Physics 115/242
Monte Carlo simulation of the 2-dimensional Ising model.

Peter Young
(Dated: June 3, 2013)

Here is my source code for the calculation of $\langle m^2 \rangle$.

I `#define` a variable `LMAX` to equal 32, and dimension the spin array to be `LMAX` by `LMAX`. Hence I can simulate any size up to `LMAX` without having to recompile the code.

During execution I read in the value of `L` using the `scanf ("%d", &L);` command. I also read in the smallest and largest temperatures, and the interval between successive temperatures that will be used. I use only temperatures near T_c. For larger sizes the range is closer to T_c than for smaller sizes, but the spacing between them is smaller so I have a similar number of temperatures, about 15, for each size.

I use `sweeps_for_equil` $=$ `5L`^{2} sweeps for equilibration and `sweeps_for_meas` $=$ `30L`^{2} sweeps for measurement. I repeat these sweeps for `nrun` $=$ `50` statistically independent runs to get better statistics and to get the error bars.

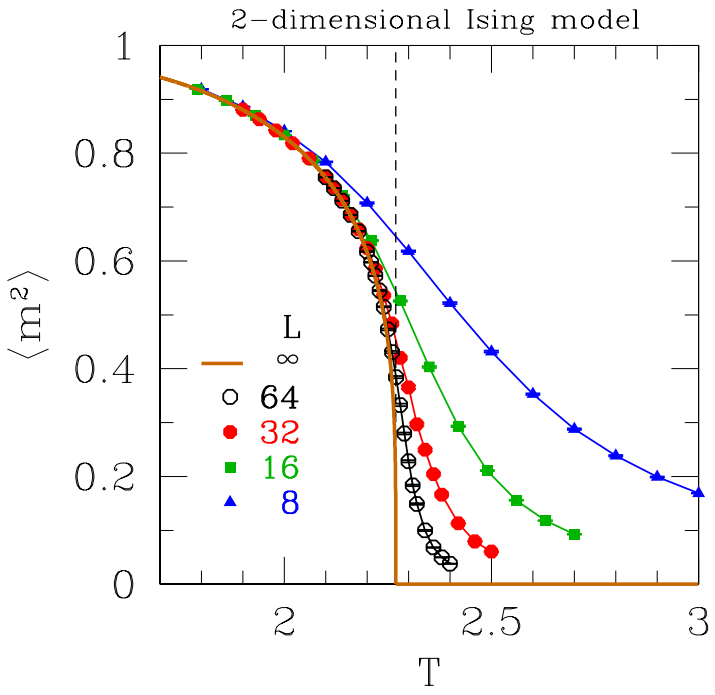
To speed up the simulation I precompute the possible values of the exponentials in the (heatbath) flip probability in an array `P`[17]. The energy to flip, `de`, can only be one of 5 values, $\pm 8, \pm 4, 0$ and I compute the flip probabilities for all integer values between -8 and 8. The line which flips the spins in the `update_1_sweep` is

```
if(random() < P[de+8]) s[x][y] = - s[x][y];
```

in which I shift `de` by 8 because in C, unlike in fortran, arrays have to start at zero.

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>

#define LMAX 32
/*****************************************************************************
*
* Does 1 MC sweep
*
*****************************************************************************/
void update_1_sweep(int s[LMAX][LMAX], int L, int P[])
{
    int x, xm, xp, y, ym, yp, de;

    xm = L - 2;
    x = L - 1;
    for (xp = 0; xp < L; xp++)
    {
        ym = L - 2;
        y = L - 1;
        for (yp = 0; yp < L; yp++)
        {
            de = 2 * s[x][y] * (s[xp][y] + s[xm][y] + s[x][yp] + s[x][ym]);
            if(random() < P[de+8]) s[x][y] = - s[x][y];
            ym = y;
            y = yp;
        }
        xm = x;
        x = xp;
    }
}

/*****************************************************************************
*
```

```
* Tabulates the exponentials
*
******************************************************************************/
void set_tabs(int P[], double beta)
{
    int i;
    for (i = -8; i <= 8;  i++)
        P[i+8] = (int) round( RAND_MAX /  (exp(i *beta) + 1) );
}

/*****************************************************************
 *
 * main program
 *
*****************************************************************/
main()
{
    int    s[LMAX][LMAX], run, nrun, P[17];
    int    iT, NT, L, N, sweep, sweeps_for_equil, sweeps_for_meas, x, y;
    double Tmin, Tmax, DT, T, beta, m, m2, m2_av, m2_err;
    void   update_1_sweep(int s[LMAX][LMAX], int L, int P[]);
    void   set_tabs(int P[], double beta);

    srandom(time(NULL));

    printf (" L = ? ");
    scanf ("%d", &L);
    N = L * L;
    sweeps_for_equil = 5 * N;    // Sweeps for equilibration
    sweeps_for_meas =  30 * N;   // Sweeps for measurement
    nrun = 50;                   // Number of independent runs

    printf (" Tmin, Tmax, DE = ? ");
    scanf ("%lf %lf %lf", &Tmin, &Tmax, &DT);
    printf (" \n");
    fflush(stdout);
    NT = (int) round((Tmax - Tmin) / DT) + 1;     // Number of temperatures

    for (iT = 0; iT < NT; iT++)  // Sum over temperatures
    {
        T = Tmax - DT * iT;
        beta = 1 / T;
        set_tabs(P, beta);

        for (y = 0; y < L; y++)   // Initialize spins up
        {
            for (x = 0; x < L; x++) s[x][y] = 1;
        }
        m2_av = 0; m2_err = 0;

        for (run = 0; run < nrun; run++)  // Sum over independent runs
        {
            for (sweep = 0; sweep < sweeps_for_equil; sweep++) //Sweeps for eq.
                update_1_sweep(s, L, P);

            m2 = 0;
            for (sweep = 0; sweep < sweeps_for_meas; sweep++) //Sweeps for mea.
            {
                update_1_sweep(s, L, P);
```



L=∞ curve is the exact result of Onsager(1949) and Yang(1952)

FIG. 1: Data for $\langle m^2 \rangle$. The results vary from close to 1 well below the transition temperature $T_c \simeq 2.269$ to 0 well above T_c. I also show the exact result for $L \to \infty$, which has a sharp transition at T_c with $\langle m^2 \rangle$ precisely zero at higher temperatures. For a finite system we see that this sharp transition is rounded out, with the range of T where the rounding occurs being larger for the smaller sizes.

```
                m = 0;
                for (y = 0; y < L; y++)      // Calculate the magnetization
                {
                    for (x = 0; x < L; x++) m += s[x][y];
                }
                m2 += m*m;      // m is the magnetization for one configuration
            }                   // End of loop over measurement sweeps
            m2 /= ((double) sweeps_for_meas * N * N);// <m^2> 1 run
            m2_av += m2;
            m2_err += m2*m2;
        }                       // End of loop over runs
        m2_av /= nrun;          // <m^2> averaged over runs
        m2_err /= nrun;
        m2_err = sqrt(fabs(m2_err - m2_av*m2_av)/(nrun - 1)); // error bar
        printf (" %9.5f %5d %9.5f %9.5f \n", T, L, m2_av, m2_err);
        fflush(stdout);
    }                           // End of loop over temperatures
}
```

Now for the results, In the plots the error bars are present, but are smaller than the size of the points so they are difficult to see.

I also show results for the "dimensionless" Binder ratio g, discussed in the question for 242 students. The point is that the data intersects at the transition temperature, which is therefore indicated directly.

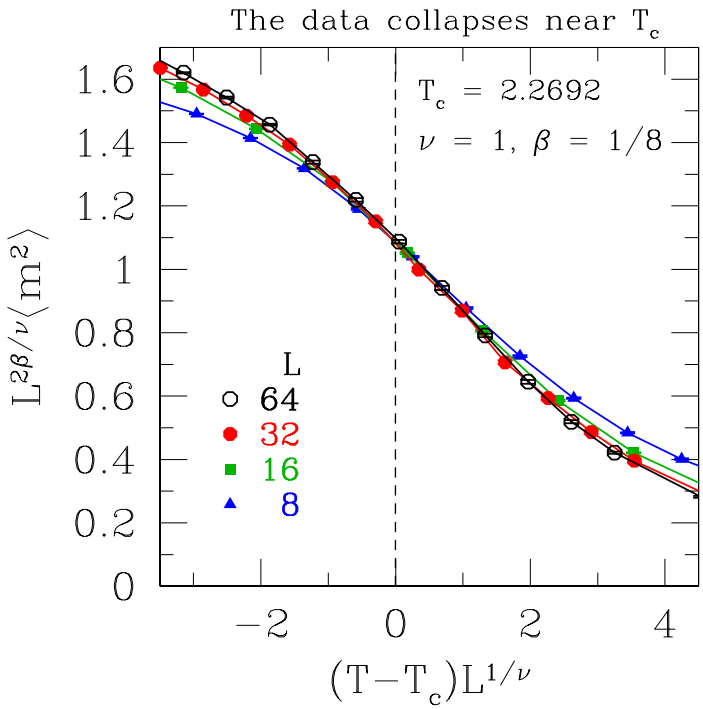


FIG. 2: A scaling plot of the data for $\langle m^2 \rangle$. The data is expected to collapse for $T - T_c$ small and L large. Here, we see that the $L = 8$ data deviates a bit from the other sizes, an indication that $L = 8$ is too small for the data collapse to work well. The collapse of the data works better for $L = 16$ and 32. I also show data for $L = 64$ (not required for the question), which collapses extremely well on to the $L = 32$ data.

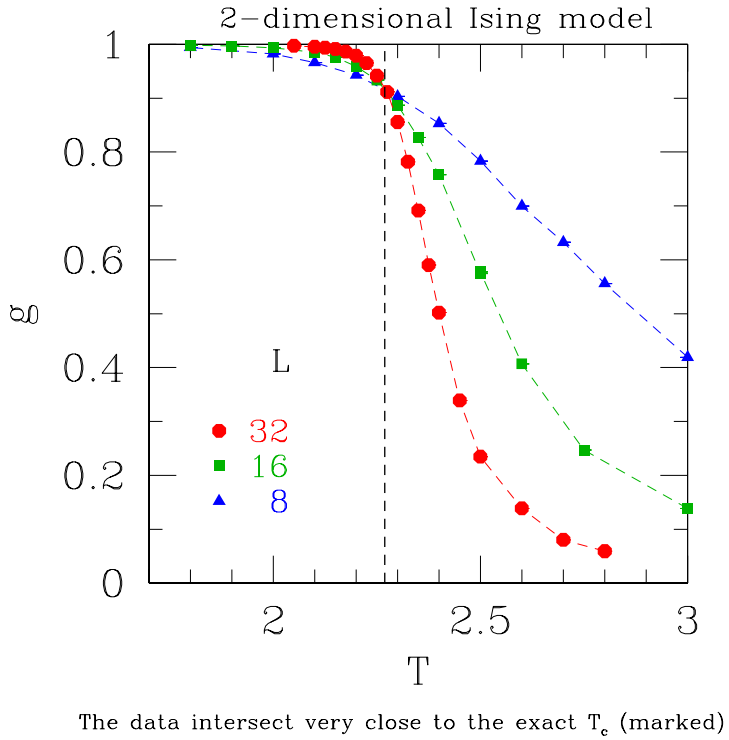


FIG. 3: Data for the "Binder ratio" g.

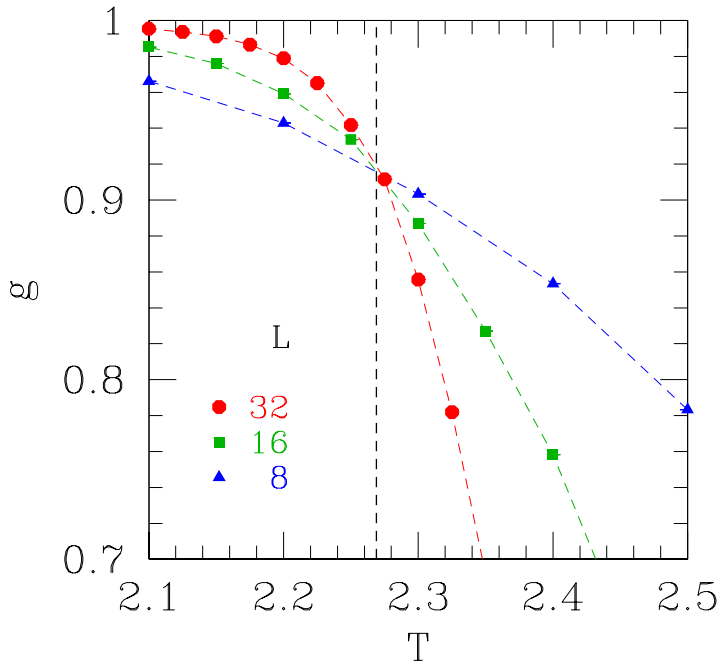


FIG. 4: Blow up of the Binder ratio data near T_c.

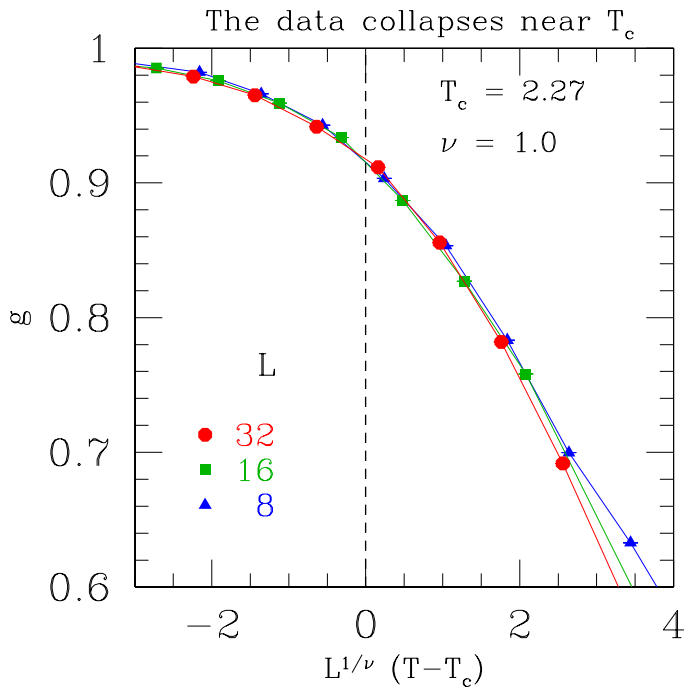


FIG. 5: Scaling plot of the data for g. The data collapses well with the correct exponent $\nu = 1$, but it does not collapse well if $\nu \lesssim 0.8$ or $\nu \gtrsim 1.2$