 12$ and $\beta = 5.5$. Here is the command line for that:

   ```
   ./hmm 12 5.5 1000 5 test & <return>
   ```

   If you forget the order of arguments, just run the command without any arguments

   ```
   ./hmm <return>
   ```

   and you will get the reminder

   ```
    expected 5 arguments:  n beta sweeps runs outfile
   ```

   In the command above we specified 1000 sweeps, 5 runs, and the name `test` for the output file. The `&` at the end of the command line puts the "job" in the "background", so you can do other things on the computer while it's simulating the HMM. Assuming things went fine, the file `test` will have the parameters in the first line followed by

   run-number energy heat-capacity rubicons Givens-angle

   in each of five lines (the number of runs). The first three are clear and the rubicon number is just the average number of matrix elements that change sign in one sweep. The program adjusts the maximum Givens angle (last number in each line) so the acceptance rate of the transitions is 50%.

   To demonstrate that you are able to use the program, locate the β's of the first order transitions for $n = 12$ and $n = 16$. Hadamard matrices exist for

those orders, so there can be a kind of equilibrium between a "fluid" and "crystal" (Hadamard) phase, marked by a maximum in the heat capacity. The peak gets both higher and sharper in the thermodynamic limit. One million sweeps are sufficient to find the $n = 16$ transition. In 2019, with one billion sweeps, we were even able to locate the $n = 20$ transition.

2. The program `hmm.c` represents the simplest experimental protocol. There is nothing like heating or cooling, where the temperature is incremented or decremented at regular intervals. The user sets the inverse temperature and the program equilibrates and measures the model at that same temperature. This makes it convenient to do multiple experiments at different temperatures β, or different n, just by issuing multiple command lines with suitably names output files. Current computers can run multiple "threads" simultaneously, so you might be able to run 16 experiments in the same time it takes to run just one.

   The results of the "runs" are not aggregated in any way, say by averaging. Think of each run as an experiment in an actual lab on a real system. The different outcomes let you assess whether the system equilibrates to have unique averages (energy, heat capacity), or shows signs of non-equilibrium behavior (e.g. gets stuck in nonunique states). Five runs (at the same n and β) is plenty. If the five measurements are just as dispersed with 10^7 seeps as with 10^6, then your matrices are probably getting stuck because the rubicon number is too low.

   Show that you understand how `hmm.c` implements the simple experimental protocol by adding comments. How is the system initialized? How is the Givens angle optimized during equilibration? What is the point of the function `orthogonalize()`?

3. The Constellation Graph Model (CGM) is the 1-nearest-neighbor graph model (of contemporary data science) for the special case that the data nodes are uniformly distributed in the plane[1]. In lecture we learned that the connected clusters (of stars) are always trees, and that each tree always has a unique pair of stars that are mutual nearest neighbors. Call the distinguished stars *core stars*. The average constellation "size" can be defined as

$$s = \frac{\text{total number of stars}}{\text{total number of constellations}} = \frac{2}{p_{\text{core}}} \ ,$$

[1]The sky, when the range of angles is small.

where $p_{\rm core}$ is the probability any randomly selected star is a core star. Calculate s by calculating $p_{\rm core}$. You will need to use the Poisson distribution, which says that if stars have uniform areal density σ, then the probability of finding k stars in a region of area A is

$$p_k = \frac{(\sigma A)^k}{k!} e^{-\sigma A}.$$

4. In lecture we studied a quantity of central importance in the Random Graph Model (RGM), the fraction of nodes in finite connected clusters:

$$f = \frac{1}{n} \sum_{k=1}^{\infty} k \langle n_k \rangle \ . \tag{1}$$

We were able to express f as a function of the mean degree d, both below and above $d = 1$, in terms of Lambert's w function:

$$f(d) = \frac{w(de^{-d})}{d} \ . \tag{2}$$

The w function has power series

$$w(z) = \sum_{k=1}^{\infty} \frac{k^{k-1}}{k!} z^k, \tag{3}$$

converging for $|z| < 1/e$, which is all we need for any $d \neq 1$, and satisfies the implicit equation

$$z = w(z) e^{-w(z)}. \tag{4}$$

Here are two ways to define the average "size" of the finite connected clusters:

$$\begin{aligned} s_1 &= \left(\sum_{k=1}^{\infty} k \langle n_k \rangle \right) / \left(\sum_{k=1}^{\infty} \langle n_k \rangle \right) \\ s_2 &= \left(\sum_{k=1}^{\infty} k^2 \langle n_k \rangle \right) / \left(\sum_{k=1}^{\infty} k \langle n_k \rangle \right) . \end{aligned}$$

   (a) Contrast in *words* the two definitions of size. One of these coincides with how size was defined for the CGM, which one?

   (b) Using (1)-(4) express s_1 and s_2 explicitly in terms of the w function.

   (c) Re-express s_1 and s_2 even more explicitly in terms of just d and the dual degree $\tilde{d}(d)$, depending on whether $d < 1$ or $d > 1$. Sketch the two "sizes" as a function of d.