This homework is due on Nov 16 at 4pm.

I. CODING

In this homework we will write a new sub-routine for updating the spin configurations using the Wolff single cluster algorithm. This function should have the same prototype as the "sweep" function we used in the single spin flip algorithm:

```
void wolff_update(const PARAMS&,const LATTICE&,MTRand&);
```

As far as the main program is concerned, this function has the same purpose as the old function, *i.e.* it updates the Ising configuration satisfying detailed balance – thats why its prototype is exactly the same. You have to think by yourself what changes to the rest of the program including the class variables need to be made in order to code in this new update scheme. Complete writing the code for the new function "wolff_update". Also, include a new estimator for the average Wolff cluster size in your "data.out" file. You can debug your code by comparing the values for χ and $\langle e\rangle$ that you produced in your last homework with the single spin flip Metropolis, the full enumeration and also checking against Wolff's original paper (access from campus)
http://journals.aps.org.ezproxy.uky.edu/prl/abstract/10.1103/PhysRevLett.62.361
(80 points)

II. PERFORMANCE ANALYSIS

How much better is the cluster algorithm than the single spin flip algorithm? It seems to update the configurations much better, but how do we make this quantitative? A major problem with Monte-Carlo simulations is long auto-correlation times (in "MC time") next to criticality. We can extract the "auto-correlation time" Θ for a given observable O of our simulation by looking at the autocorrelation function,

$$A_O(\tau) = \frac{\langle O_t O_{t+\tau}\rangle - \langle O_t\rangle^2}{\langle O_t^2\rangle - \langle O_t\rangle^2} \tag{1}$$

where the averages are implied to be taken over the time-index t. In a perfect simulation, the function $A_O(\tau)$ should be 0 for all τ except at $\tau = 0$ where $A_O(0) = 1$ by definition. In practice however it decays continuously to zero. For sufficiently large τ, it should decay as,

$$A_O(\tau) \to e^{-\tau/\Theta} \tag{2}$$

The indefinite increase of Θ as one approaches the critical point $\beta = \ln(1+\sqrt{2})/2 \approx 0.440687\ldots$ and makes the size $L \to \infty$ is referred to as "critical slowing down".

In this section we will study the critical slowing down in both the single-site Metropolis and Wolff algorithms for the Ising model. To do so, write a python program that reads in a "data.out" file and then calculates the function above for $\tau = 0 \ldots \tau_{\text{max}}$. Use the observable m^2 to do the analysis.

(a) Work at the critcal temperature. Plot $A_O(\tau)$ for a few different L for both the single-site and Wolff algorithms. Show that you obtain a function form consistent with Eq. (2).

(b) To estimate Θ, use the integrated auto-corrlelation time $\Theta_{\text{int}} = \sum_{\tau=0}^{\tau_{\text{max}}} A_O(\tau)$. Sitting at the critical point one expects Θ to be limited only the system size, hence $\Theta \sim L^z$, where the exponent z tells you how bad the critical slowing down is. The larger the power the worse the algorithm suffers from critical slowing down. Studying data for both algorithms, how much do you estimate $z_{\text{ss-met}}$ and z_{Wolff} are? (Don't worry about doing an error analysis in this section) *(60 points)*