Ising Min\_Energy 6/30/2020

Brendan Philbin, Chris Herdman

In attemptSwap()

* Rather than having a calculation of dE manually, make a function that computes the energy difference dE
  + This function will use the computeEnergy function, as we will need that for the min\_energy tracking
* Calculating dE really only needs one column or row of Jij matrix in order to calculate
  + Because we are only dealing with flipping one spin
  + dE between these rows = dE among the entire matrix
    - dE should just be -2x the energy contribution from that spin
  + Need to integrate this reworking into the min\_energy tracking as well

Jij Matrix

* Restrict j for loop to take into account for the fact that the matrix is symmetric
* \*\*\* THE JIJ MATRIX NEEDS TO BE SYMMETRIC!!! \*\*\*
* Line 93 – make j = i+1 (same with 153)
* Assign one half of the matrix, above the diagonal, and then assign over the diagonal

Minimum Energy

* Not a conventional quantity to track in a MC simulation
* Useful in this case because it does demonstrate the freezing that we’re looking for
  + Indicates that population annealing would be effective at increasing efficiency
* Generally, one sweep doesn’t change the system very much
  + Most simulations have a # of sweeps that are performed before measurements are taken to account for the enormous number of sweeps being performed in a large scale simulations
  + “Bin” a few measurements together – something like averaging X measurements and storing that to disk
* If measurements are expensive (lots of quantities, for example) we don’t want to do that too often – might end up needing to add extra layers of loops to implement this

Strategy moving forward

* Ultimate goal will be to, within one call to main(), spawn several random seeds in parallel to periodically exchange configurations between CPUs – POPULATION ANNEALING
* First step – add a command line argument to be the number of seeds (# of initial configurations) and add that iteration as an inner loop
  + Perform a certain number of sweeps on each seed before comparing between them and exchanging
  + Try to build this structure into the code now as a sequential process to understand the algorithm

Freezing

* We see that certain trials over the same Jij matrix don’t equilibrate to the same energy
* We also don’t know if any of them are actually the minimum energy (NP-Complete problem)
* Flat line – the simulation is essentially frozen
* Ferromagnetic trials would equilibrate very quickly and to the same energy
  + TRY RUNNING THIS AND SEE WHAT THE PLOTS LOOK LIKE

To get an identical Markov chain, we would need to set both MC\_SEED and SPIN\_SEED to be identical

* Try changing the MC seeds to be equal to SPIN\_SEED in order to ensure that we don’t have the same set of random numbers determining how the MC loop chooses spins
* We do generally want MC\_SEED to be independent between trials
* Try having a vector of seeds through which to iterate
* Boost library has a very optimized RNG in it
  + Mersenne Twister is a good one
  + Probably not worth using for this code but is used in PIMC

Another thing worth implementing

* Simulated annealing – not population annealing – instead of having a fixed beta, we start with beta = 0 and gradually ramp up beta to the final value throughout the simulation
* Simplest case- every sweep, increment beta by beta\_final / num\_sweeps
* By doing this, we are less likely to get stuck at local minima initially
* This can be implemented easily by adding an “annealing” flag
* Can analyze by creating histograms of the minimum energy for each seed running simultaneously with annealing on and off to see the improvements granted