Quantum Monte Carlo Methods at Work for Novel Phases of Matter

SSE TUTORIAL I

SSE for Heisenberg model

Ying Tang
Anders Sandvik

Trieste 30/01/2012

BOSTON UNIVERSITY

TASKS TO DO (DAY I)

- Tour inside ssebasic.f90
- Test convergence of the program
- Test correction of the program
- Finite size scaling of sublattice magenetization
- Write your own measurement of spin-spin correlation (z component only)

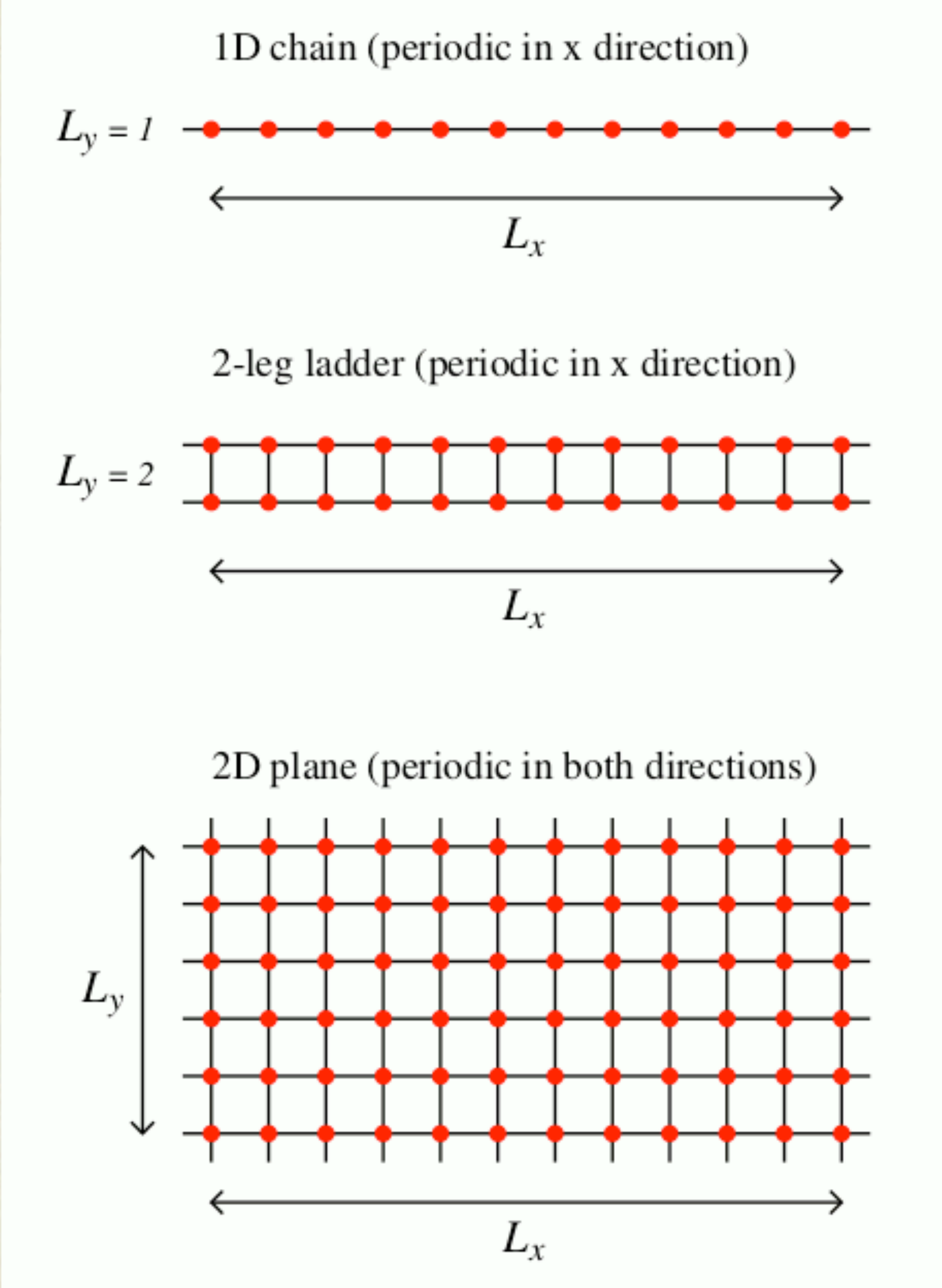
TOUR INSIDE SSEBASIC.F90

- The program can be found at:

http://physics.bu.edu/~sandvik/trieste12/

- Heisenberg Model

$$H = \sum_{<i,j>} J\mathbf{S_i} \cdot \mathbf{S_j}$$



GET STARTED

- Download it to a SSE directory (e.g. ~/SSE)
- Generate input file **read.in** and random number seed file **seed.in** in the same directory.
- Compile program with **g95/gfortran**
    - **gfortran -O ssebasic.f90**
- run **./a.out**

lx and ly should be even or ly=1

read.in

```
lx, ly, beta
nbins, msteps, isteps
```

seed.in

```
an integer
```

```
lx,ly  : lattice size in x and y direction
beta   : inverse dimensionless temperature J/T
nbins  : Number of bins (averages written to file 'res.dat' after each bin
msteps : Number of MC sweeps in each bin (measurements after each sweep)
isteps : Number of MC sweeps for equilibration (no measurements)
```

AN EXAMPLE

read.in

```
8  8  2
10 10000 10000
```

(takes a few seconds on macbook air)

10 bins

res.dat

energy	specific heat	sublattice $\langle m^2 \rangle$	susceptibility
-0.60545312499999993	0.45663118750002241	0.12939503187647464	7.4906249999999937E-002
-0.60682343749999990	0.44244269437507455	0.12865359015698782	7.4896875000000016E-002
-0.60736640624999993	0.42738905609377298	0.12857748019236501	7.6584374999999962E-002
-0.60639140624999999	0.47596175609373859	0.13036687638192762	7.6315624999999980E-002
-0.60511015625000009	0.39663114359370866	0.12531685491597871	7.7206250000000040E-002
-0.60642031250000006	0.42985882437494638	0.13028933087043010	7.7446874999999984E-002
-0.60657500000000009	0.52802733499993337	0.13139164458652658	7.6490625000000065E-002
-0.60733203124999990	0.42871590234381074	0.12873853782797756	7.7559374999999998E-002
-0.60921562500000004	0.53947162249994562	0.13185469109301354	7.4915624999999996E-002
-0.60873593750000010	0.53154232437492510	0.12912797510686908	7.7259375000000051E-002

AN EXAMPLE

res.dat

energy <e>	specific heat <c>	sublattice <m^2>	susceptibility χ
-0.60545312499999993	0.45663118750002241	0.12939503187647464	7.49062499999999937E-002
-0.60682343749999990	0.44244269437507455	0.12865359015698782	7.48968750000000016E-002
-0.60736640624999993	0.42738905609377298	0.12857748019236501	7.65843749999999962E-002
-0.60639140624999999	0.47596175609373859	0.13036687638192762	7.63156249999999980E-002
-0.60511015625000009	0.39663114359370866	0.12531685491597871	7.72062500000000040E-002
-0.60642031250000006	0.42985882437494638	0.13028933087043010	7.74468749999999984E-002
-0.60657500000000009	0.52802733499993337	0.13139164458652658	7.64906250000000065E-002
-0.60733203124999990	0.42871590234381074	0.12873853782797756	7.75593749999999998E-002
-0.60921562500000004	0.53947162249994562	0.13185469109301354	7.49156249999999996E-002
-0.60873593750000010	0.53154232437492510	0.12912797510686908	7.72593750000000051E-002

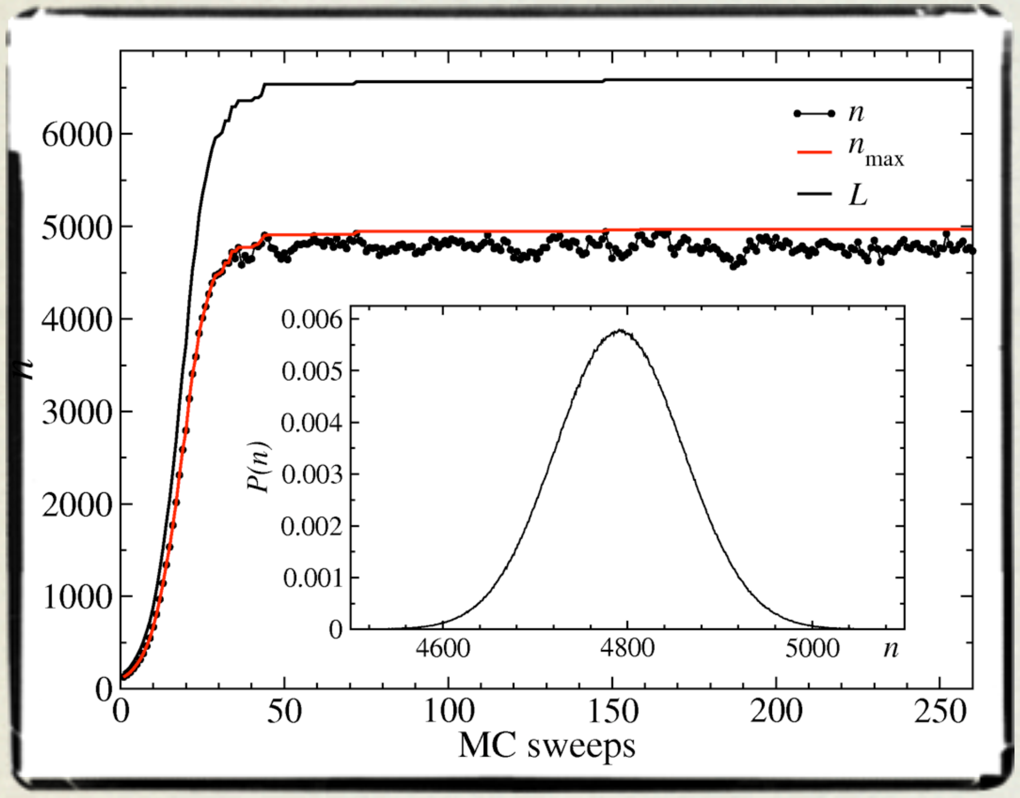
- Calculate error bars
  - compile res.f90 **gfortran -o b.out -O res.f90**
  - run **./b.out**

	<e>	e error	<c>	c error
e.dat	-0.60694234	0.00040847	0.46566718	0.01608355

	<m^2>	m^2 error	<χ>	χ error
m.dat	0.12937120	0.00057857	0.07635813	0.00034245

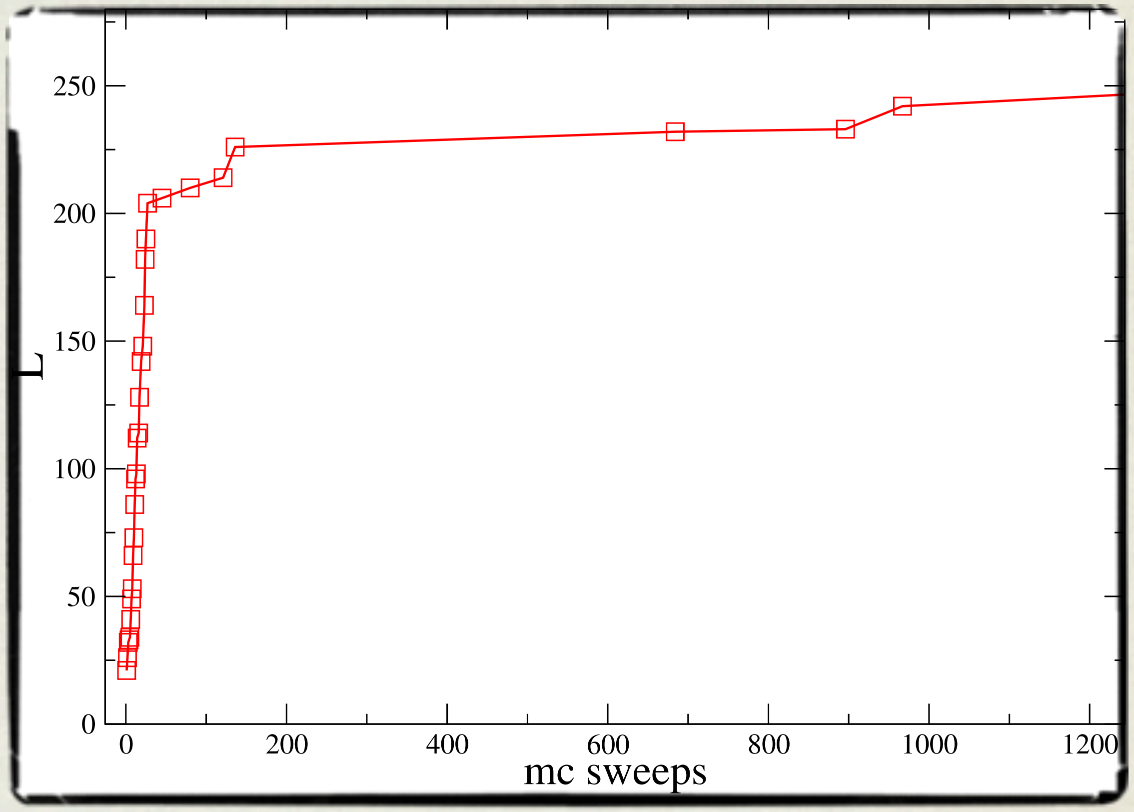
TEST OF CONVERGENCE

From Lecture



From previous example

cut.dat two columns: mc sweep, L



In this program, we only record L values once there is an increment.

TEST OF CORRECTNESS

CHECK THE GROUND STATE ENERGY OF HEISENBERG CHAIN

Lx=32, Ly=1 20 bins and 10,000 MC steps, use large β

beta	<e>	error
1.0	-0.20477922	0.00016893
2.0	-0.34135748	0.00016988
4.0	-0.41897368	0.00007303
8.0	-0.43776850	0.00003716
16.0	-0.44240240	0.00002830
32.0	-0.44377160	0.00002360
64.0	-0.44394747	0.00002178

exact e: -0.44395398

about 10 mins on macbook air

FINITE SIZE SCALING OF M_S^2

$$< m_s >= \frac{1}{N}\sum_{i=1}^{N}\phi_i \mathbf{S_i}$$

$$< m_s^2 >= \frac{1}{N^2}\sum_{i,j=1}^{N}\phi_i\phi_j \mathbf{S_i}\mathbf{S_j} = \frac{1}{N}\int C(\mathbf{r})d\mathbf{r}$$

$$C(r) = < \mathbf{S_i}\cdot\mathbf{S_{i+r}} >$$

TASK TO DO

- run 1D, 2-leg ladder and 2D program for several sizes L_x
- Plot $\mathbf{L_x <m_s^2>}$ vs $\mathbf{L_x}$ for 1D and Ladder
- Plot $\mathbf{<m_s^2>}$ vs $\mathbf{1/L_x}$ for 2D

Suggested Parameters (takes a few minutes)
*1D: β=2**L**, L=8, 16, 32, 64*
2-Leg Ladder: β=64, L=8, 16, 32
*2D (N=L²): β=2**L**, L=4 to 16*
bin=10 mc step=10,000

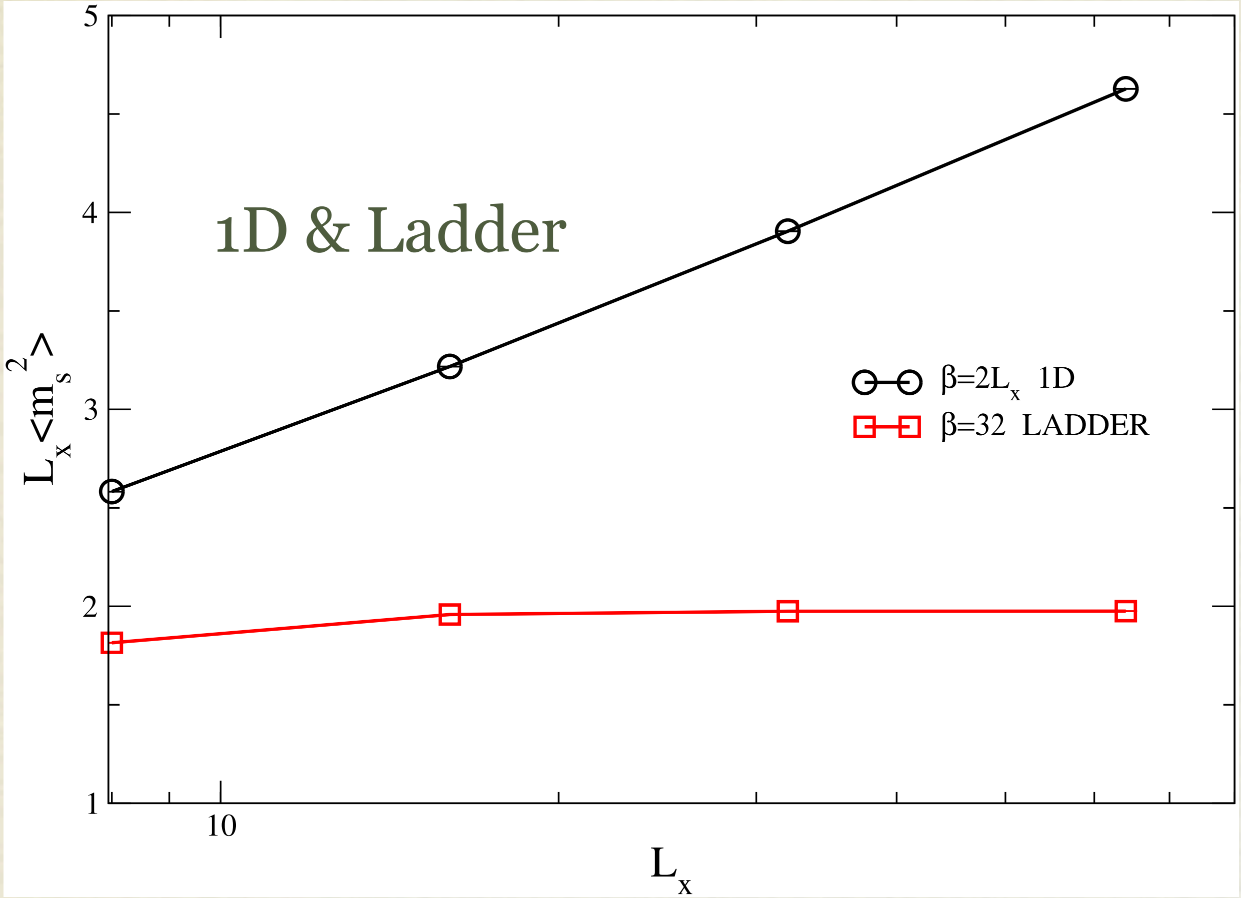
Do more data points if time allows!

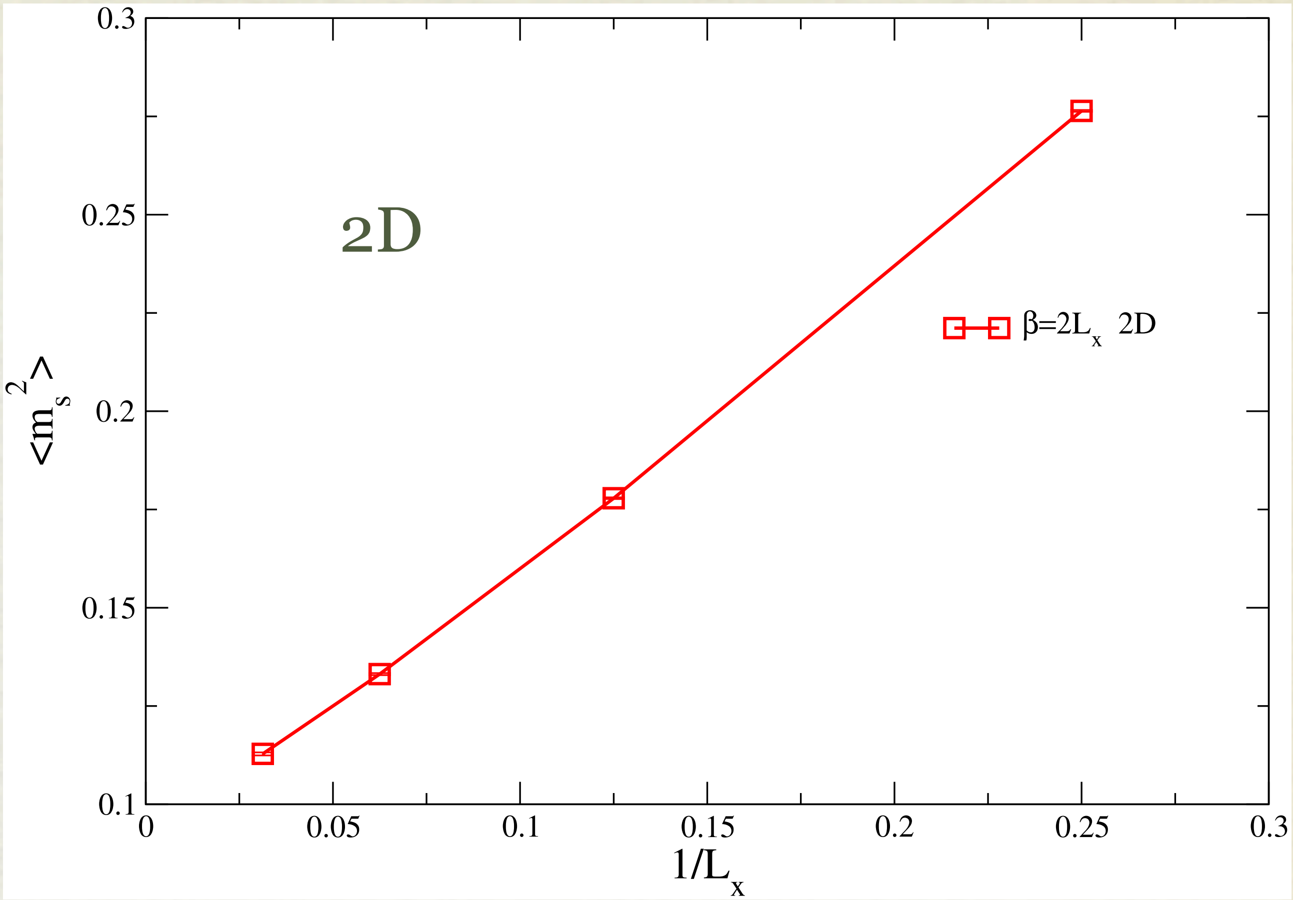
Questions to think about:

Why do we multiply 1D and Ladder $<m_s^2>$ with L_x while plot 2D $<m_s^2>$ versus inverse L_x? (key: C(r))

Why do we increase β as a function of system size in 1D and 2D Heisenberg Model? Why do you keep β as constant in 2-leg ladder? (key: gap)

FINITE SIZE SCALING OF M_S^2





Can be explained by:

$$< m_s^2 >= \frac{1}{N} \int C(\mathbf{r}) d\mathbf{r}$$

MEASUREMENT OF $C^Z(R)$

$$C^z(r) = < S_i^z \cdot S_{i+r}^z >$$

TASK TO DO

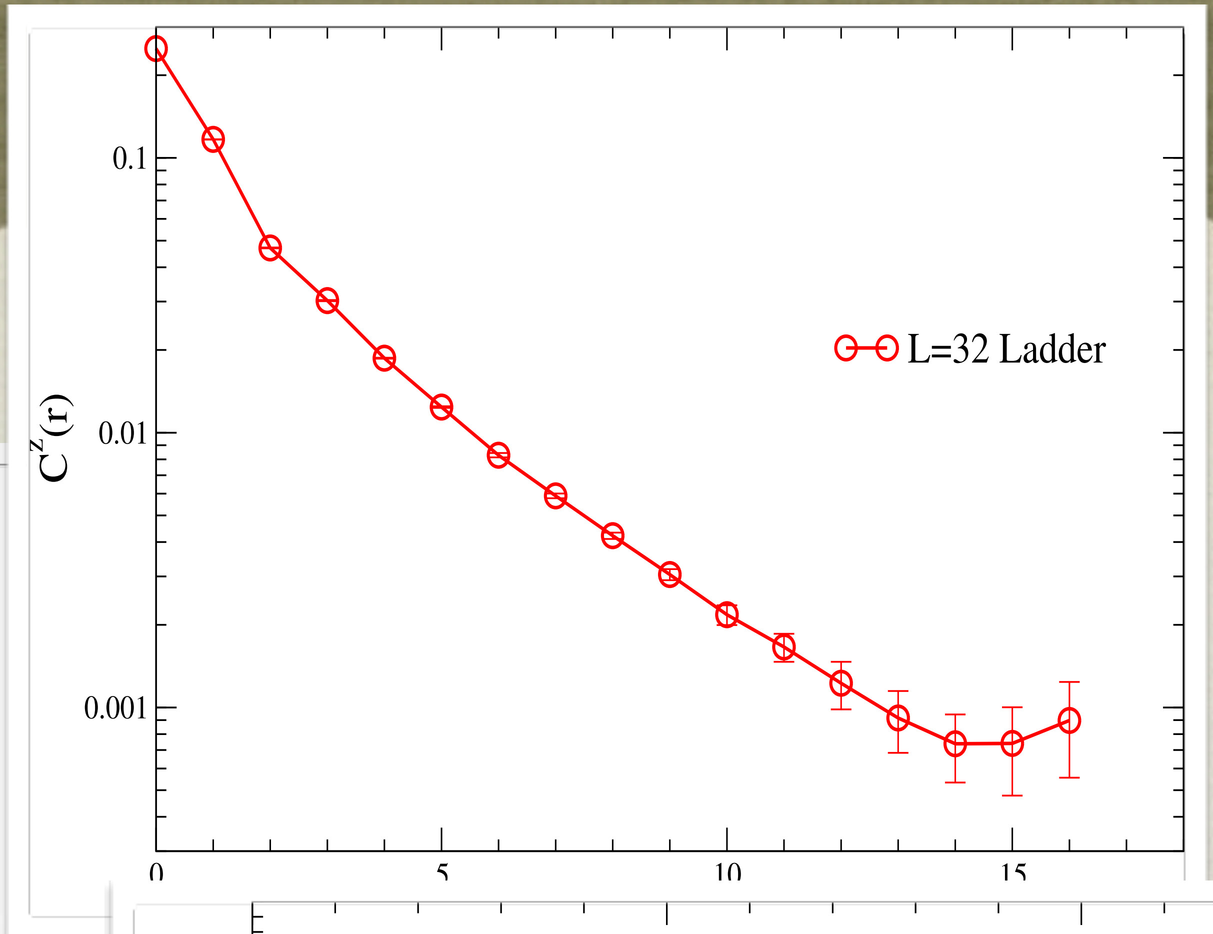
- Add a simple subroutine to measure z component of spin-spin correlation $C^z(r)$ for **1D, 2-leg Ladder and 2D** systems (only along y=1 for 2-leg ladder and 2D systems)
- Modify res.f90 to calculate error bars for $C^z(r)$.
- Plot $C^z(r)$ versus L_x. What forms do you get? Are they expected?

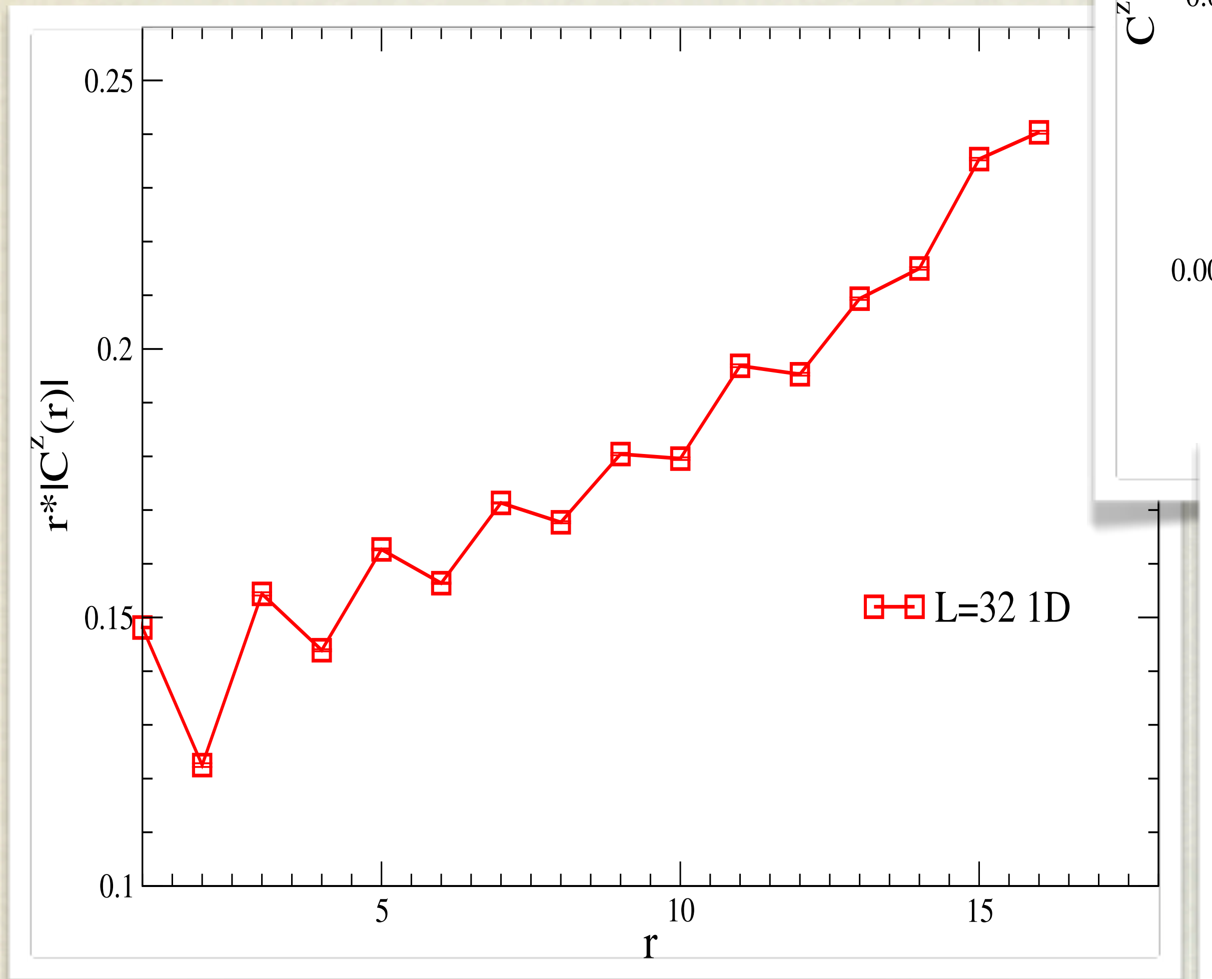
1st step ->
create an array in **measurementdata** *module:*

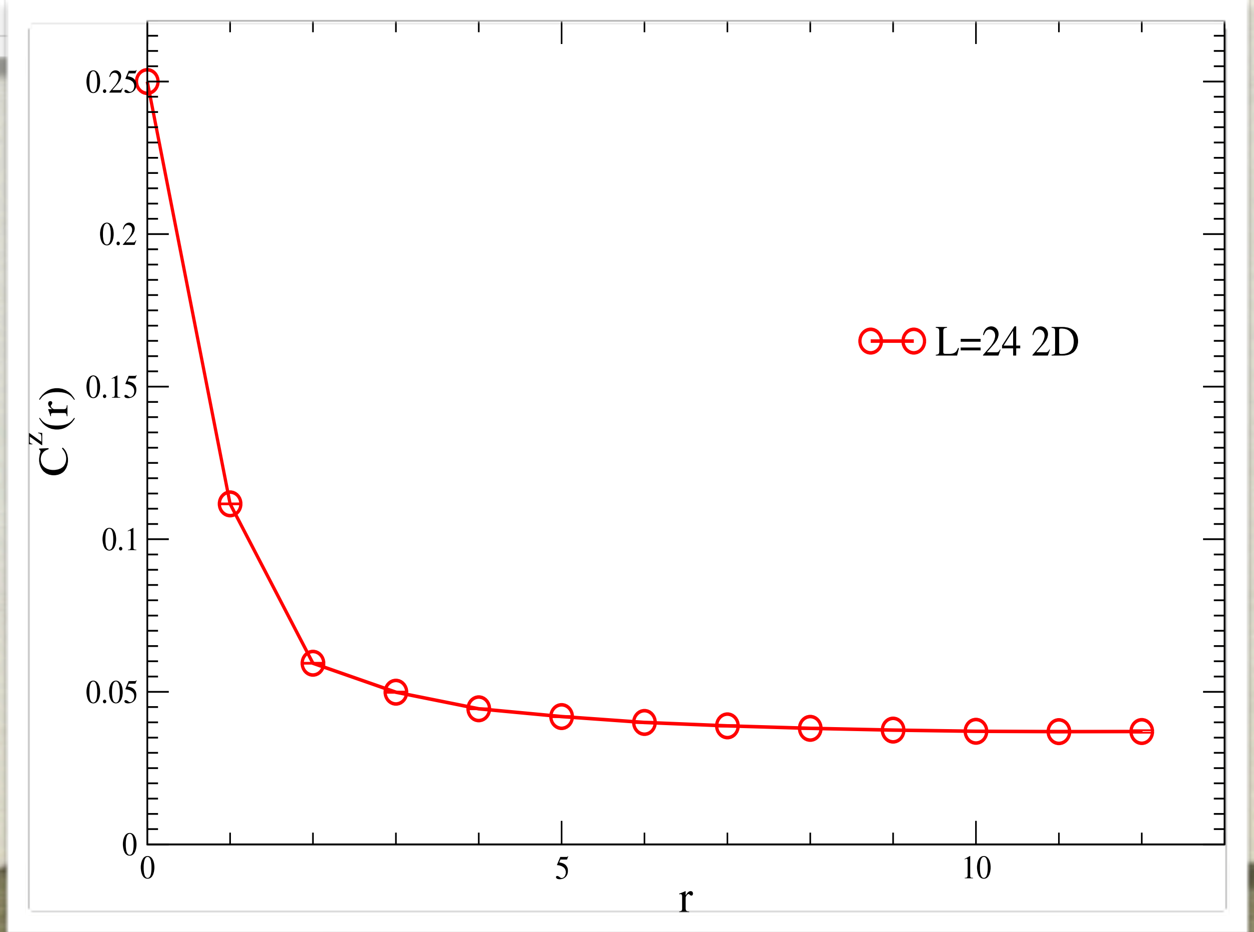
real(8), allocatable :: crr(:)

Do the measurement in the subroutine **measure**

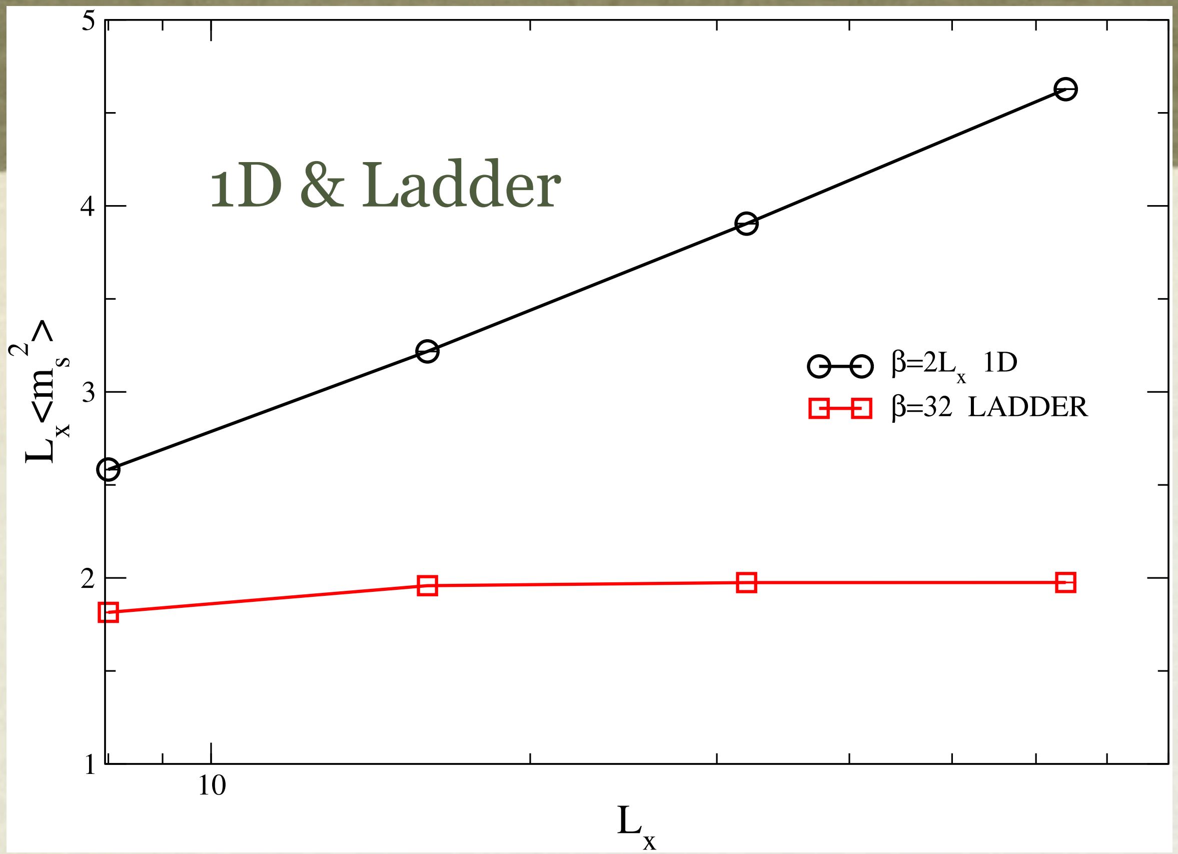
MEASUREMENT OF $C^Z(R)$

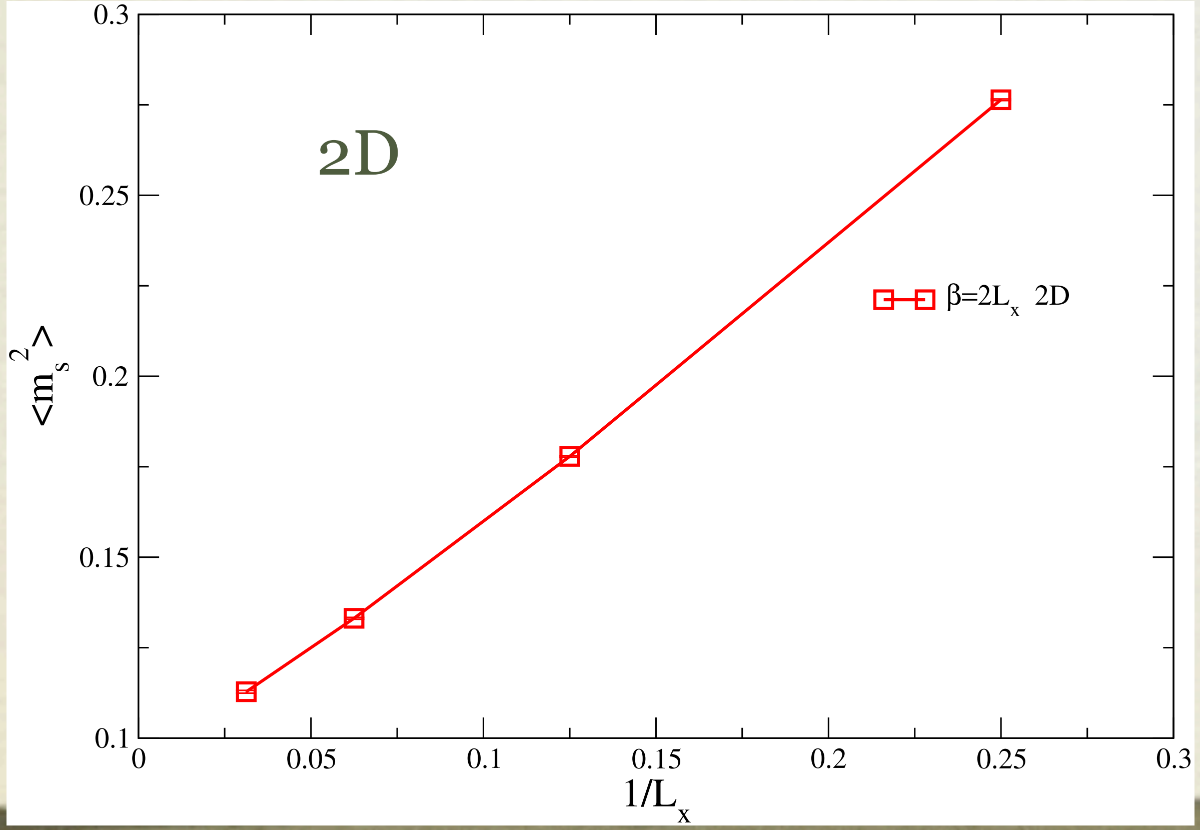






FINITE SIZE SCALING OF M_S^2





$$< m_s^2 >= \frac{1}{N} \int C(\mathbf{r}) d\mathbf{r}$$

r>>1

1D $\quad C(\mathbf{r}) \propto \frac{1}{r}$

Ladder $\quad C(\mathbf{r}) \propto e^{-\frac{r}{\xi}}$

2D $\quad$ constant

MAIN STEPS IN $C^Z(R)$ MEASUREMENT

```
real(8), allocatable :: crr(:)
allocate(crr(0:nn/2)) !allocate spin-spin correlator
```

```
do s1=1, nn  ! nn: spin number
      x=mod(s1-1,lx)     !x index
      y=int((s1-1)/lx)    !y index
      do r=0, lx/2
         s2=mod(x+r,lx)+y*lx+1
         crr(r)=crr(r)+spin(s1)*spin(s2)
      end do
end do
```

```
crr=0.25d0*crr/dble(nn)/dble(msteps)

deallocate(crr)
```