# Ising Model Simulation

Xin He, 1500011805

November 27, 2018

CCME, Peking University

---

## 1 Source

the program contains gibbs sampling Metropolis, Swendsen-Wang, Wolff algorithms. (but the KMC is still need added.)
For different problems, here we may use different algorithms.
the source code can be found at the author's mainpage of github, for IsingModel.
And all result data can also be found in the progect on github.

### 1.1 the procedure implement

Here gives the `Makefile` by author, make sure `gfortran` is available. Then you can compile the program by `make` to give a executable procedure `ising.run`.

you can run the program in two ways:

- run without any arguments
  `./ising.run`
  the program will read $\beta, J, h$ from default parameter file — `is.parms`.

- run with only 3 arguments
  `./ising.run input_beta input_J input_h`
  so the relative arguments will be read from command line.

the ising simulation program has 4 parts.

- ising_utils.f90:
  gives some useful subroutines for generating rand number, contains **init_seed, rand_int**
- ising_parms.f90:
  read parameters from the file (default **is.parms**), and sharing the public paramters.
  contains **Nsize, Nstep, parmb, parmJ, parmh, samp_type, work_type**

the simulattion parameter

| name | mean |
|---|---|
| Nsize | the size of 2-D grid of ising model |
| Nstep | the total steps for simulation |
| parmb | beta ($\beta = 1/k_B T$) |
| parmJ | coupling coefficient |
| parmh | the external magnetic field |

the sampling setting

| samp_type | method |
|---|---|
| 0 | Metropolis with gibbs sampling |
| 1 | Swendsen-Wang sampling |
| 2 | Wolff sampling |

the work type setting (suit for specific problem)

| work_type | method |
|---|---|
| 1 | calc both c and m, is for problem 1 |
| 2 | calc only m, is for problem 2 |
| 3 | calc only correlation, is for problem 3 |
| 4 | calc all, not support? |

- ising_model.f90:
  define **ising** type
  support Metropolis with gibbs sampling
  support Swendsen-Wang sampling
  support Wolff sampling

- the control scripts/anaysis scripts
  here use shell scripts to give a series simulation, ref `run.sh.example`
  here use python3 scripts to analysis the result

# 2 Study relation of of u~T, and c~T

## 2.1 Basic method (Gribbs)

$\beta$ is from a array $(0.01, 0, 02, \cdots, 1.00)$. For exact critical temperature of 2-D Ising model, the value is

$$k_B T_c / J = \frac{2}{\ln(1+\sqrt{2})} \approx 2.26918$$

which you can refer it from Wiki: square Ising model.

- For N=10 grid, need $10^7$ steps to converge;

- For N=50 grid, need $5 \times 10^7$ steps to converge;
- For N=100 grid, need $> 10^8$ steps to converge (for time reason, here we still use $5 \times 10^7$ steps);

Following, we use gibbs sampling, Swendsen-Wang sampling, Wolff sampling study the relation between

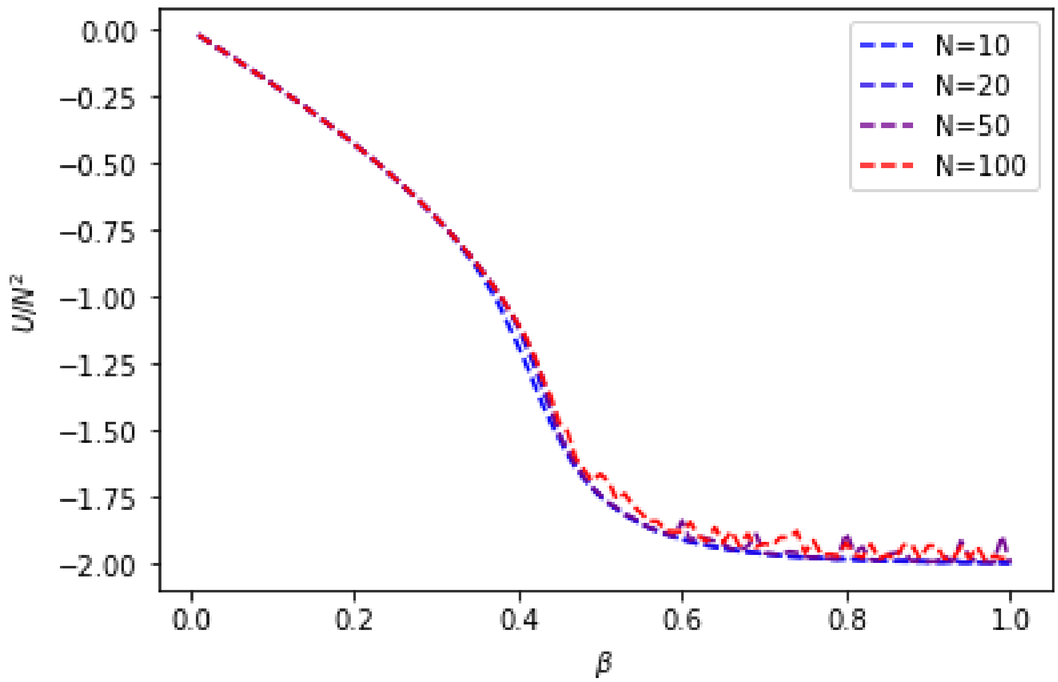
$$u \sim \beta$$

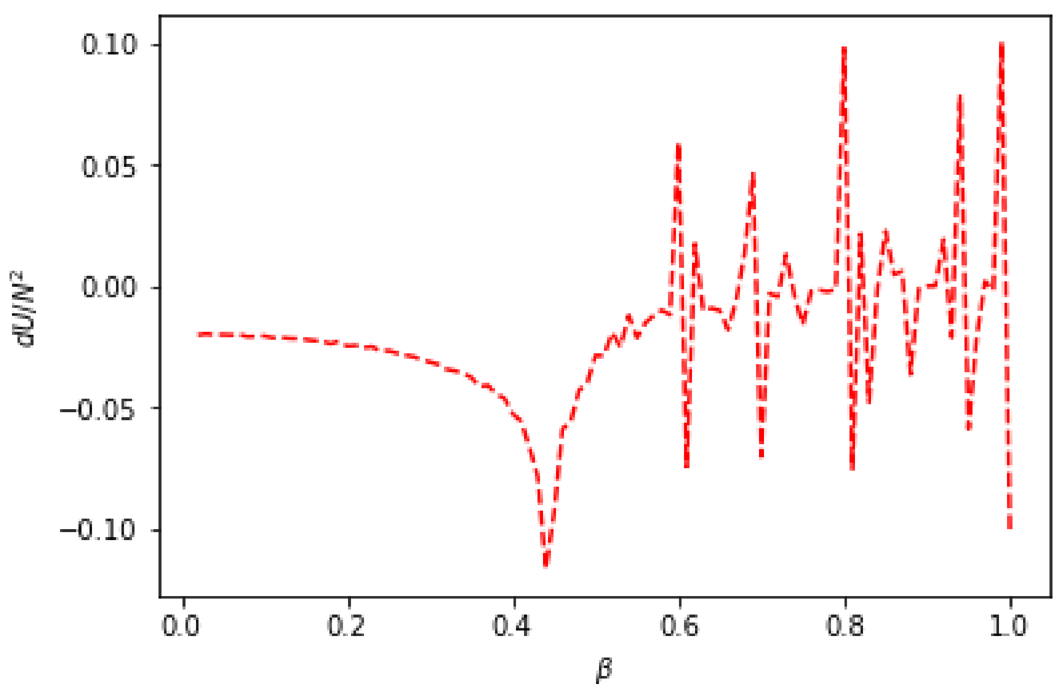
and

$$c \sim \beta$$

respectively, the result show that $N = 50$ is sufficient for energy to be converged, and suit for convergence in low temperature.

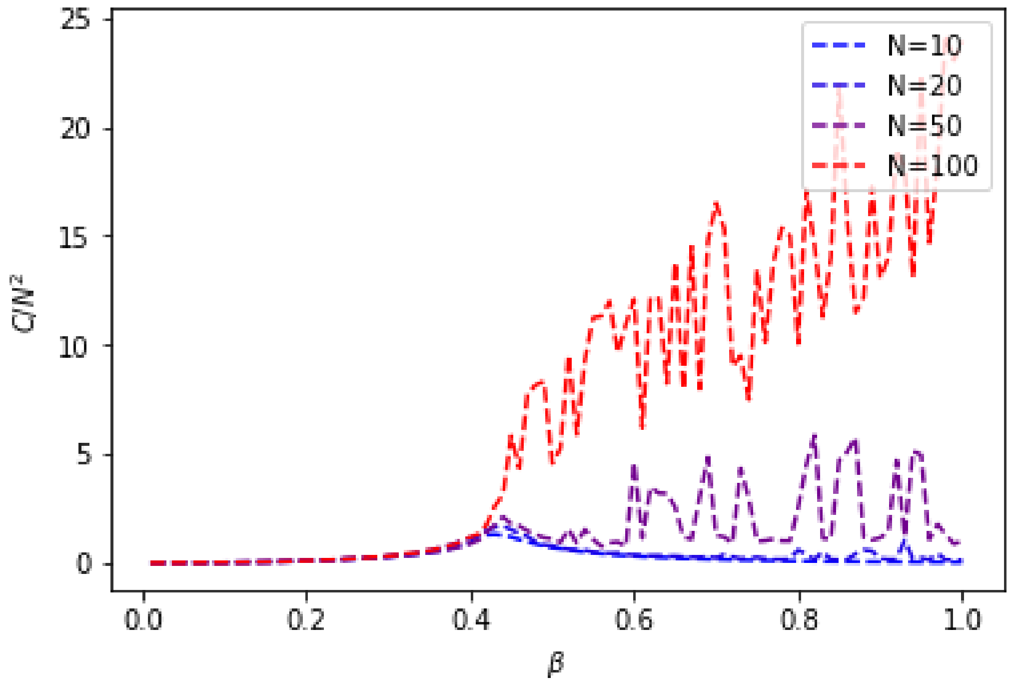
```
In [1]: # from H to find the critical temperature

        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        %matplotlib inline

        b=np.linspace(0.01,1,100)

        for i in [10,20,50,100]:
            a=pd.read_csv('./Converge/N%d/%d.csv'%(i,i),sep='\s+').values
            h=a[:,0]
            plt.plot(b,h,'--',c=((i-10)/90,0,1-(i-10)/90),label='N=%d'%i)

        plt.xlabel(r'$\beta$')
        plt.ylabel(r'$U/N^2$')
        plt.legend(loc=1)
        plt.show()

        # using N=50 for example, we find the max of |dh| to locate Tc
        a=pd.read_csv('./Converge/N50/50.csv',sep='\s+').values
        h=a[:,0]
        h2=h[1:]; h1=h[:-1]
        dh = h2-h1
        plt.plot(b[1:],dh,'r--')
        plt.xlabel(r'$\beta$')
        plt.ylabel(r'$dU/N^2$')
        plt.show()

        idx=np.where(dh == dh[:50].min())[0]
        bc = 0.01+idx*0.01
        print('the Tc=%f'%(1/bc), ', comparing the theory value=2.26918')
```




```
the Tc=2.325581 , comparing the theory value=2.26918
```

In [2]:
```
# from C to find the critical temperature

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline

b=np.linspace(0.01,1,100)

for i in [10,20,50,100]:
    a=pd.read_csv('./Converge/N%d/%d.csv'%(i,i),sep='\s+').values
    c=a[:,1]*b*b*i**2
    plt.plot(b,c,'--',c=((i-10)/90,0,1-(i-10)/90),label='N=%d'%i)

plt.xlabel(r'$\beta$')
plt.ylabel(r'$C/N^2$')
plt.legend(loc=1)
plt.show()

# using N=50 for example, we find the max of |dh| to locate Tc
a=pd.read_csv('./Converge/N50/50.csv',sep='\s+').values
c=a[:,1]
# for low temperature, the sampling is not sufficient
# so behave poor with some peaks, here we only peak the
# first peak for beta in (0,0.5)
idx=np.where(c == c[:50].max())[0]
bc = 0.01+idx*0.01
print('the Tc=%f'%(1/bc), ', comp. the theory val=2.26918')
```



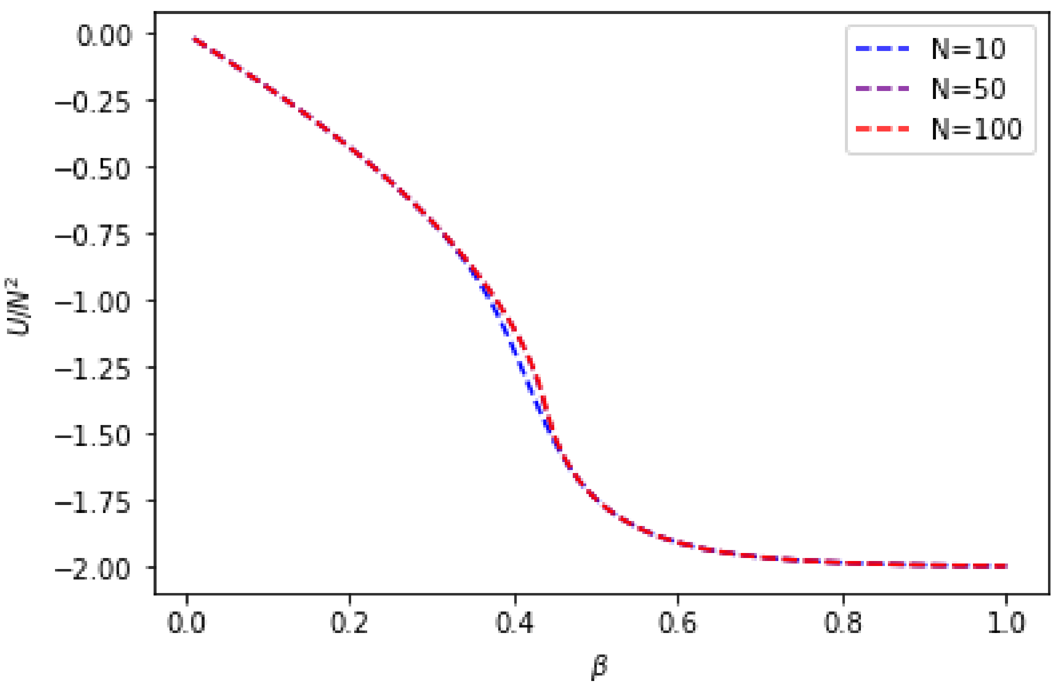

```
the Tc=2.272727 , comp. the theory val=2.26918
```

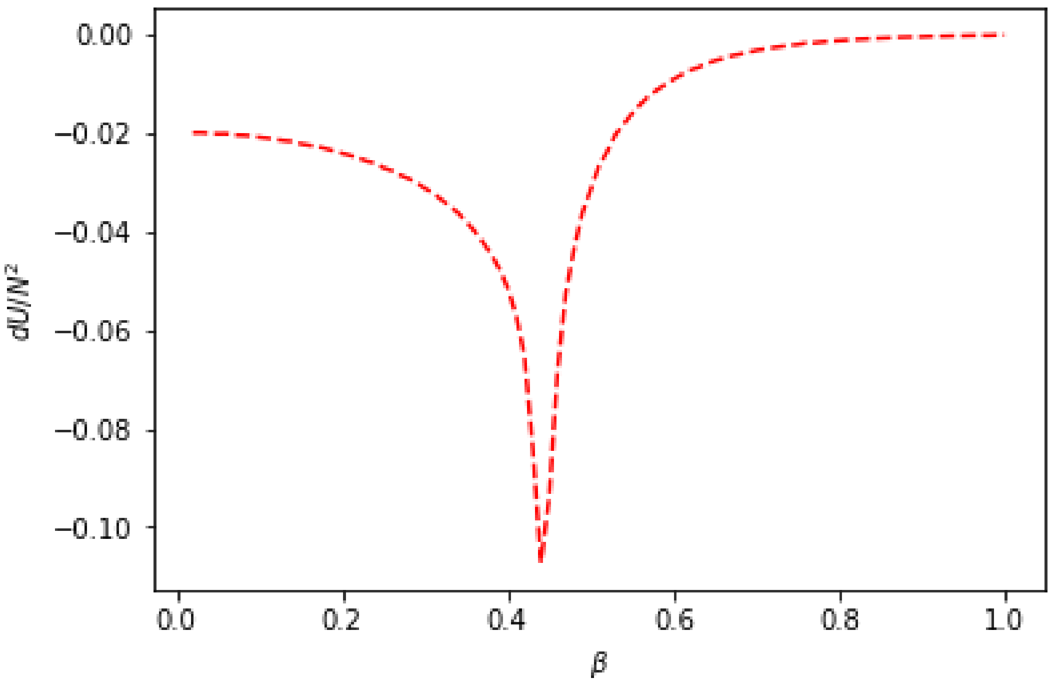
As we see, for $100 \times 100$ grid, the gribbs sampling is not sufficient to be ergodic, especially for low temperature (or large $\beta$), so it is better to use other sampling methods.

### 2.2 Swendsen-Wang sampling

```
In [3]: # from H to find the critical temperature

        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        %matplotlib inline

        b=np.linspace(0.01,1,100)

        for i in [10,50,100]:
            a=pd.read_csv('./SwendsenWang/N%d/%d.csv'%(i,i),sep='\s+').values
            h=a[:,0]
            plt.plot(b,h,'--',c=((i-10)/90,0,1-(i-10)/90),label='N=%d'%i)

        plt.xlabel(r'$\beta$')
        plt.ylabel(r'$U/N^2$')
```

```
plt.legend(loc=1)
plt.show()

# using N=50 for example, we find the max of |dh| to locate Tc
a=pd.read_csv('./SwendsenWang/N50/50.csv',sep='\s+').values
h=a[:,0]
h2=h[1:]; h1=h[:-1]
dh = h2-h1
plt.plot(b[1:],dh,'r--')
plt.xlabel(r'$\beta$')
plt.ylabel(r'$dU/N^2$')
plt.show()

idx=np.where(dh == dh[:50].min())[0]
bc = 0.01+idx*0.01
print('the Tc=%f'%(1/bc), ', comparing the theory value=2.26918')
```



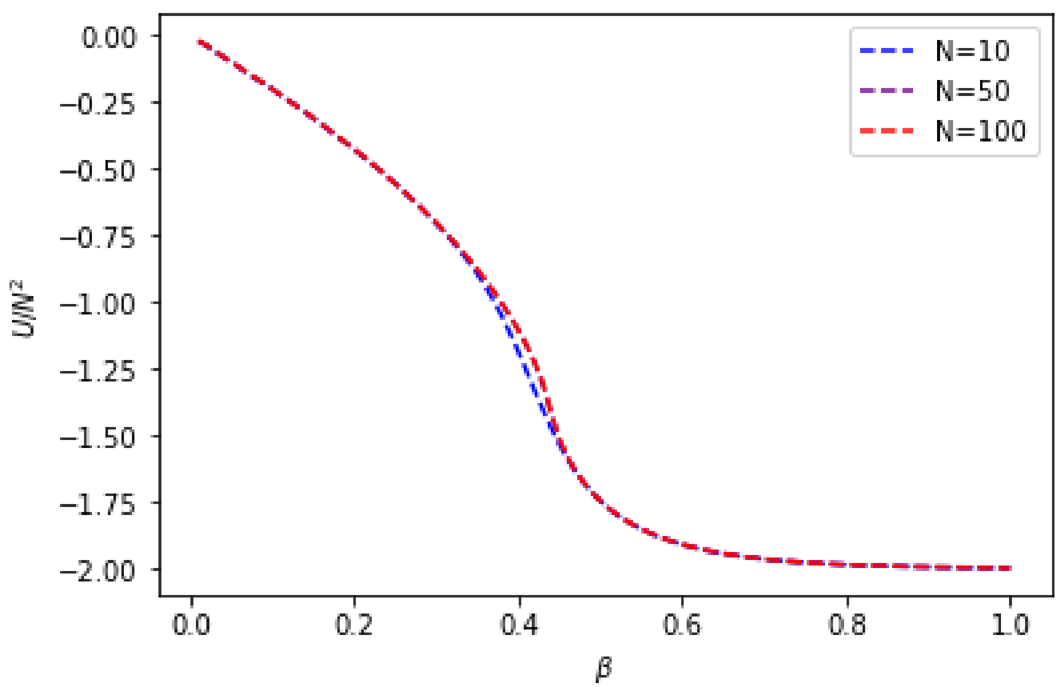


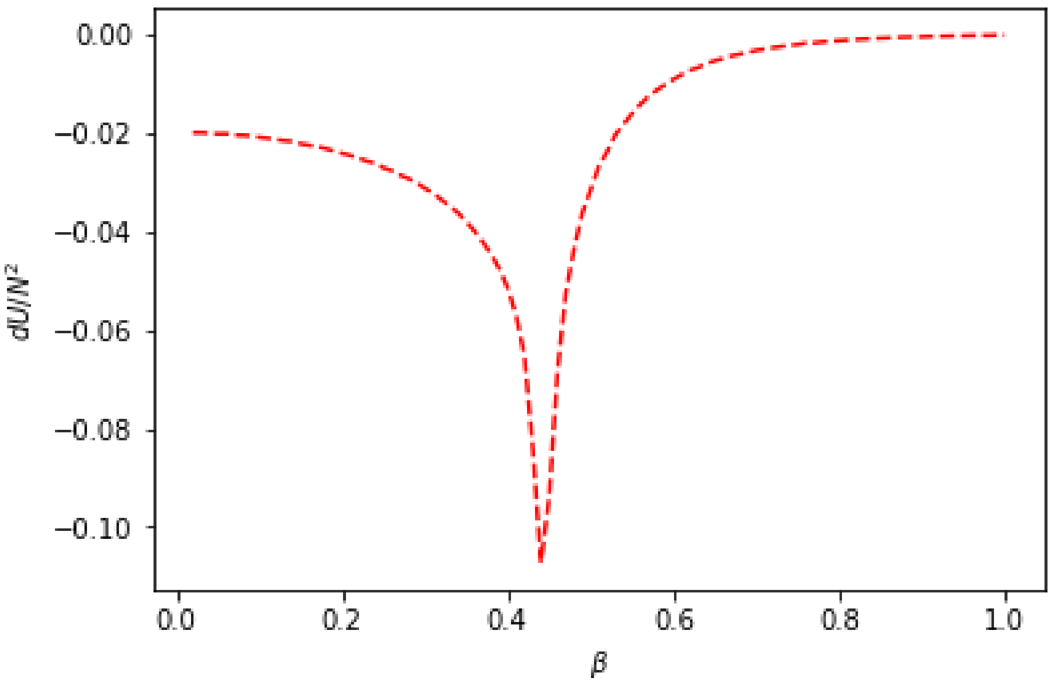

```
the Tc=2.325581 , comparing the theory value=2.26918
```

### 2.3 Wolff sampling

```
In [4]: # from H to find the critical temperature

        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        %matplotlib inline

        b=np.linspace(0.01,1,100)

        for i in [10,50,100]:
            a=pd.read_csv('./Wolff/N%d/%d.csv'%(i,i),sep='\s+').values
            h=a[:,0]
            plt.plot(b,h,'--',c=((i-10)/90,0,1-(i-10)/90),label='N=%d'%i)

        plt.xlabel(r'$\beta$')
        plt.ylabel(r'$U/N^2$')
        plt.legend(loc=1)
        plt.show()
```

```
# using N=50 for example, we find the max of |dh| to locate Tc
a=pd.read_csv('./SwendsenWang/N50/50.csv',sep='\s+').values
h=a[:,0]
h2=h[1:]; h1=h[:-1]
dh = h2-h1
plt.plot(b[1:],dh,'r--')
plt.xlabel(r'$\beta$')
plt.ylabel(r'$dU/N^2$')
plt.show()

idx=np.where(dh == dh[:50].min())[0]
bc = 0.01+idx*0.01
print('the Tc=%f'%(1/bc), ', comparing the theory value=2.26918')
```

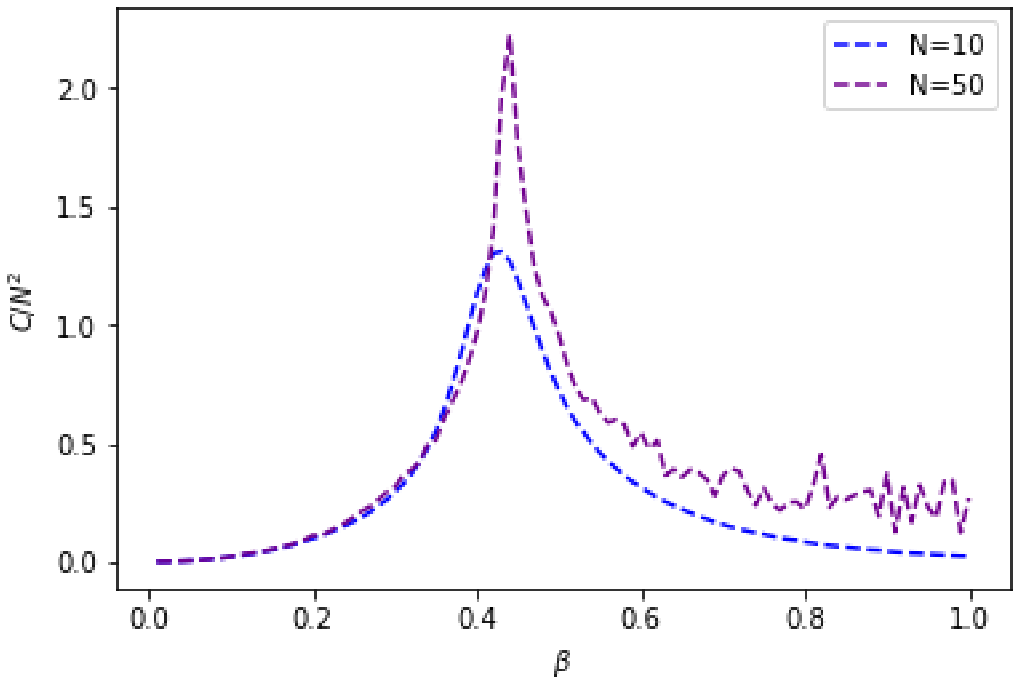





the Tc=2.325581 , comparing the theory value=2.26918

```
In [5]: # from C to find the critical temperature

        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        %matplotlib inline

        b=np.linspace(0.01,1,100)

        for i in [10,50]:
            a=pd.read_csv('./Wolff/N%d/%d.csv'%(i,i),sep='\s+').values
            c=a[:,1]*b*b*i*i
            plt.plot(b,c,'--',c=((i-10)/90,0,1-(i-10)/90),label='N=%d'%i)

        plt.xlabel(r'$\beta$')
        plt.ylabel(r'$C/N^2$')
        plt.legend(loc=1)
        plt.show()

        # using N=50 for example, we find the max of |dh| to locate Tc
        a=pd.read_csv('./Converge/N50/50.csv',sep='\s+').values
```

```
c=a[:,1]
# for low temperature, the sampling is not sufficient
# so behave poor with some peaks, here we only peak the
# first peak for beta in (0,0.5)
idx=np.where(c == c[:50].max())[0]
bc = 0.01+idx*0.01
print('the Tc=%f'%(1/bc), ', comp. the theory val=2.26918')
```



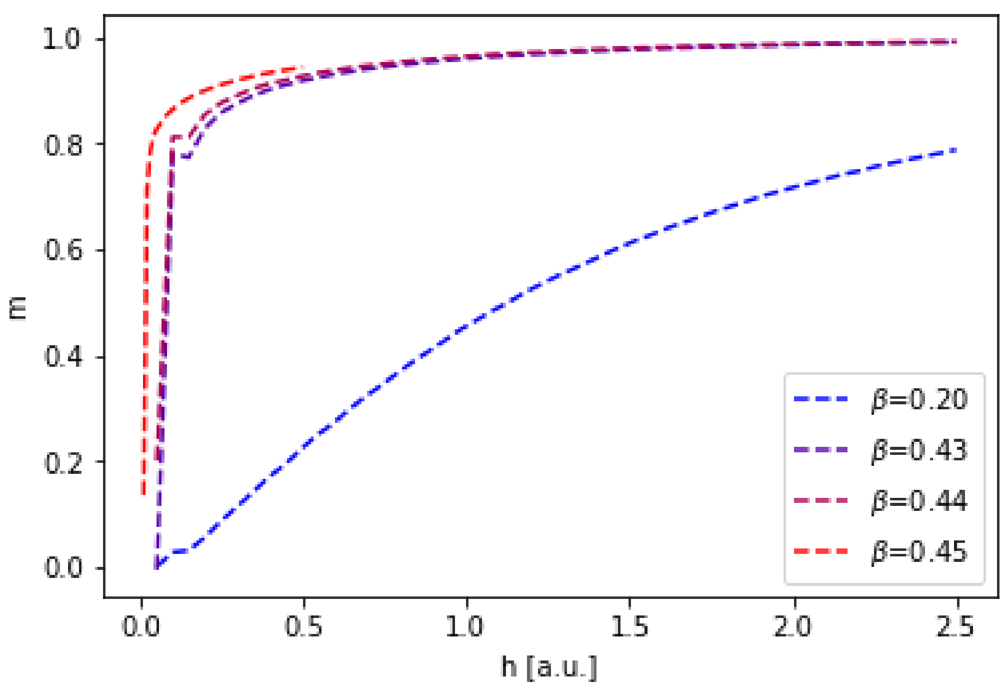

```
the Tc=2.272727 , comp. the theory val=2.26918
```

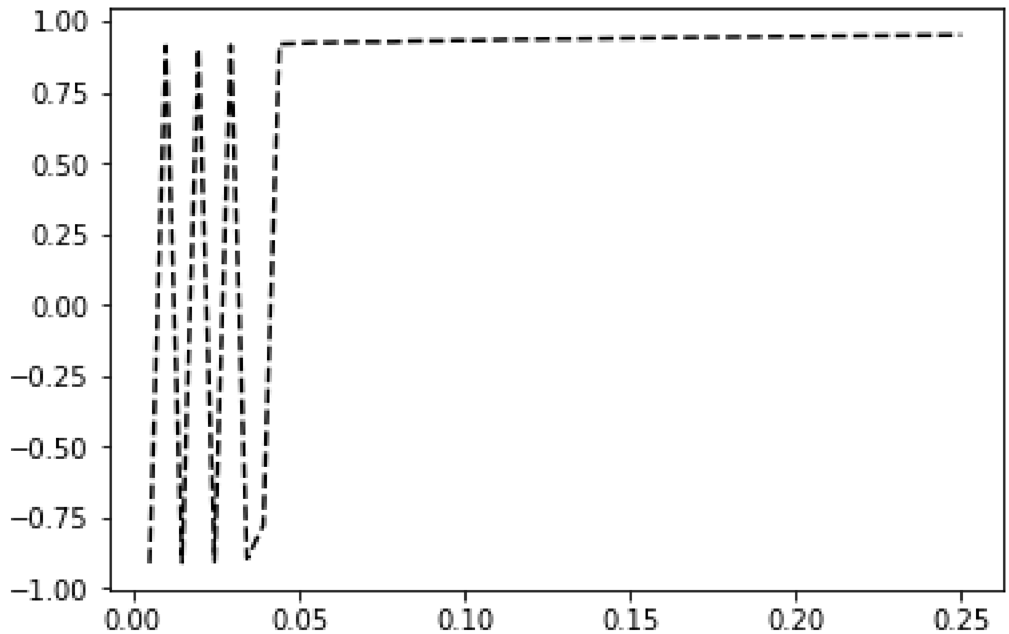
## 3 Plot m~h at different $\beta$

at $\beta = 0.2, 0.43, 0.44, 0.45, 0.5$, its gives $m \sim h$ like (onte the curve must to be centrosymmetric, so we here only plot when $h > 0$ side):

```
In [6]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd

        h=np.linspace(1,50,51)

        # for beta = 0.20, 0.43, 0.44, 0.45 (where T_c = 0.441 )
        # for beta=0.45 > T_c , the m is not zero when h approch 0
```

```
j=0
for i in ['0.20','0.43','0.44','0.45']:
    a=pd.read_csv('Magnetic/b'+i+'/m'+i+'.csv',sep='\s+').values[:,0]
    if i=='0.45':
        plt.plot(h*0.01,a,'--',c=(j/3,0,1-j/3), label=r'$\beta$='+i)
    elif i=='0.50':
        plt.plot(h*0.005,a,'-',c=(j/3,0,1-j/3), label=r'$\beta$='+i)
    else:
        plt.plot(h*0.05,a,'--',c=(j/3,0,1-j/3), label=r'$\beta$='+i)
    j+=1
plt.legend(loc=4)
plt.xlabel('h [a.u.]')
plt.ylabel('m')
plt.show()

# for beta = 0.50, with h=0, the m is Spontaneous magnetization, m never
# equals zero
a=pd.read_csv('Magnetic/b0.50/m0.50.csv',sep='\s+').values[:,0]
plt.plot(h*0.005,a,'k--', label=r'$\beta$='+i)
plt.show()
```






the result show that: for temperature near critical temperature, there will be a spontaneous magnetization, which will lead non-zero $\langle M \rangle$ at zeros magnetic field.

# 4 Study of correlation length $\xi$

- at different $\beta$ and different $N$
  the following simulation is at $N = 100$, using 1000000 steps, each step average all spin-spin correlation function. Here have total $N^2$ number of $\sigma(i)$, and each have 2 another spin with a distance of $r$.

  $$\Gamma(r) = \langle \frac{1}{2N^2} \sum_i \sigma(i)\sigma(i+r) \rangle$$

  here $\xi$ is calculated from $\Gamma(r)$ for:

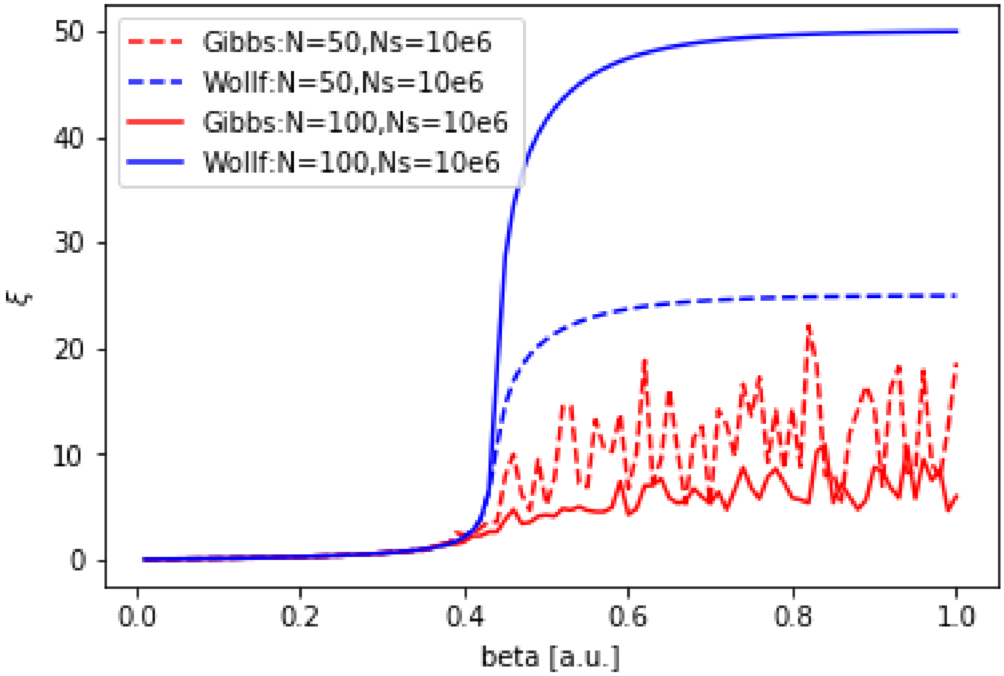
  $$\xi(\beta) = \sum_{k=1}^{\text{cut off}} \Gamma_\beta(k)$$

  for large $\beta$, the temperatue is too low to use Gibbs sampling to converge the result.

  The result also compare with the wolff sampling of a simulation at $N = 100$, using 1000000 steps, and for Wolff sampling (even Swenden-Wang sampling), when beta is large, the correlation will be long range, and always stay spontaneous magnetization. (for we always throw a random number to decide the whole spin direction of a region, this step makes the model always lang correlation at vary low temperature).

```
In [7]: import numpy as np
        import pandas as pd
```

```
import matplotlib.pyplot as plt
%matplotlib inline

# Gibbs sampling
a=pd.read_csv('./Correlation/N50/Gibbs/length.csv',header=None,
sep='\s+').values
b=np.linspace(0.01,1,100)
l=a[:,0]
plt.plot(b,l,'r--',label='Gibbs:N=50,Ns=10e6')

# Wolff sampling
a=pd.read_csv('./Correlation/N50/Wolff/length.csv',header=None,
sep='\s+').values
b=np.linspace(0.01,1,100)
l=a[:,0]
plt.plot(b,l,'b--',label='Wollf:N=50,Ns=10e6')

# Gibbs sampling
a=pd.read_csv('./Correlation/N100/Gibbs/length.csv',header=None,
sep='\s+').values
b=np.linspace(0.01,1,100)
l=a[:,0]
plt.plot(b,l,'r-',label='Gibbs:N=100,Ns=10e6')

# Wolff sampling
a=pd.read_csv('./Correlation/N100/Wolff/length.csv',header=None,
sep='\s+').values
b=np.linspace(0.01,1,100)
l=a[:,0]
plt.plot(b,l,'b-',label='Wollf:N=100,Ns=10e6')

plt.ylabel(r'$\xi$')
plt.xlabel('beta [a.u.]')

plt.legend(loc=0)
plt.show()
```




# 5 Study of critical exponent

## 5.1 Study behavior of heat capacity

First we see relation of

$$c = c_0|1 - \frac{T}{T_c}|^{-\gamma}$$

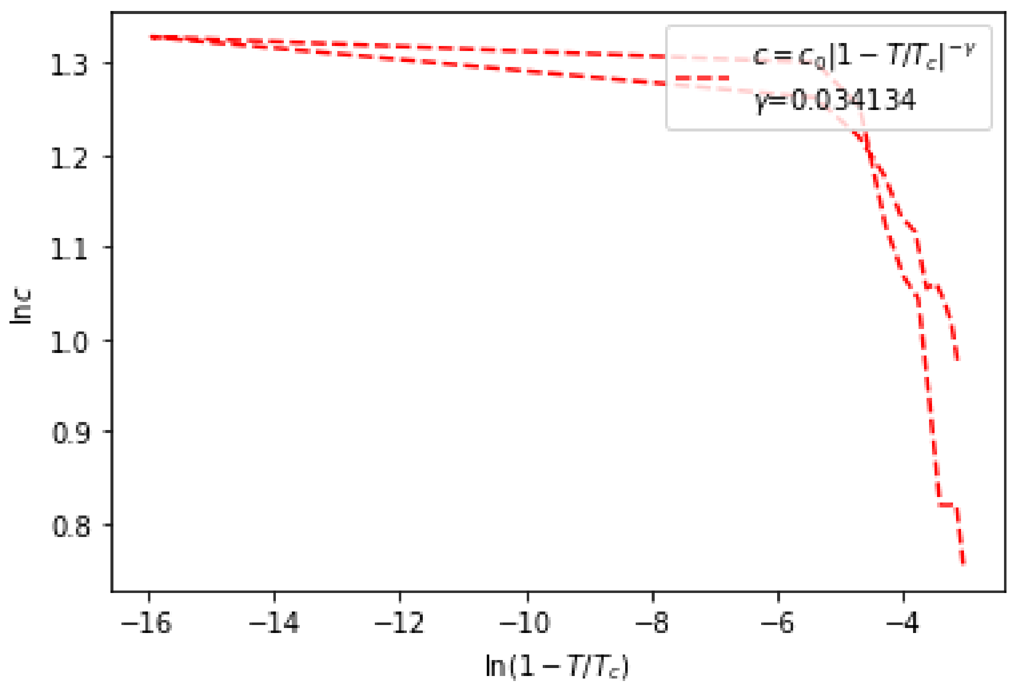
for better discription of heat capacity, here we use Wolff sampling, dealing with $100 \times 100$ grid for example.
With step $N = 10^6$, and work type 1. choose the $\beta$ in the critical area of $[0.42, 0.46]$, and use the empirical critical point with $T_c = 2.269$.
The result compare with the theotical value $\gamma = 0$ for $C$ is singular at $T_c$:

$$(C/N) \sim (8k_B/\pi)(\beta J)^2 \ln[1/(T - T_c)]$$

```
In [8]: import numpy as np
        import matplotlib.pyplot as plt
        import scipy.optimize as optimization

        def func(params, xdata, ydata):
            return (ydata - numpy.dot(xdata, params))

        # using N=100 for example, we find the max of |dh| to locate Tc
        b=np.linspace(0.42,0.46,21)
```

```
e = np.abs(1-1/(b*2.272727))
a=pd.read_csv('./Exponent/c/100.csv',sep='\s+').values
c=a[:,1]*100*100*b*b

lnc=np.log(c)
lne=np.log(e)

def func(x, k, m):
    return k*x + m
sigma = np.zeros(len(lne)) + 0.5
args = optimization.curve_fit(func, lne, lnc , np.array([0.0,0.0]), sigma)[0]
print('the critical exponent for heat capacity is: %f ~ 0'%(-args[0]))

plt.plot(lne,lnc,'r--',label=
r'$c=c_0 |1-T/T_c|^{-\gamma}$'+'\n'+ r'$\gamma$=%f'%(-args[0]))
plt.xlabel(r'$\ln(1-T/T_c)$')
plt.ylabel(r'$\ln c$')
plt.legend(loc=1)
plt.show()
```

```
the critical exponent for heat capacity is: 0.034134 ~ 0
```



### 5.2 Study behavior of magnetic moment

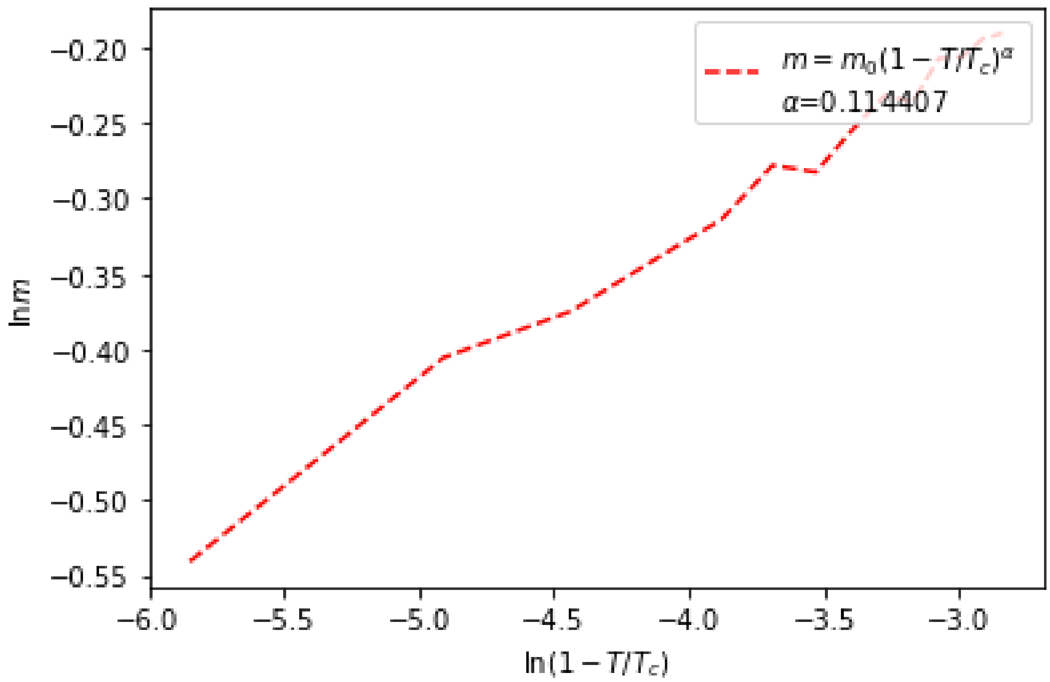
Second, we see relation of

$$m = m_0|1 - \frac{T}{T_c}|^{\alpha}, \quad T < T_c$$

Here we use gibbs sampling, dealing with $100 \times 100$ grid.
With step $N = 5 \times 10^8$, and work type 1. choose the $\beta$ in the critical area of $[0.44, 0.48]$, and use the empirical critical point with $T_c = 2.269$.
The result compare with the theotical value $\alpha = 1/8$.

```
In [9]: import numpy as np
        import matplotlib.pyplot as plt
        import scipy.optimize as optimization

        def func(x, k, m):
            return k*x + m

        # using N=50 for example, we find the max of |dh| to locate Tc
        b=np.linspace(0.44,0.48,21)
        #b=np.linspace(0.01,1,100)
        e = np.abs(1-1/(b*2.269))
        a=pd.read_csv('./Exponent/m/m100.csv',sep='\s+').values

        # here why we use abs, for h=0, only absolute of m is meaningful
        m=np.abs(a[:,0])

        # use parts of data most near the critical point
        lnm=np.log(m[1:15])
        lne=np.log(e[1:15])

        sigma = np.zeros(len(lne)) + 0.5
        args = optimization.curve_fit(func, lne, lnm, np.array([0.0,0.0]), sigma)[0]
        print('the critical exponent for m is: %f ~ 0.125'%args[0])

        plt.plot(lne,lnm,'r--',label=
        r'$m=m_0(1-T/T_c)^{\alpha}$'+'\n'+ r'$\alpha$=%f'%(args[0]))
        plt.xlabel(r'$\ln(1-T/T_c)$')
        plt.ylabel(r'$\ln m$')
        plt.legend(loc=1)
        plt.show()
```

```
the critical exponent for m is: 0.114407 ~ 0.125
```




### 5.3 Study of correlation length

last, we see relation of

$$\xi = \xi_0 |1 - \frac{T}{T_c}|^{-\delta}$$

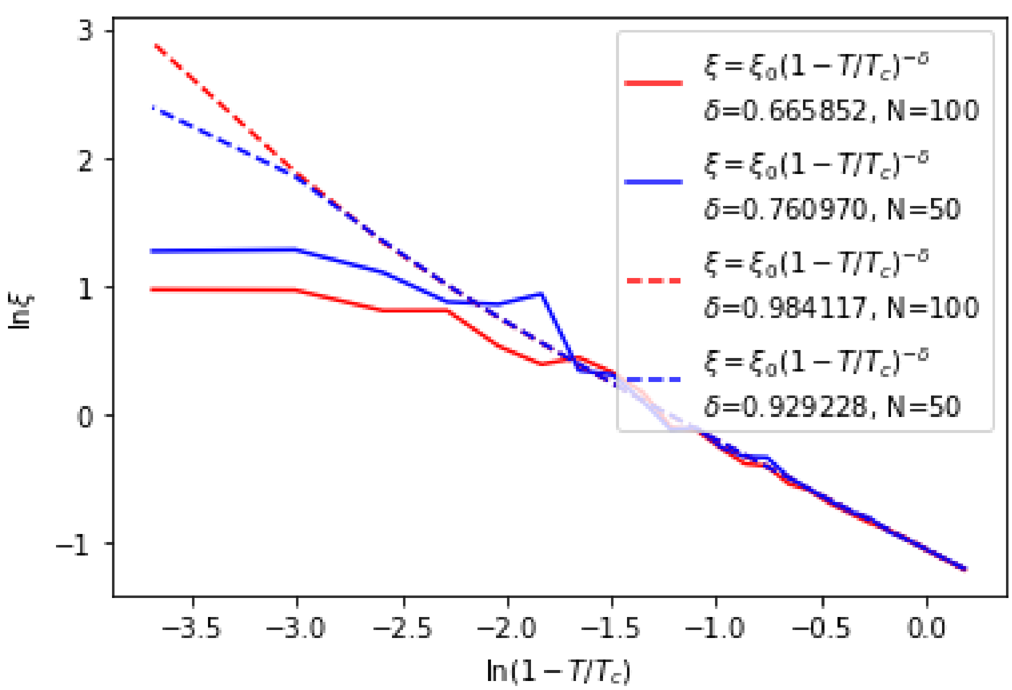
Here we use gibbs sampling, dealing with $100 \times 100$ grid.
With step $N = 5 \times 10^8$, and work type 1. choose the $\beta$ in the critical area of $[0.42, 0.46]$, and use the empirical critical point with $T_c = 2.269$ (where from the peak of C).
The result compare with the theotical value $\delta = 1$.
(for low temper, the sampling of simulation is not sufficient (for Gibbs sampling) or the throw a regoin random number problem (for Wolff/Swendsen-Wang), we only use the high temperature side data to fit the curve). But result is quite enough to illustrate the relationship.

```
In [10]: import numpy as np
         import pandas as pd
         import scipy.optimize as optimization
         import matplotlib.pyplot as plt
         %matplotlib inline

         def func(x, k, m):
             return k*x + m

         # using N=100, Gibbs sampling
         a=pd.read_csv('./Correlation/N100/Gibbs/length.csv',header=None,
```

```
sep='\s+').values
l=a[:,0]
l=l[20:44]
b=np.arange(20,44)*0.01
e = np.abs(1-1/(b*2.269))
lnl=np.log(l)
lne=np.log(e)

sigma = np.zeros(len(lne)) + 0.5
args = - optimization.curve_fit(func, lne, lnl, np.array([0.0,0.0]), sigma)[0]
plt.plot(lne,lnl,'r-',label=
r'$\xi=\xi_0(1-T/T_c)^{-\delta}$'+'\n'+ r'$\delta$=%f'%(args[0])+', N=100')
print('the critical exponent for length of (Gibbs N=100) is: %f ~ 1'%args[0])

# using N=50, Gibbs sampling
a=pd.read_csv('./Correlation/N50/Gibbs/length.csv',header=None,
sep='\s+').values
l=a[:,0]
l=l[20:44]
b=np.arange(20,44)*0.01
e = np.abs(1-1/(b*2.269))
lnl=np.log(l)
lne=np.log(e)

sigma = np.zeros(len(lne)) + 0.5
args = - optimization.curve_fit(func, lne, lnl, np.array([0.0,0.0]), sigma)[0]
plt.plot(lne,lnl,'b-',label=
r'$\xi=\xi_0(1-T/T_c)^{-\delta}$'+'\n'+ r'$\delta$=%f'%(args[0])+', N=50')
print('the critical exponent for length of (Gibbs N=50) is: %f ~ 1'%args[0])

# using N=100, Wolff sampling
a=pd.read_csv('./Correlation/N100/Wolff/length.csv',header=None,
sep='\s+').values
l=a[:,0]
l=l[20:44]
b=np.arange(20,44)*0.01
e = np.abs(1-1/(b*2.269))
lnl=np.log(l)
lne=np.log(e)

sigma = np.zeros(len(lne)) + 0.5
args = - optimization.curve_fit(func, lne, lnl, np.array([0.0,0.0]), sigma)[0]
plt.plot(lne,lnl,'r--',label=
r'$\xi=\xi_0(1-T/T_c)^{-\delta}$'+'\n'+ r'$\delta$=%f'%(args[0])+', N=100')
print('the critical exponent for length of (wolff N=100) is: %f ~ 1'%args[0])

# using N=50, Wolff sampling
a=pd.read_csv('./Correlation/N50/Wolff/length.csv',header=None,
```

```
    sep='\s+').values
    l=a[:,0]
    l=l[20:44]
    b=np.arange(20,44)*0.01
    e = np.abs(1-1/(b*2.269))
    lnl=np.log(l)
    lne=np.log(e)

    sigma = np.zeros(len(lne)) + 0.5
    args = - optimization.curve_fit(func, lne, lnl, np.array([0.0,0.0]), sigma)[0]
    plt.plot(lne,lnl,'b--',label=
    r'$\xi=\xi_0(1-T/T_c)^{-\delta}$'+'\n'+ r'$\delta$=%f'%(args[0])+', N=50')
    print('the critical exponent for length of (wolff N=50) is: %f ~ 1'%args[0])

    plt.xlabel(r'$\ln(1-T/T_c)$')
    plt.ylabel(r'$\ln \xi$')
    plt.legend(loc=1)
    plt.show()
```

```
the critical exponent for length of (Gibbs N=100) is: 0.665852 ~ 1
the critical exponent for length of (Gibbs N=50) is: 0.760970 ~ 1
the critical exponent for length of (wolff N=100) is: 0.984117 ~ 1
the critical exponent for length of (wolff N=50) is: 0.929228 ~ 1
```



As we see, near the critical point, the Gibbs sampling behaves poorly, cause a deviation from linear relation shape. But Wolff sampling behave well.