An example of a simulation: 2d Ising model

```
/*
 * Slightly adapted from a code (c) Kari Rummukainen.
 * Simple 2d Ising model simulation program in c.
 * Uses heat bath update.
 * Prints out measurements of energy and magnetization.
 *
 * Needs files: ising_sim.c mersenne.h mersenne_inline.c
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <sys/time.h>

/* Mersenne twister random number generator */
#include "mersenne.h"

/* Lattice size, adjust */
#define NX 32
#define NY 32

/* How many iterations before each measurement */
#define N_MEASURE 10

/* spin storage and coupling constant*/
static int s[NX][NY];
static double beta;
```

The variable `s[NX][NY]` corresponds to $\sigma_{\mathbf{x}} = \pm 1$.

```
/* 2 options to search for the neighbouring locations:
 * i - current location, xup(i) - location up to x-dir.
 * xup(i) is defined either
 * a) as an index array (which is filled at the beginning) or
 * b) as a modulo ((i+1) % NX)
 * option a) is OK for all lattice sizes; option b) is fast
 * only for lattice sizes which are powers of 2!  Modulo
 * to other powers is very slow.
 * a) : define USE_ARRAY
 */
```

```
#define USE_ARRAY

#ifdef USE_ARRAY
static int x_up[NX],y_up[NY],x_dn[NX],y_dn[NY];
#define xup(i) x_up[i]
#define xdn(i) x_dn[i]
#define yup(i) y_up[i]
#define ydn(i) y_dn[i]
```

The above is what we use: the arrays will be defined presently.

```
#else
#define xup(i) ((i+1) % NX)
#define xdn(i) ((i-1+NX) % NX)
#define yup(i) ((i+1) % NY)
#define ydn(i) ((i-1+NY) % NY)
#endif

char usage[] = "Usage: ising_sim N_LOOPS\n";

void update(void);
void measure(int iter);
```

The main subroutines.

```
int main(int argc, char* argv[])
{
  struct timeval tv;
  struct timezone tz;
  int i,x,y,n_loops;

  /* require correct number of input arguments
   */
  if (argc != 3) {
    fprintf(stderr,usage);
    exit(0);
  }
```

```
  n_loops = atoi(argv[1]);    /* number of loops */
  beta = atof(argv[2]);       /* value of beta */

  gettimeofday( &tv, &tz );

  /* seed the random numbers, using current time */
  /* seed_mersenne( tv.tv_sec + tv.tv_usec ); */
  /* seed the random numbers, in a reproducible way: */
  seed_mersenne( 2003 );

  /* critical beta
  beta = log(1+sqrt(2.0))/2.0 = 0.4406868; */

#ifdef USE_ARRAY
  /* fill up the index array */
  for (i=0; i<NX; i++) {
    x_up[i] = (i+1) % NX;
    x_dn[i] = (i-1+NX) % NX;
  }

  for (i=0; i<NY; i++) {
    y_up[i] = (i+1) % NY;
    y_dn[i] = (i-1+NY) % NY;
  }
#endif

  /* Initial spins - completely ordered, "cold" */
  for (x=0; x<NX; x++) for (y=0; y<NY; y++) s[x][y] = 1;
  /* Initial spins - completely disordered, "hot"
    for (x=0; x<NX; x++) for (y=0; y<NY; y++) {
     if (mersenne() < 0.5) s[x][y] = 1; else s[x][y] = -1;
    }                                              */
```

The main loop:

```
  for (i=0; i<n_loops; i++) {
    update();
    if (i % N_MEASURE == 0) measure(i);
  }
}
```

```
void update()
{
  int sum,x,y;
  double p_plus,p_minus;


  /* sum over the neighbour sites - typewriter fashion */
  for (x=0; x<NX; x++) for (y=0; y<NY; y++) {
    sum = s[xup(x)][y] + s[xdn(x)][y]
        + s[x][yup(y)] + s[x][ydn(y)];

    /* Heat bath update - calculate probability of +-1 */
    p_plus  = exp( (beta) * sum );   /* prob. of +1, unnormalized */
    p_minus = 1.0/p_plus;                  /* and -1 */
    p_plus  = p_plus/(p_plus + p_minus);  /* normalized prob of +1 */

    /* and choose the state appropriately */
    if (mersenne() < p_plus) s[x][y] = 1; else s[x][y] = -1;
  }
}
```

Recall:

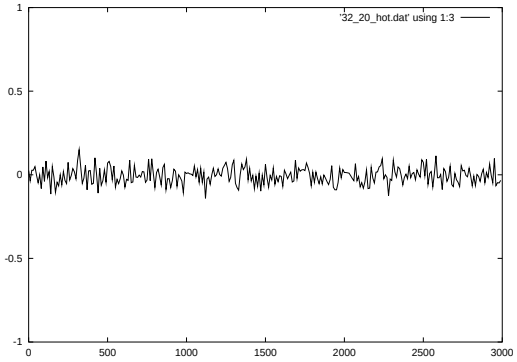
$$S = \ldots - \beta\sigma_{\mathbf{z}}\left(\sigma_{\mathbf{z}+\hat{\mathbf{1}}} + \sigma_{\mathbf{z}+\hat{\mathbf{2}}} + \sigma_{\mathbf{z}-\hat{\mathbf{1}}} + \sigma_{\mathbf{z}-\hat{\mathbf{2}}}\right) \equiv \ldots + S_{\mathbf{z}}(\sigma_{\mathbf{z}}) \ ,$$

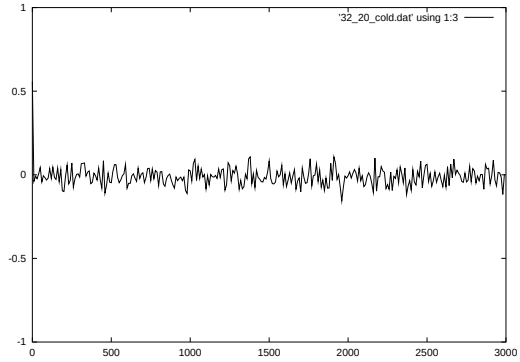
$$p(\sigma'_{\mathbf{z}}) = \frac{\exp\left[-S_{\mathbf{z}}(\sigma'_{\mathbf{z}})\right]}{\exp\left[-S_{\mathbf{z}}(+1)\right] + \exp\left[-S_{\mathbf{z}}(-1)\right]} \ .$$

```
void measure(int iter)
{
  /* measure energy and magnetization, and print out
   */
  int sum,x,y;
  double e,m;

  /* sum over the neighbour sites - typewriter fashion */
  e = m = 0.0;
  for (x=0; x<NX; x++) for (y=0; y<NY; y++) {
    e += - s[x][y] * (s[xup(x)][y] + s[x][yup(y)]);
    m += s[x][y];
  }

  printf("%d %f %f\n", iter, e/NX/NY, m/NX/NY);
}

/*
 *
 * To compile:
 *   cc -O2 -o ising_sim ising_sim.c mersenne_inline.c -lm
 *
 * To use:
 *   ising_sim #_loops beta > result_file
 * (e.g. #_loops = 3000, beta = 0.44)
 *
 */
```

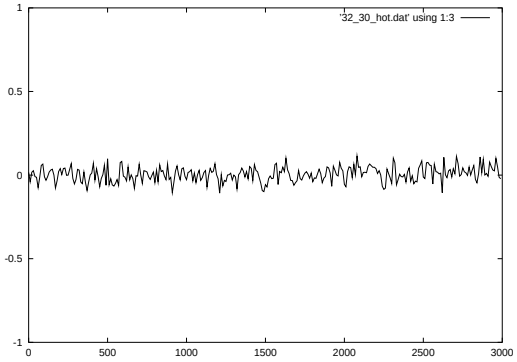
NX = NY = 32, n_loops = 3000, N_MEASURE = 10, hot and cold start.

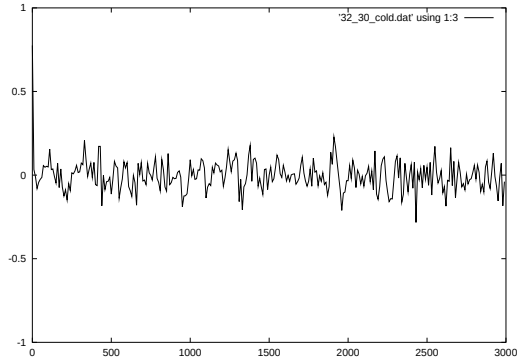
$\beta = 0.20$



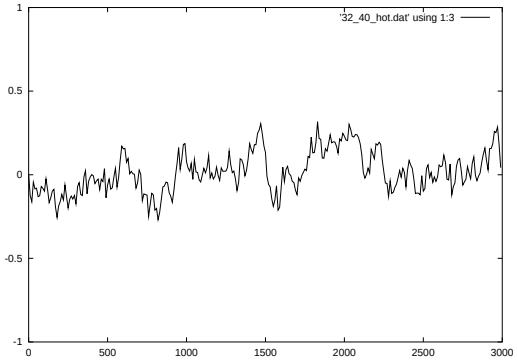


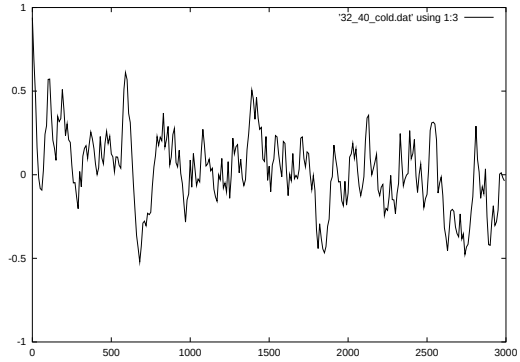
$\beta = 0.30$



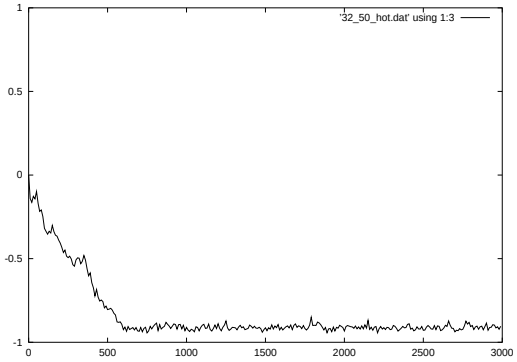


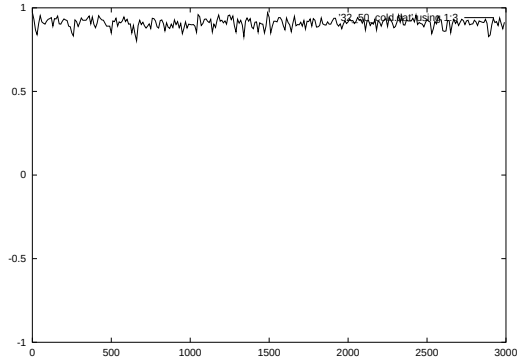
$\beta = 0.40$



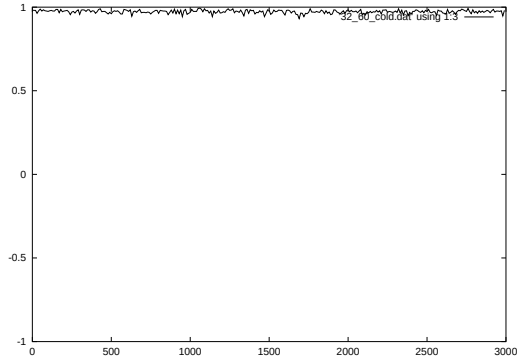


$\beta = 0.50$



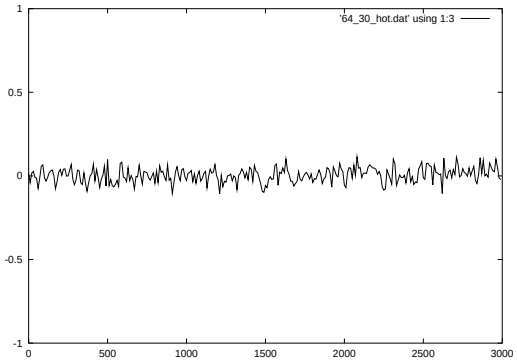


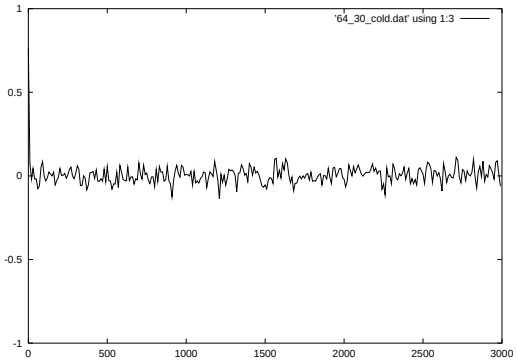
$\beta = 0.60$



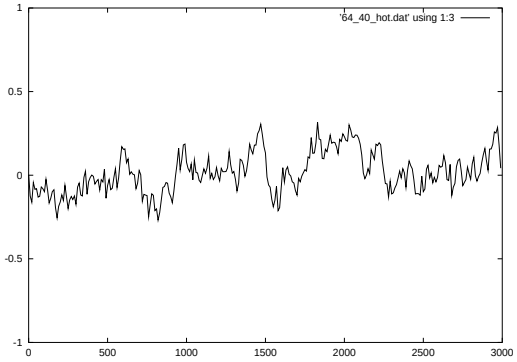
NX = NY = 64, n_loops = 3000, N_MEASURE = 10, hot and cold start.

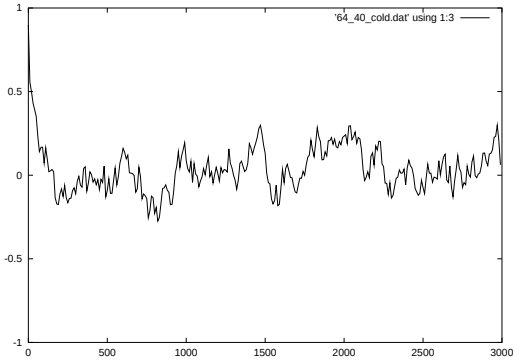
$\beta = 0.30$



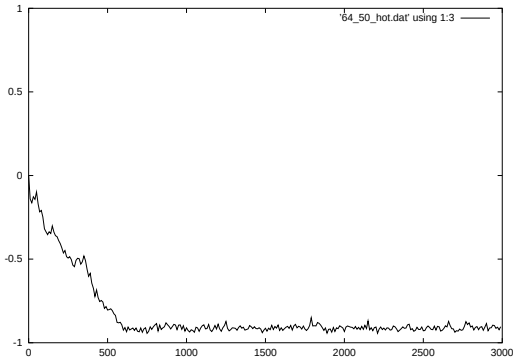


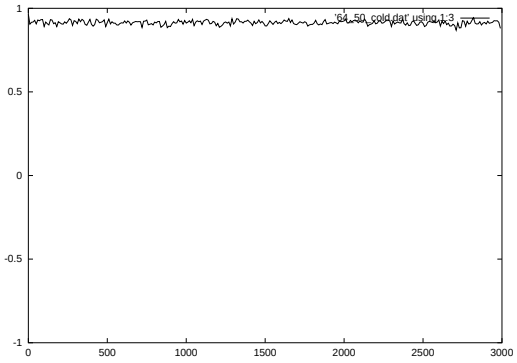
$\beta = 0.40$





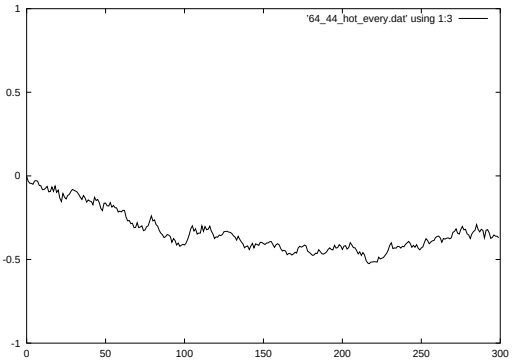
$\beta = 0.50$

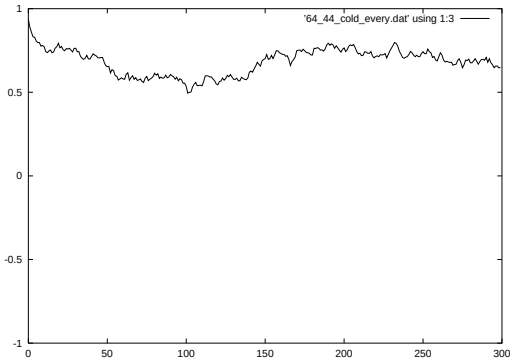




NX = NY = 64, n_loops = 300, N_MEASURE = 1, hot and cold start.

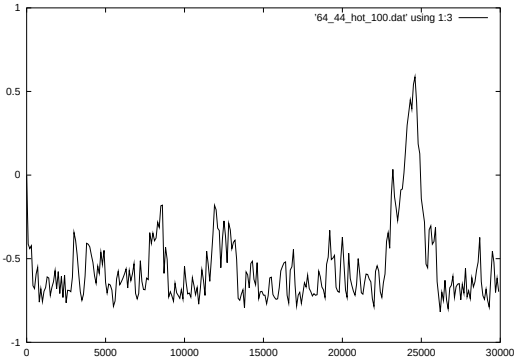
$\beta = 0.44$

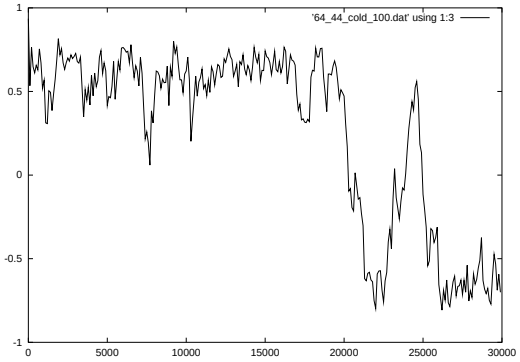




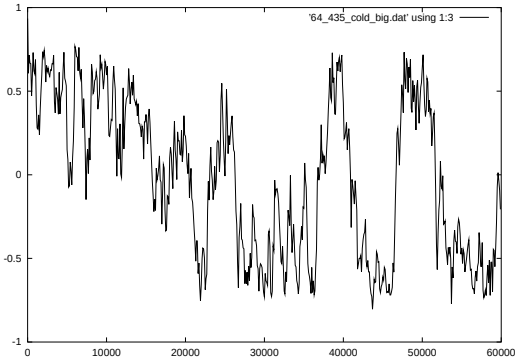
NX = NY = 64, n_loops = 30000, N_MEASURE = 100, hot and cold start.

$\beta = 0.44$

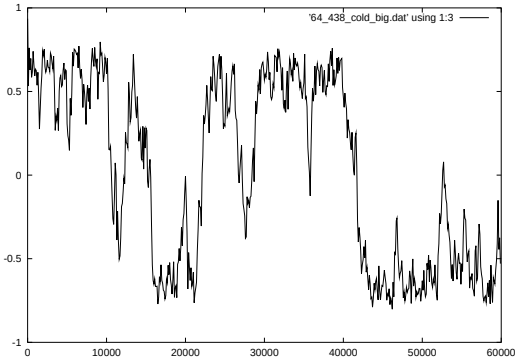




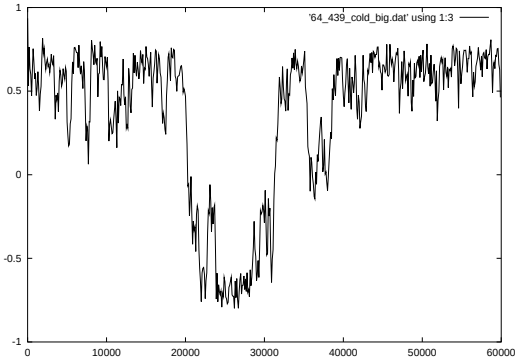
NX = NY = 64, n_loops = 60000, N_MEASURE = 100, cold start; $\beta_c = 0.4406868$ at inf.vol.



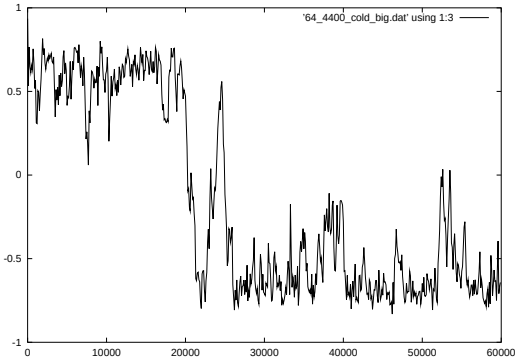
$\beta = 0.435$



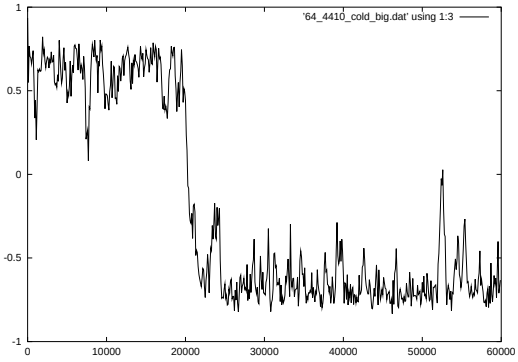
$\beta = 0.438$



$\beta = 0.439$



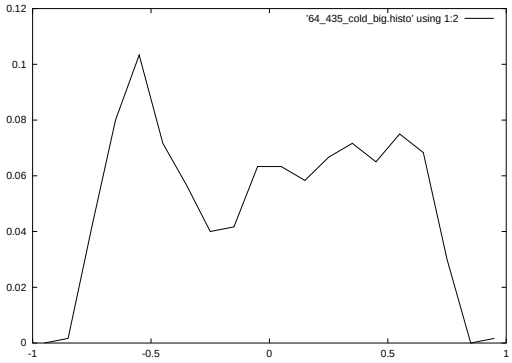
$\beta = 0.440$



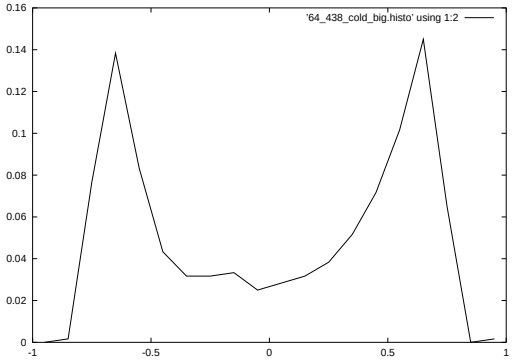
$\beta = 0.441$

The corresponding histograms:

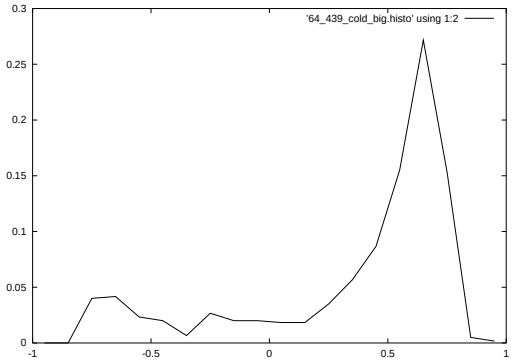
$\beta = 0.435$



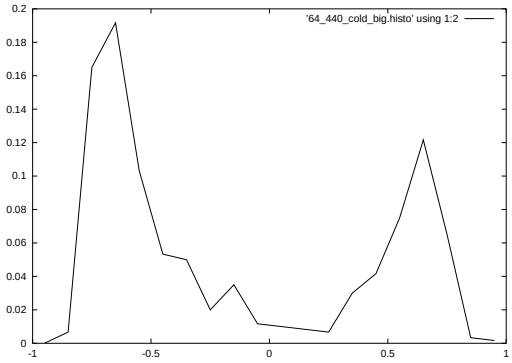
$\beta = 0.438$



$\beta = 0.439$



$\beta = 0.440$



$\beta = 0.441$