Numerical Methods to Study Critical Phenomena

for the 19th KIAS-APCTP Winter School on Statistical Physics

Dong-Hee Kim

Gwangju Institute of Science and Technology

Outline

- Data analysis with the least-square fitting

  ✓ Linear model, meaning of the error estimate, goodness of the fit

  ✓ How to use Gnuplot and Python for a curve-fit

- Fundamentals of Monte Carlo methods

  ✓ Random number, Markov Chain, Autocorrelation, Metropolis algorithm, Cluster algorithms, Critical slowing down

  ✓ Finite-size-scaling analysis with the example of the 2D Ising model

Data Fitting

Things everyone seems know but me

Dong-Hee Kim

Gwangju Institute of Science and Technology

References

“**Everything you wanted to know about Data Analysis and Fitting**

but were afraid to ask”

by A. Peter Young [arXiv:1210.3781].

Goals of data fitting

DATA

"N" DATA POINTS $= \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

UNCERTAINTY (ERROR BAR) $= \{\sigma_1, \sigma_2, \ldots, \sigma_3\}$

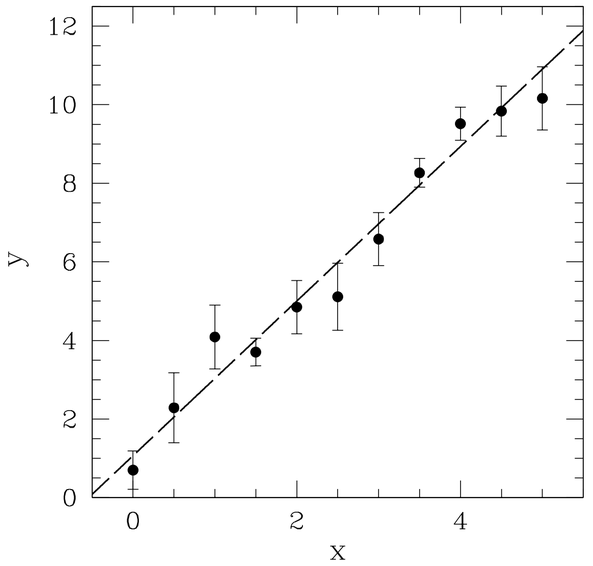
→

MODEL

$f(x)$

"M" FITTING PARAMETERS $= (a_0, a_1, \ldots, a_{M-1})$

1. Find the fit parameters.
2. Provide error estimate on the fit parameters.
3. Provide a measure of how good the fit is.



Least-square fitting

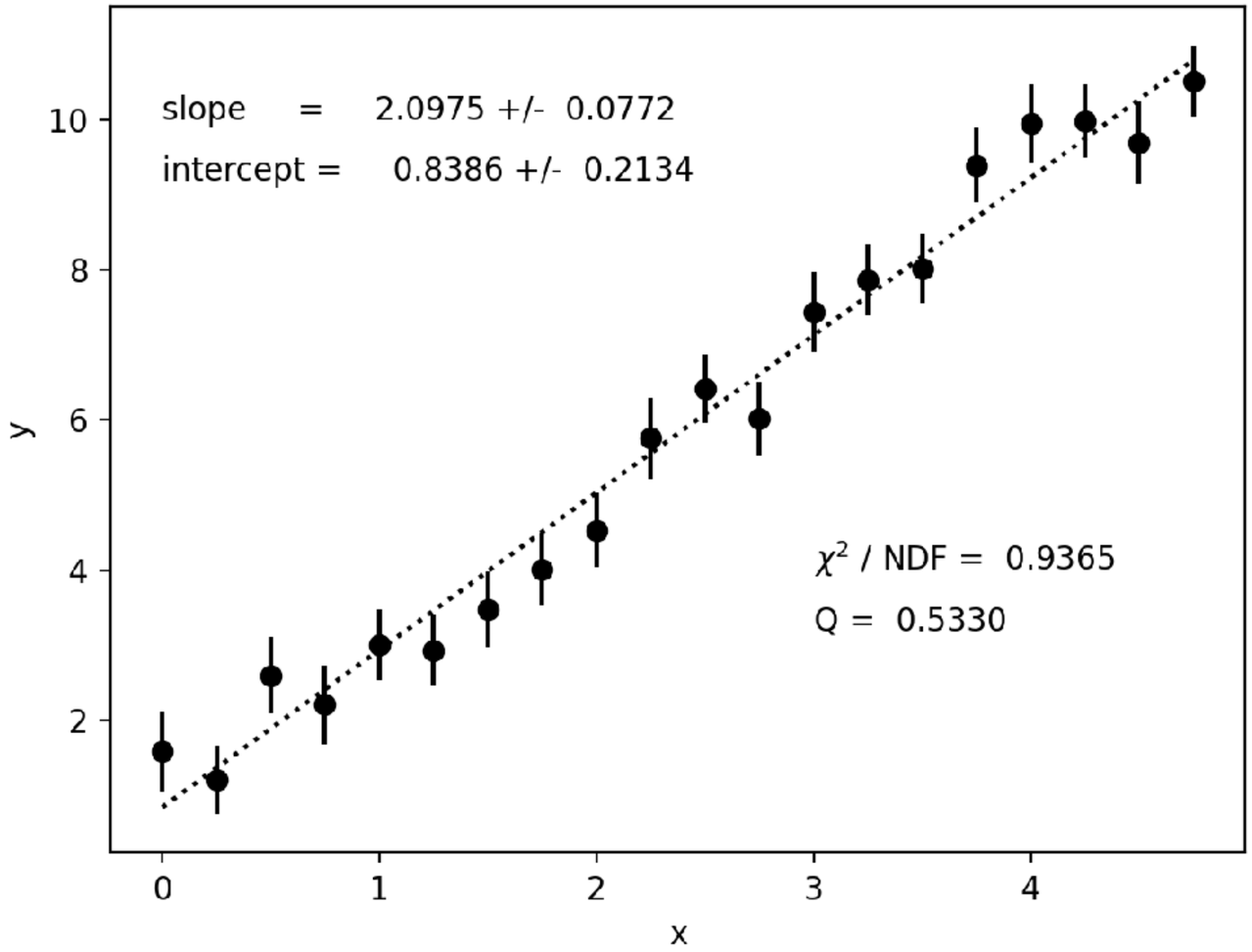
- The definition of the “best” fit is not unique.
- Probably the most popular one:

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2 \longrightarrow$$ **MINIMIZE!**

x	y	sigma
0.0	1.5891700001638700	0.5320185951347170
0.25	1.2046979031630900	0.4542722925491940
0.5	2.598484110379440	0.5077027054798450
0.75	2.2043375760987900	0.5205826529502230
1.0	3.002229375836470	0.4698090780151540
1.25	2.928655222350250	0.4773624121076510
1.5	3.4769844321649000	0.5018787752525470
1.75	4.007335261529810	0.4897362325985810
2.0	4.534192485682100	0.4946214185451030
2.25	5.76185759925369	0.5405394898833090
2.5	6.41954394976063	0.4523449367708450
2.75	6.023919923933230	0.488980475014175
3.0	7.443742735575060	0.5342659697011440
3.25	7.870686775946450	0.4730380798563350
3.5	8.022587710435350	0.4572104512757930
3.75	9.395420889477870	0.4990257792682120
4.0	9.948288024421440	0.530527413902035
4.25	9.986332770533490	0.4896967661903670
4.5	9.694674681091840	0.5443240285483060
4.75	10.519384785632700	0.4761656747219080

Fitting In Action!

Curve-Fit with Python




```
import numpy as np
from scipy.optimize import curve_fit
from scipy.special import gammaincc

func = lambda x, a0, a1 : a0 + a1 * x

x, y, err = np.loadtxt('line.data',
                       unpack = True)

NDF = x.size - 2

popt, pcov = curve_fit(func, x, y,
                       sigma = err,
                       absolute_sigma = True)

perr = np.sqrt(np.diag(pcov))

chi2 = np.sum((y - func(x,*popt))**2 / err**2)
Q = gammaincc(0.5 * NDF, 0.5 * chi2)
```

Curve-Fit with Gnuplot

```
gnuplot> f(x) = a + b * x
gnuplot> fit f(x) "line.data" using 1:2:3 yerrors via a, b
```

…

```
degrees of freedom    (FIT_NDF)                        : 18
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.967717
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.936476
p-value of the Chisq distribution (FIT_P)              : 0.532987

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a                  = 0.838641      +/- 0.2066       (24.63%)
b                  = 2.09749       +/- 0.07469      (3.561%)

correlation matrix of the fit parameters:
                   a      b
a                   1.000
b                  -0.856  1.000
```

!!!

Different from the previous ones!

0.2134

0.0772

Which ones are the right ones?

Chi-Squared

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2$$

- Assume the distribution of the deviation is "***Gaussian***" and uncorrelated.
- Assume $f(x)$ is *exact*.
- So, $\chi^2 =$ sum of N squares of Gaussian random variables (normalized).
- It may not work if your data have "outliers."
- In minimization, there are $(N - M)$ independent variables.

$$N_{\mathrm{DOF}} = N - M \quad \textit{(Degrees of Freedom)}$$

Why do you assume Gaussian?

If y_i is sampled from a parent Gaussian distribution with mean $f(x_i) = a_0 + a_1 x$ and variance σ^2,

(Likelihood of y_i)

$$L_i = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_i - f(x_i))^2}{2\sigma^2}\right] .$$

If the data points are uncorrelated,

(Likelihood of $\{y_1, \ldots, y_N\}$)

$$L = \prod_{i=1}^{N} L_i \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - f(x_i)}{\sigma}\right)^2\right] = \exp\left[-\frac{1}{2}\chi^2\right] .$$

Maximum Likelihood ⟶ **Minimizing Chi-Squared**

Central Limit Theorem

- Sum of many random variables → Gaussian distribution
- Independent, identically distributed random variables, of course.
- Finite average and variance

$$X = \sum_{i=1}^{N} x_i$$

$$\mu = \langle x \rangle$$

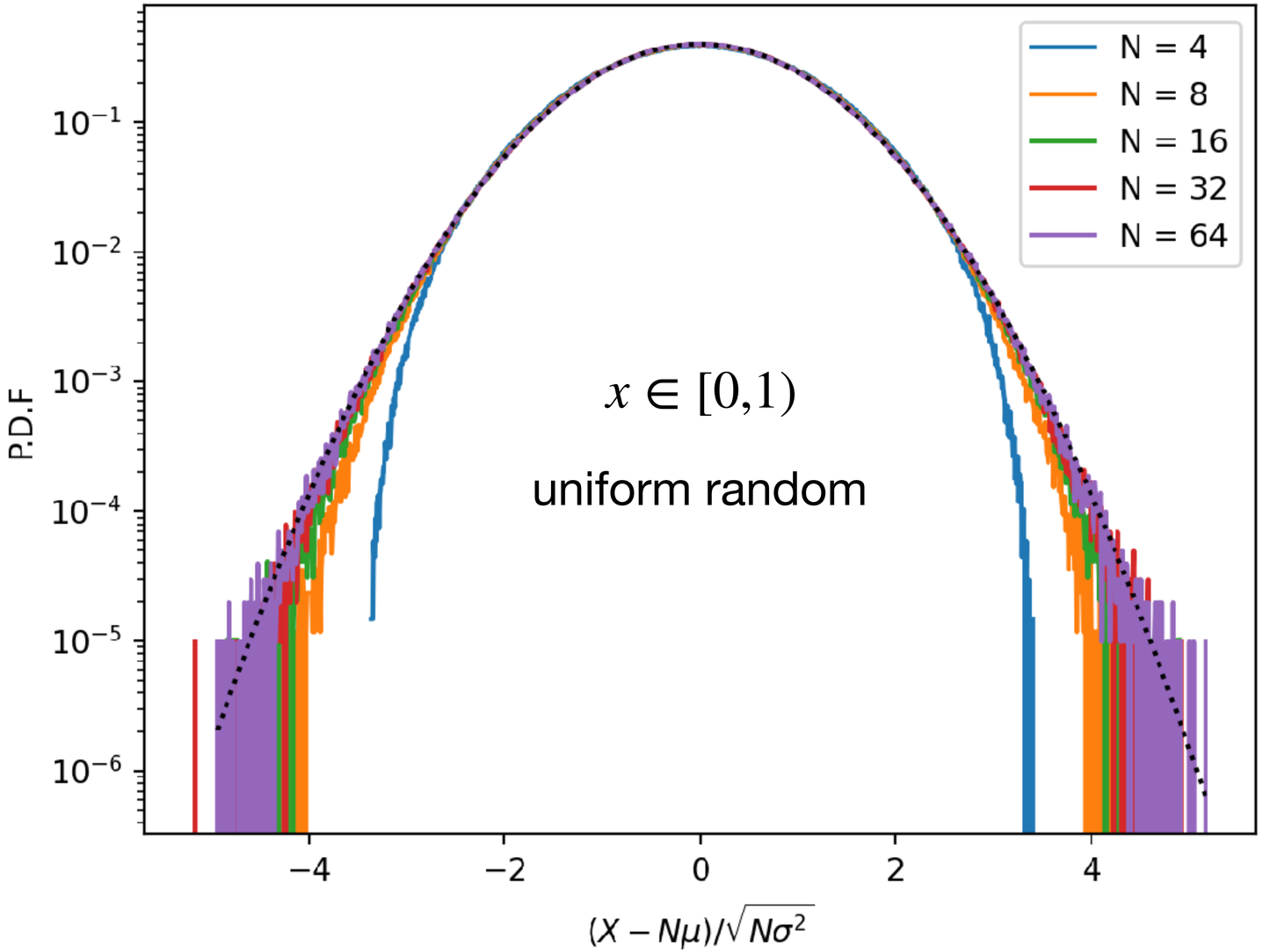
$$\sigma^2 = \langle x^2 \rangle - \langle x \rangle^2$$

$$N \to \infty$$

$$P_N(X) = \frac{1}{\sqrt{2\pi N\sigma^2}} \exp\left[-\frac{(X - N\mu)^2}{2N\sigma^2}\right]$$

$$\langle X \rangle = N\mu \qquad \langle X^2 \rangle - \langle X \rangle^2 = N\sigma^2$$

Central Limit Theorem



Linear model

The simplest case (M=2) : $f(x) = a_0 + a_1 x$

$$\chi^2(a_0, a_1) = \sum_{i=1}^{N} \left(\frac{y_i - a_0 - a_1 x_i}{\sigma_i} \right)^2$$

Minimization →

$$\begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \end{pmatrix}$$

$$U_{\alpha\beta} = \sum_{i=1}^{N} \frac{x_i^{\alpha+\beta}}{\sigma_i^2} \qquad v_\alpha = \sum_{i=1}^{N} \frac{y_i x_i^\alpha}{\sigma_i^2}$$

(symmetric, constant) *(Sum of sample values)*

They are Gaussian if the data noise is Gaussian!

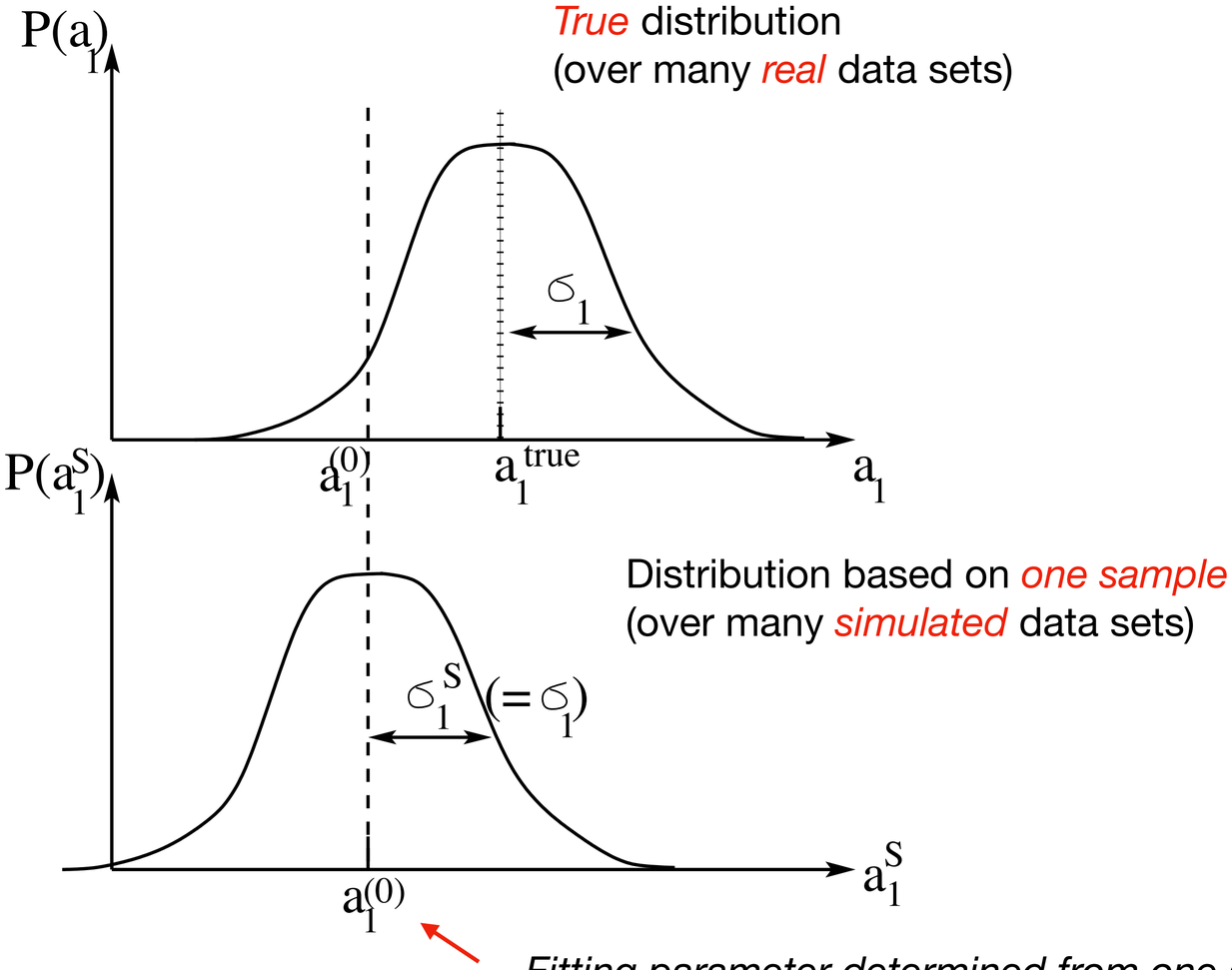
←

$$a_0 = \frac{U_{11} v_0 - U_{01} v_1}{U_{00} U_{11} - U_{01}^2}$$

$$a_1 = \frac{-U_{01} v_0 + U_{00} v_1}{U_{00} U_{11} - U_{01}^2}$$

U^{-1} ***: covariance matrix***

Error estimates



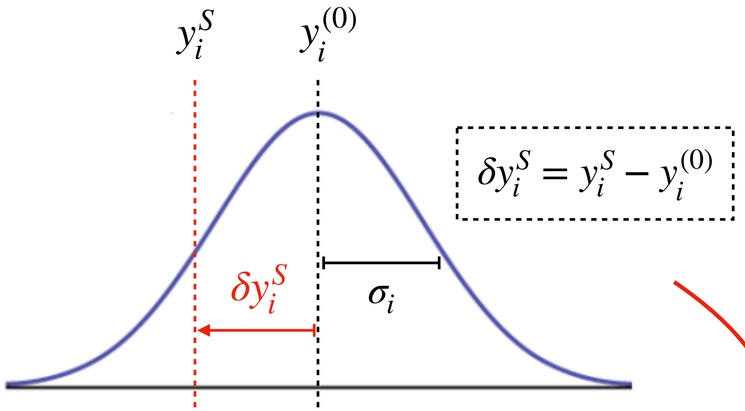
For the linear model,

$$\sigma_1^S = \sigma_1$$

(Simulated error bar) = (True error bar)

Error estimate

Simulated distribution centered at $y_i^{(0)}$



$$\delta y_i^S = y_i^S - y_i^{(0)}$$

$$a_\alpha \equiv a_\alpha(\{y_i\})$$

$$a_\alpha = \sum_\beta \left[\mathrm{U}^{-1}\right]_{\alpha\beta} v_\beta$$

$$U_{\alpha\beta} = \sum_{i=1}^{N} \frac{x_i^{\alpha+\beta}}{\sigma_i^2}$$

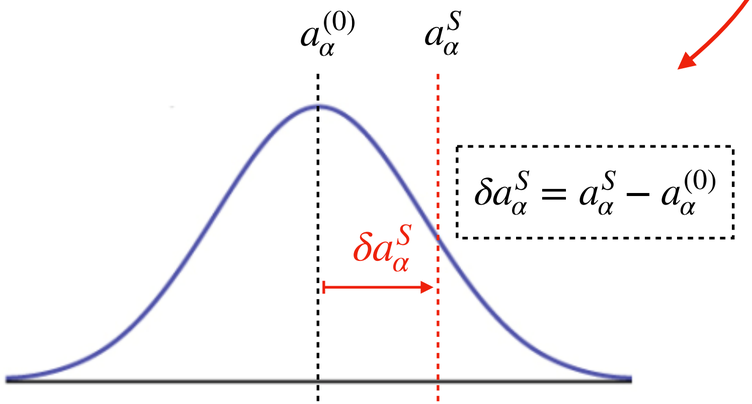
$$v_\alpha = \sum_{i=1}^{N} \frac{y_i x_i^\alpha}{\sigma_i^2}$$

$$\delta a_\alpha^S = \sum_{i=1}^{N} \frac{\partial a_\alpha}{\partial y_i} \delta y_i^S$$

*uncorrelated δy

$$\rightarrow \quad (\sigma_\alpha^S)^2 \equiv \langle (\delta a_\alpha^S)^2 \rangle = \sum_{i=1}^{N} \sigma_i^2 \left(\frac{\partial a_\alpha}{\partial y_i} \right)^2$$

$$(\sigma_\alpha^S)^2 = \sum_{\beta,\gamma} \left[\mathrm{U}^{-1}\right]_{\alpha\beta} \left[\mathrm{U}^{-1}\right]_{\alpha\gamma} \left[\sum_{i=1}^{N} \frac{x_i^{\beta+\gamma}}{\sigma_i^2} \right] = \left[\mathrm{U}^{-1}\right]_{\alpha\alpha}$$



$$\delta a_\alpha^S = a_\alpha^S - a_\alpha^{(0)}$$

Error estimate of the fitting parameter

$$(\sigma_\alpha^S)^2 = \left[\mathrm{U}^{-1}\right]_{\alpha\alpha}$$

i.e. it's independent of "S": $\sigma_\alpha^S = \sigma_\alpha$

Covariance matrix

Correlation between fluctuations of fitting parameters

$$\mathrm{Cov}(\alpha,\beta) = \langle \delta a_\alpha \, \delta a_\beta \rangle = \sum_{i,j} \frac{\partial a_\alpha}{\partial y_i} \frac{\partial a_\beta}{\partial y_j} \langle \delta y_i \, \delta y_j \rangle = \sum_i \sigma_i^2 \frac{\partial a_\alpha}{\partial y_i} \frac{\partial a_\beta}{\partial y_i} = \sum_{\mu,\nu} \left[\mathbf{U}^{-1}\right]_{\alpha\mu} \left[\mathbf{U}^{-1}\right]_{\beta\nu} \mathbf{U}_{\nu\mu} = \left[\mathbf{U}^{-1}\right]_{\alpha\beta}$$

Correlation coefficient

$$r_{\alpha\beta} = \frac{\mathrm{Cov}(\alpha,\beta)}{\sigma_\alpha \sigma_\beta} = \frac{\left[\mathbf{U}^{-1}\right]_{\alpha\beta}}{\sigma_\alpha \sigma_\beta}$$

$$a_\alpha = \sum_\beta \left[\mathrm{U}^{-1}\right]_{\alpha\beta} \sum_{i=1}^{N} \frac{y_i x_i^\beta}{\sigma_i^2}$$

$$U_{\alpha\beta} = \sum_{i=1}^{N} \frac{x_i^{\alpha+\beta}}{\sigma_i^2}$$

Quality of the fit

Null hypothesis (Chi-squared distribution)

$$P^{(m)}(X) = \frac{1}{2^{m/2}\Gamma(m/2)} X^{(m/2)-1} e^{-X/2}$$

$$X \equiv \chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2$$

$$m = N - M \equiv N_{\mathrm{DOF}}$$

1. Generate a sample of random variables $\{y_0, \ldots, y_N\}$ from their parent Gaussian distributions.
2. Minimize χ^2.
3. Repeat [1-2] to generate many χ^2 samples
4. Now, you have the probability distribution of χ^2 under *null* hypothesis.

Derivation of χ^2-distribution

Residual (Gaussian random variable, $\mathcal{N}(0,1)$)

$$\epsilon_i = \frac{y_i - a_0 - a_1 x_i}{\sigma_i}$$

Two equation (minimizing χ^2)

$$\sum_{i=1}^{N} \frac{1}{\sigma_i}\epsilon_i = 0 \quad \sum_{i=1}^{N} \frac{x_i}{\sigma_i}\epsilon_i = 0 \longrightarrow \mathbf{M}\boldsymbol{\epsilon} = 0$$

(SVD)

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

Number of independent random variables

$$m = N - 2 \equiv N_{\mathrm{DOF}}$$

Independent random variables spanning the null space ($\mathbf{M}\boldsymbol{\epsilon} = 0$)

$$\mathbf{z} = \mathbf{V}^T\boldsymbol{\epsilon}$$

($\mathcal{N}(0,1)$ **N** → **N_{DOF}** random variables)

$$\|\mathbf{z}\|_2^2 = \|\boldsymbol{\epsilon}\|_2^2 = \chi^2$$

(orthogonal transformation)

Derivation of χ^2-distribution

Joint probability distribution of $(z_0, z_1, \ldots, z_m)$

Surface of a unit m-dim. sphere

$$S_m = \frac{2\pi^{m/2}}{\Gamma(m/2)}$$

$$P(\mathbf{z})d\mathbf{z} = \prod_{i=1}^{m} P(z_i)\, dz_i = \prod_{i=1}^{m} \frac{dz_i}{\sqrt{2\pi}} e^{-z_i^2/2} \longrightarrow P^{(m)}(r)\, dr = \frac{S_m}{(2\pi)^{m/2}} r^{m-1} e^{-r^2/2}\, dr$$

$$r^2 = \sum_i z_i^2$$

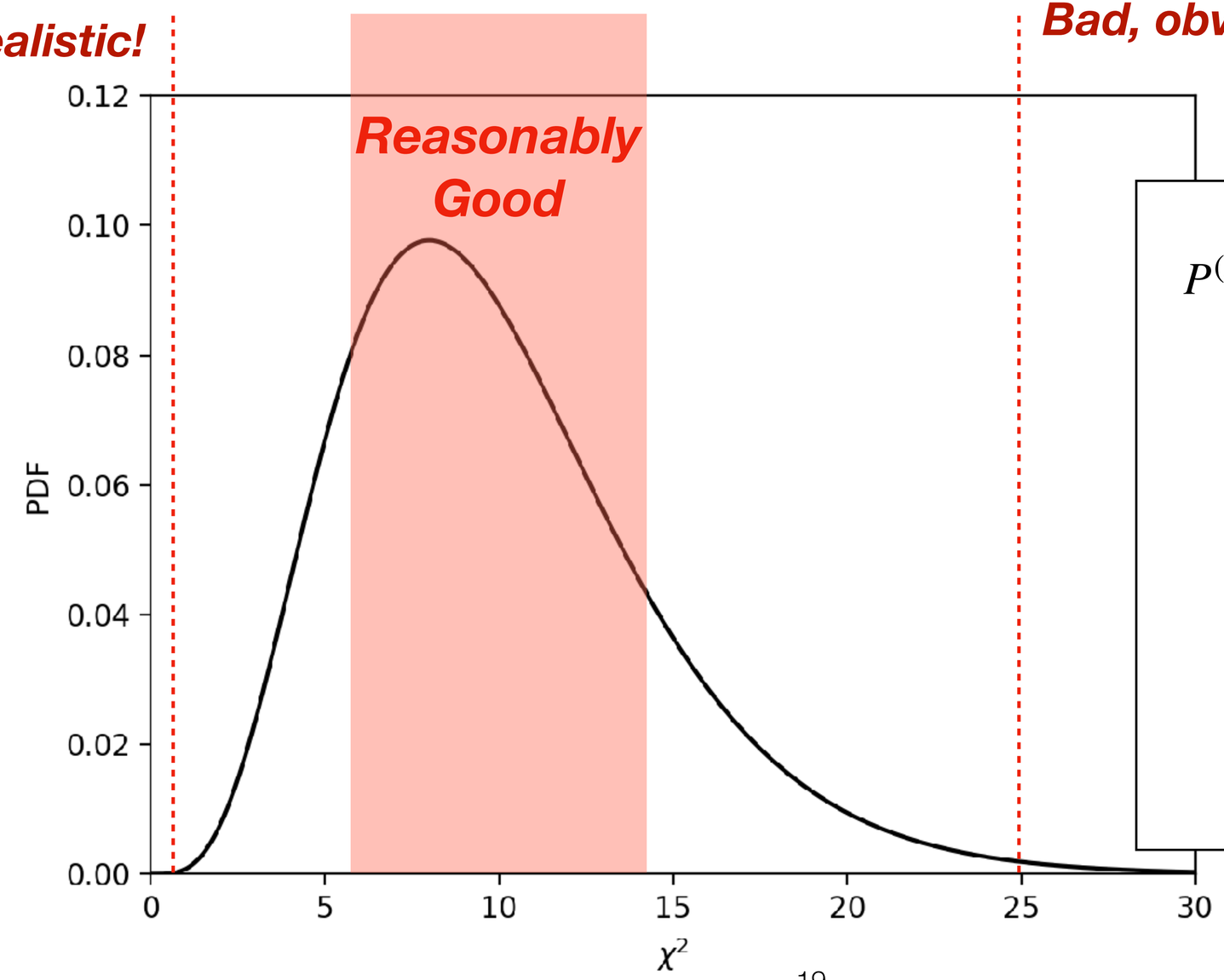
Chi-Squared distribution $(X \equiv \chi^2)$

$$P^{(m)}(X)dX = P^{(m)}(r)dr \longrightarrow P^{(m)}(X) = \frac{1}{2^{m/2}\Gamma(m/2)} X^{(m/2)-1} e^{-X/2}$$

$$1 - s\sqrt{\frac{2}{N_{\mathrm{DOF}}}} \lesssim \frac{\chi^2}{N_{\mathrm{DOF}}} \lesssim 1 + s\sqrt{\frac{2}{N_{\mathrm{DOF}}}}$$

Unrealistic!

Bad, obviously.



$$P^{(m)}(X) = \frac{1}{2^{m/2}\Gamma(m/2)} X^{(m/2)-1} e^{-X/2}$$

$$X \equiv \chi^2$$

$$N_{\mathrm{DOF}} \equiv m = 10$$

$$\langle X \rangle = N_{\mathrm{DOF}}$$

$$\sigma_X^2 = 2N_{\mathrm{DOF}}$$

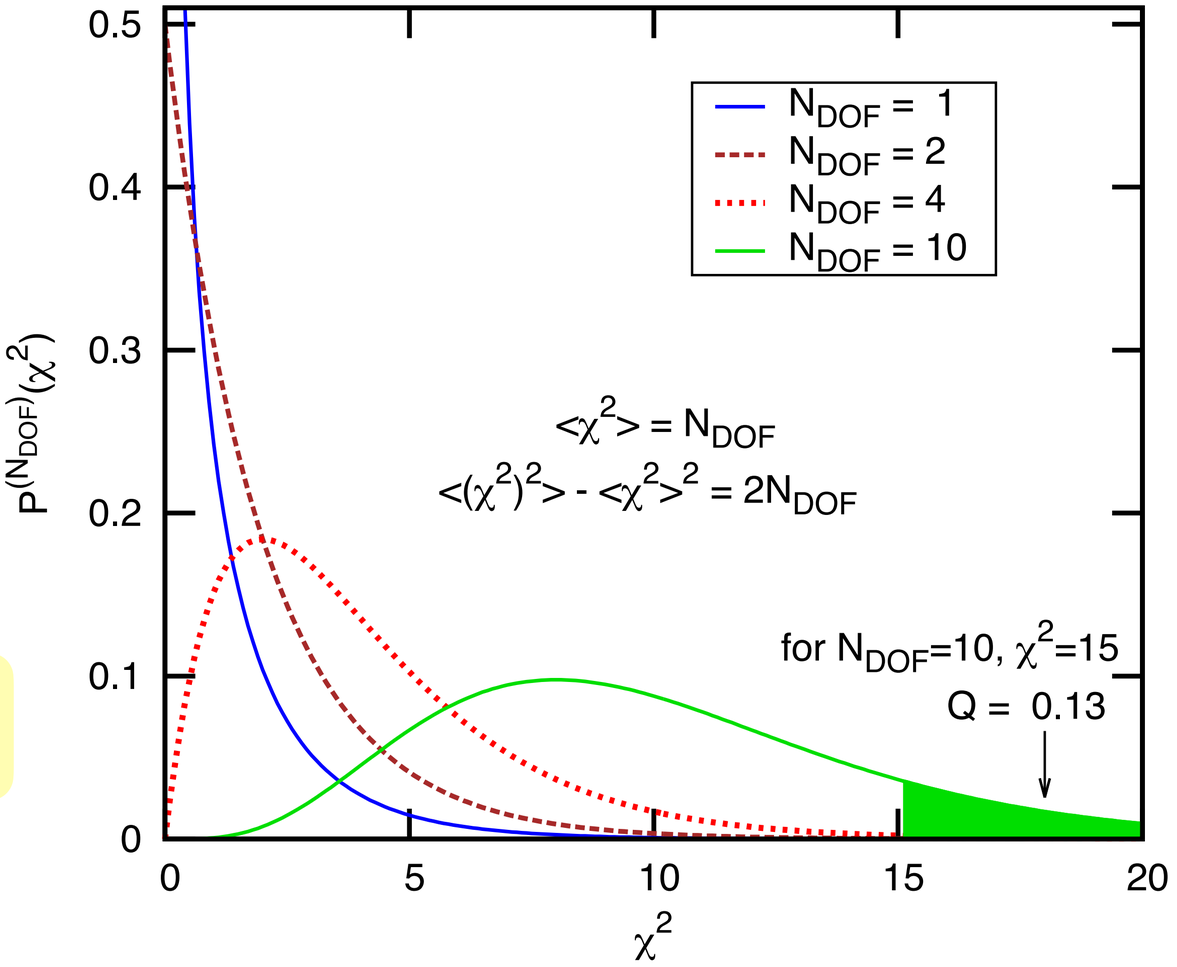
Goodness-of-Fit

- **Reduced Chi-Squared Statistic**

$$\frac{\chi^2}{N_{\mathrm{DOF}}} \sim 1$$

- **P-value:** $Q(\chi^2) = P(X > \chi^2)$

$$Q(\chi^2) = \frac{1}{\Gamma(N_{\mathrm{DOF}}/2)} \int_{\frac{\chi^2}{2}}^{\infty} y^{\frac{N_{\mathrm{DOF}}}{2}-1} e^{-y} dy \sim 0.5$$



Asymptotic standard error

- What if you do not have any error bars available on your data points?
- Still, the chi-squared can make error estimates on the fit parameters.
- Assumption 1. All data points have ***equal*** error bars. i.e. $\sigma_i = \sigma_{\text{ass}}$.
- Assumption 2. Your data points fit ***reasonably*** well to your model.

$$1 = \frac{\chi^2_{\text{ass}}}{N_{\text{DOF}}} = \frac{1}{N_{\text{DOF}}} \sum_{i=1}^{N} \left(\frac{y_i - f(x_i)}{\sigma_{\text{ass}}} \right)^2$$

Asymptotic standard error

How to get the assumed error :

$$\sigma_{\text{ass}}^2 = \frac{1}{N_{\text{DOF}}} \sum_{i=1}^{N} \left(y_i - f(x_i) \right)^2 \quad \longleftarrow \quad 1 = \frac{\chi_{\text{ass}}^2}{N_{\text{DOF}}} = \frac{1}{N_{\text{DOF}}} \sum_{i=1}^{N} \left(\frac{y_i - f(x_i)}{\sigma_{\text{ass}}} \right)^2$$

Just do the least-square fitting with $\sigma_i = 1$.

$$\chi^2 = \sum_{i=1}^{N} \left(y_i - f(x_i) \right)^2 \xrightarrow{\text{Minimize!}} \sigma_{\text{ass}}^2 = \frac{\chi_{\text{min}}^2}{N_{\text{DOF}}}$$

Asymptotic Standard Error

$$[\mathbf{U}^{-1}]_{\alpha\alpha} \longrightarrow \sigma_\alpha^2 = [\mathbf{U}^{-1}]_{\alpha\alpha} \, \sigma_{\text{ass}}^2$$

Coming back to Gnuplot …

```
gnuplot> f(x) = a + b * x
gnuplot> fit f(x) "line.data" using 1:2:3 yerrors via a, b
```

…

!!!

```
degrees of freedom    (FIT_NDF)                        : 18
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.967717
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.936476
p-value of the Chisq distribution (FIT_P)              : 0.532987

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a                  = 0.838641      +/- 0.2066       (24.63%)
b                  = 2.09749       +/- 0.07469      (3.561%)

correlation matrix of the fit parameters:
                a      b
a               1.000
b              -0.856  1.000
```

Different from the previous ones!

0.2134

0.0772

Which ones are the right ones?

Coming back to Gnuplot …

```
degrees of freedom    (FIT_NDF)                        : 18
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.967717
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.936476
p-value of the Chisq distribution (FIT_P)              : 0.532987

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a                  = 0.838641      +/- 0.2066       (24.63%)
b                  = 2.09749       +/- 0.07469      (3.561%)
```

Asymptotic Standard Error

$$\sigma_\alpha^2 = [\mathbf{U}^{-1}]_{\alpha\alpha}\,\sigma_{\text{ass}}^2$$

Gnuplot computes it with weights properly.

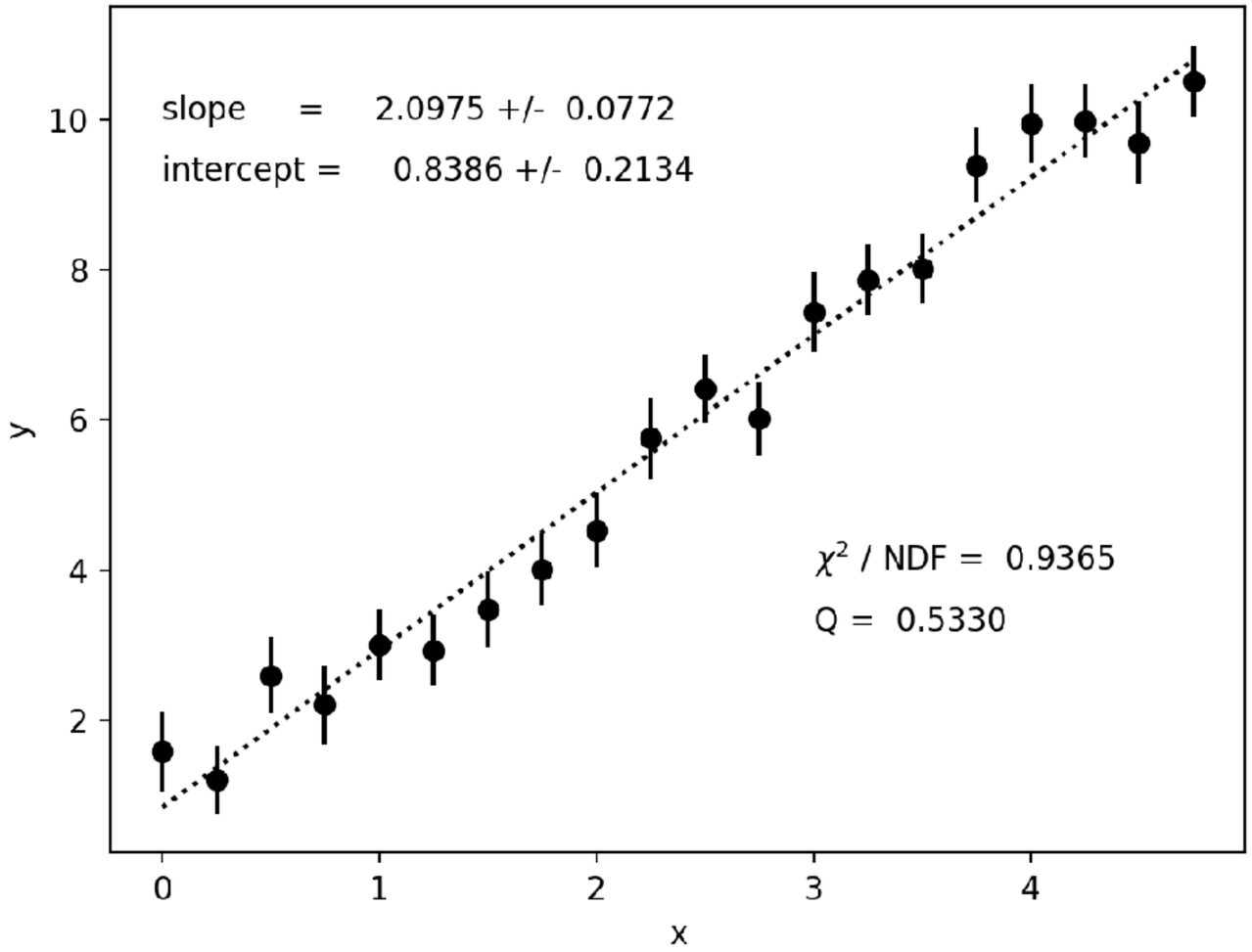
Then, it shouldn't appear.

Correct error bars :

0.2134 (scipy) = 0.2066 / FIT_STDFIT

0.0772 (scipy) = 0.07469 / FIT_STDFIT

Caution with scipy's curve_fit()




```
import numpy as np
from scipy.optimize import curve_fit
from scipy.special import gammaincc

func = lambda x, a0, a1 : a0 + a1 * x

x, y, err = np.loadtxt('line.data',
                       unpack = True)

NDF = x.size - 2

popt, pcov = curve_fit(func, x, y,
                       sigma = err,
                       absolute_sigma = True)

perr = np.sqrt(np.diag(pcov))

chi2 = np.sum((y - func(x,*popt))**2 / err**2)
Q = gammaincc(0.5 * NDF, 0.5 * chi2)
```

Caution with scipy's curve_fit()

```
def curve_fit(f, xdata, ydata, p0=None, sigma=None, absolute_sigma=False,
              check_finite=True, bounds=(-np.inf, np.inf), method=None,
              jac=None, **kwargs):
```

!!!!!!!

```
absolute_sigma : bool, optional
    If True, `sigma` is used in an absolute sense and the estimated parameter
    covariance `pcov` reflects these absolute values.

    If False (default), only the relative magnitudes of the `sigma` values matter.
    The returned parameter covariance matrix `pcov` is based on scaling
    `sigma` by a constant factor. This constant is set by demanding that the
    reduced `chisq` for the optimal parameters `popt` when using the
    *scaled* `sigma` equals unity. In other words, `sigma` is scaled to
    match the sample variance of the residuals after the fit. Default is False.
    Mathematically,
    ``pcov(absolute_sigma=False) = pcov(absolute_sigma=True) * chisq(popt)/(M-N)``
```

Set "sigma" with
"absolute_sigma = True" option

Generalized linear model

Linear model does not necessarily mean a straight line.

$$f(x) = \sum_{\alpha=1}^{M} a_\alpha X_\alpha(x)$$ *with basis functions* $X_1(x), X_2(x), \ldots, X_M(x)$ *(ex. polynomial)*

↓ *Minimize* $$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - \sum_{\alpha=1}^{M} a_\alpha X_\alpha(x_i)}{\sigma_i} \right)^2$$

The same linear equation to solve

$$\mathbf{Ua} = \mathbf{v} \qquad U_{\alpha\beta} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} X_\alpha(x_i) X_\beta(x_i) \qquad v_\alpha = \sum_{i=1}^{N} \frac{y_i}{\sigma_i^2} X_\alpha(x_i)$$

Power-law fitting

It can be converted into a *linear* model.

$$y = a_0 x^{a_1} \longrightarrow \log y = \log a_0 + a_1 \log x \longrightarrow Y = a_0' + a_1 X$$

What about the uncertainties? They are not exactly Gaussian.

→ **It does not matter if the uncertainties are *sufficiently small*.**

$$\log(y+\sigma) = \log\left[y\left(1+\frac{\sigma}{y}\right)\right] \simeq \log y + \frac{\sigma}{y} \quad \text{when} \quad \frac{\sigma}{y} \ll 1$$

Nonlinear model

- One can still try the least square fitting with a general function.
- One can still write an equation like $\mathbf{Ua} = \mathbf{v}$, however the curvature matrix $\mathbf{U}$ now depends on the data sample $\mathbf{y}$. In the linear model, only $\mathbf{v}$ depends on $\mathbf{y}$.
- One still minimize χ^2 numerically while $\mathbf{Ua} = \mathbf{v}$ is a nonlinear equation.
- The quantity $(\sigma_\alpha^S)^2 = \left[\mathrm{U}^{-1}\right]_{\alpha\alpha}$ can be evaluated at minimum χ^2, but its validity as an error estimate depends on the spread of fitted parameters.

Confidence limit

- It estimates the error bar of a fitted parameter by setting a criterion for $\Delta\chi^2 \equiv \chi^2 - \chi^2_{\min}$, the deviation from minimum χ^2.

- Joint probability distribution of fit parameters $\mathbf{a}^S = (a_1^S, a_2^S, \ldots, a_M^2)$

$$P(\mathbf{a}^S) \propto \exp\left(-\frac{1}{2}\sum_{\alpha,\beta}\delta a_\alpha^S U_{\alpha\beta}\delta a_\beta^S\right) = \exp\left(-\frac{1}{2}\Delta\chi^2\right)$$

$$\chi^2 \approx \chi^2_{\min} + \frac{1}{2}\frac{\partial^2\chi^2}{\partial a_\alpha \partial a_\beta}\bigg|_{\min}\delta a_\alpha^S \delta a_\beta^S \quad \leftarrow \quad \frac{1}{2}\frac{\partial^2\chi^2}{\partial a_\alpha \partial a_\beta}\bigg|_{\min} = \sum_{i=1}^{N}\frac{1}{\sigma_i^2}X_\alpha(x_i)X_\beta(x_i) = U_{\alpha\beta}$$

(effective linear model)

Confidence limit

$$P(\mathbf{a}^S) \propto \exp\left(-\frac{1}{2}\Delta\chi^2\right)$$

Does $\Delta\chi^2 = 1$ mean something particular?

So-call "one sigma", 68% confidence limit.

(c.f "two sigma" [$\Delta\chi^2 = 4$] : 95% confidence limit)

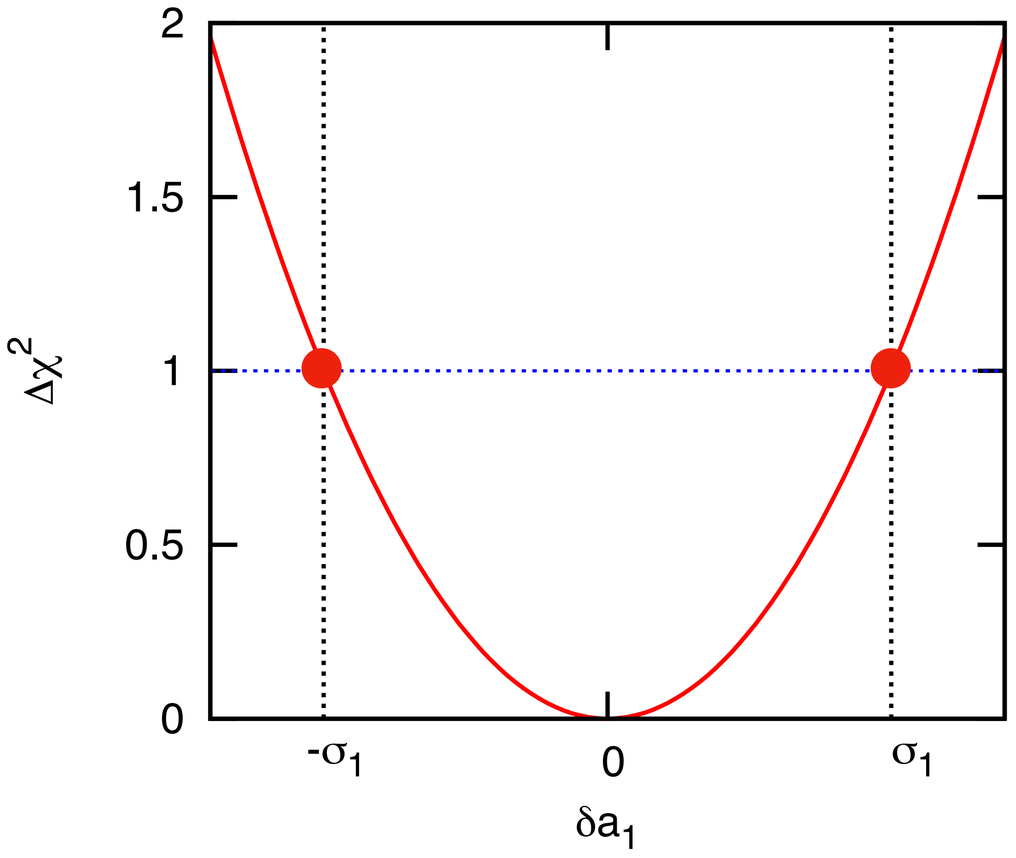
$$\Delta\chi^2 = 1 \quad \xleftrightarrow{\text{Linear model}} \quad (\sigma_\alpha^S)^2 = \left[\mathrm{U}^{-1}\right]_{\alpha\alpha}$$

Compute $P(a_1^S)$: fix a_1^S at a given value, adjust the others to minimize χ^2 .

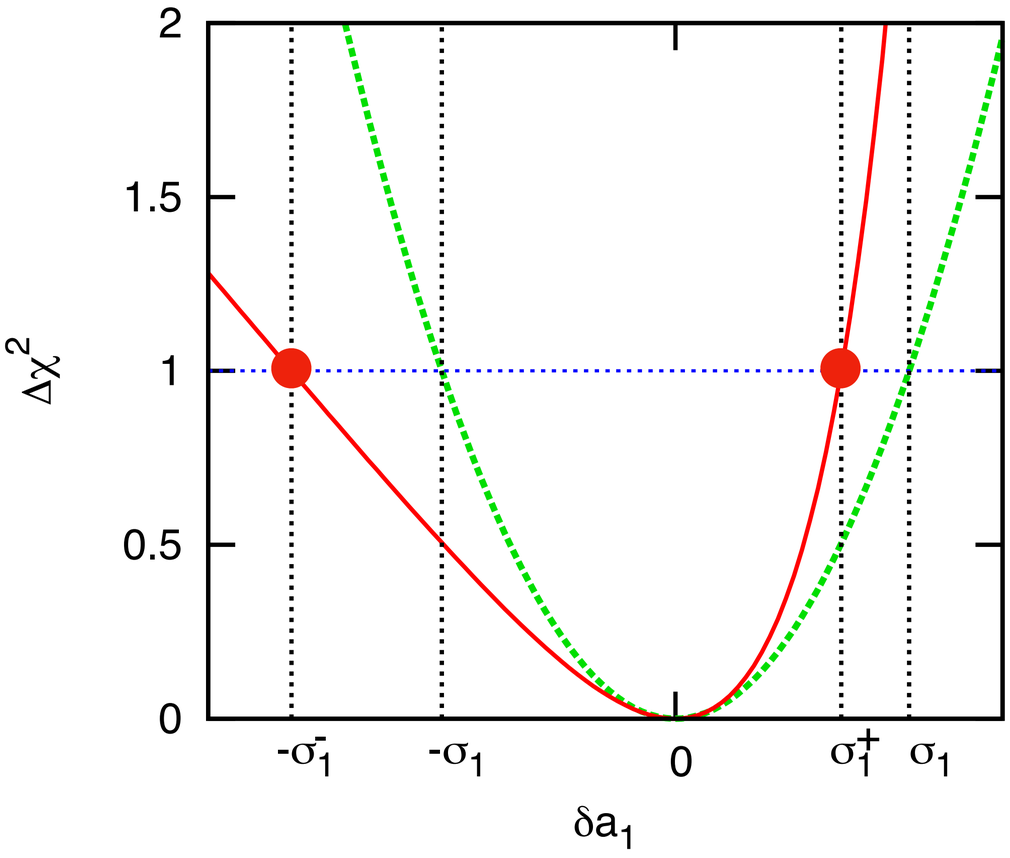
$$\frac{\partial\chi^2}{\partial a_\alpha} = 0 \quad (\alpha \neq 1) \longrightarrow \sum_{\beta=1}^{M} U_{\alpha\beta}\delta a_\beta^S = c\delta_{\alpha,1} \longrightarrow c = \frac{\delta a_1^S}{[\mathrm{U}^{-1}]_{11}} \longrightarrow P(a_1^S) \propto \exp\left(-\frac{(\delta a_1^S)^2}{2[\mathrm{U}^{-1}]_{11}}\right)$$

Confidence limit

Linear model



Nonlinear model (asymmetric error bars?)



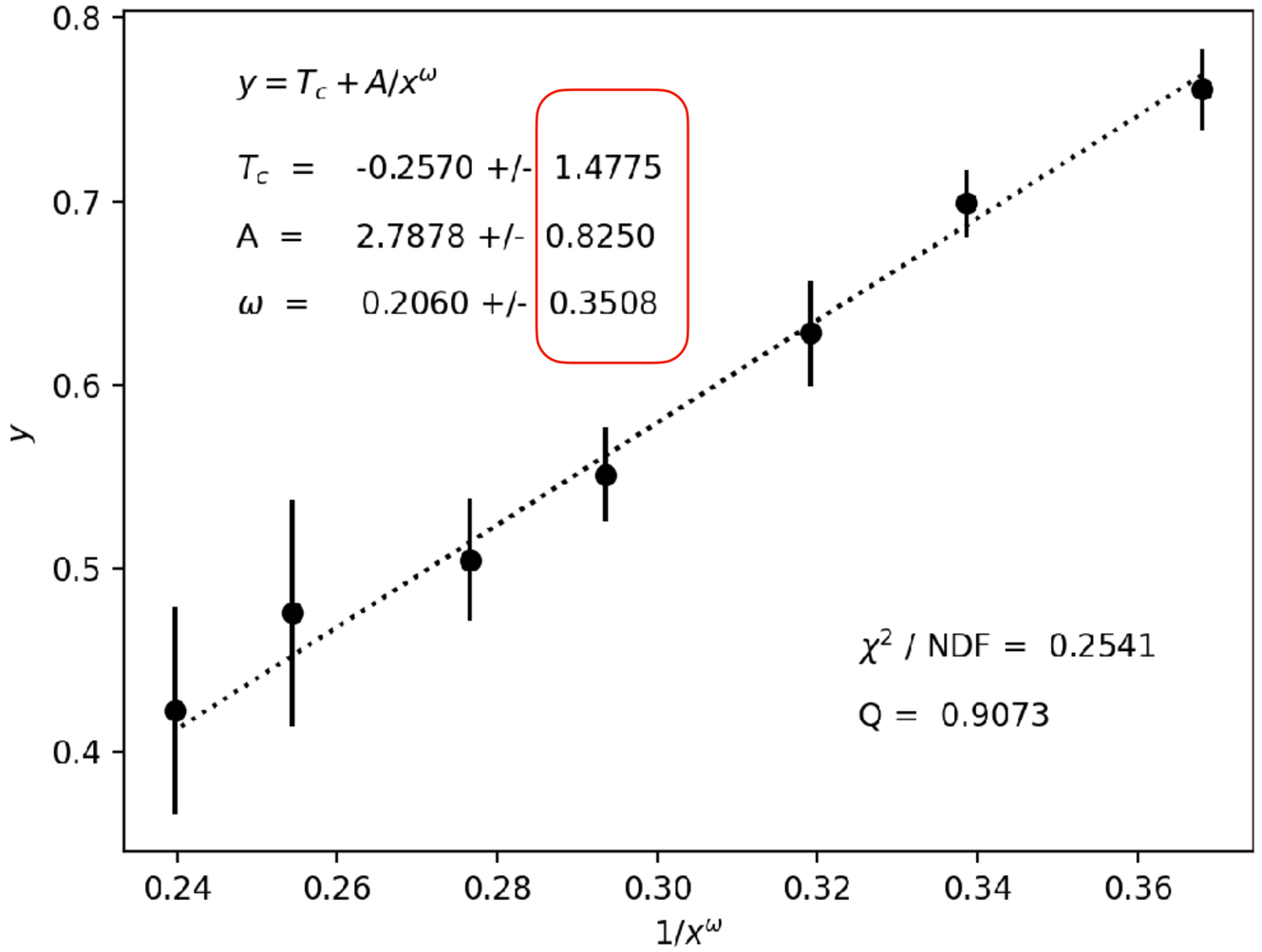
Example: $y = T_c + A / x^w$

x	y	error
128	0.760881	0.022157
192	0.698935	0.018346
256	0.628069	0.028605
384	0.550988	0.025836
512	0.504440	0.033009
768	0.475425	0.061739
1024	0.422427	0.056518

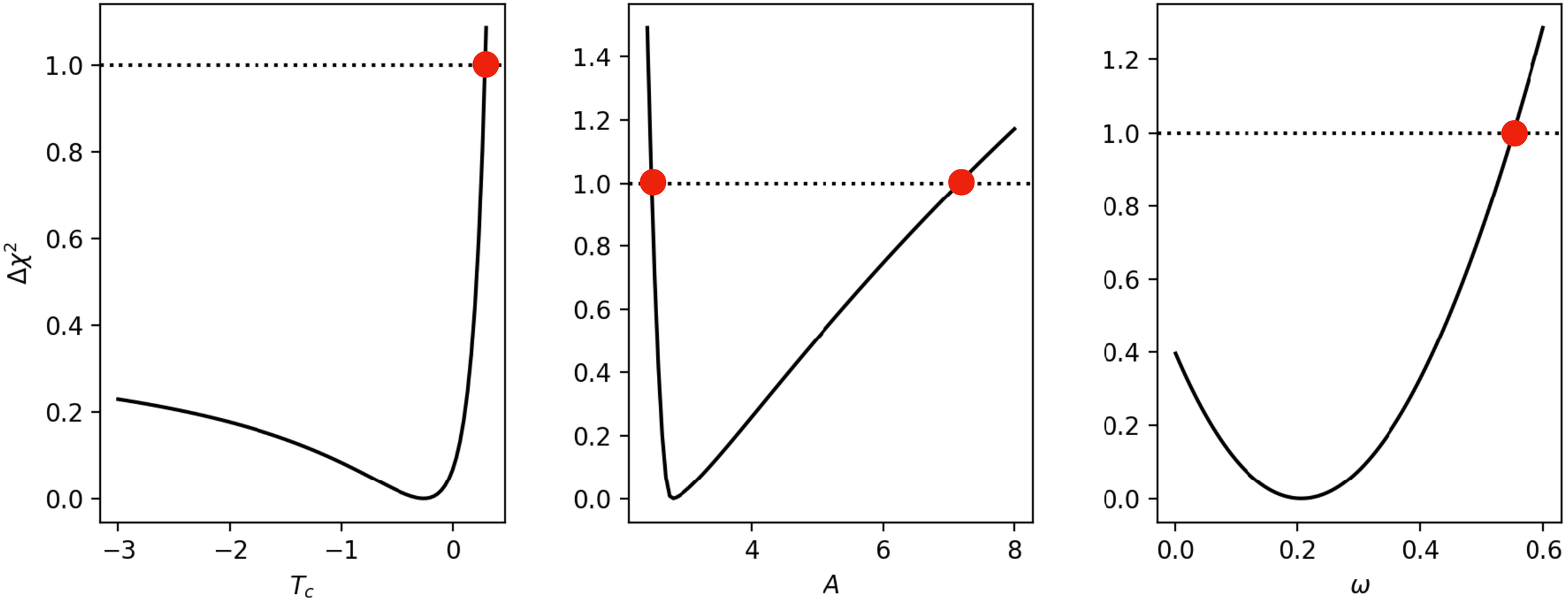
Least square fit with a nonlinear model

$$y = T_c + A/x^{\omega}$$

Example: error bars with an effective linear model



Example: highly asymmetric error bars



Confidence limit by resampling the data

- You have access to a raw data set, which may exhibit a *non-Gaussian* distribution, producing a data point (x_i, y_i).
- If it is the case, "*bootstrap resampling*" gives you *better* error estimates.
- Make many "bootstrap" data sets. Find the fit parameters for each of them to get statistics. Plot the histogram, and find the confidence interval.
- Though you do not have access to raw data, you can still make *a synthetic (simulated) raw data,* assumed that the noises are Gaussian.

Bootstrap resampling

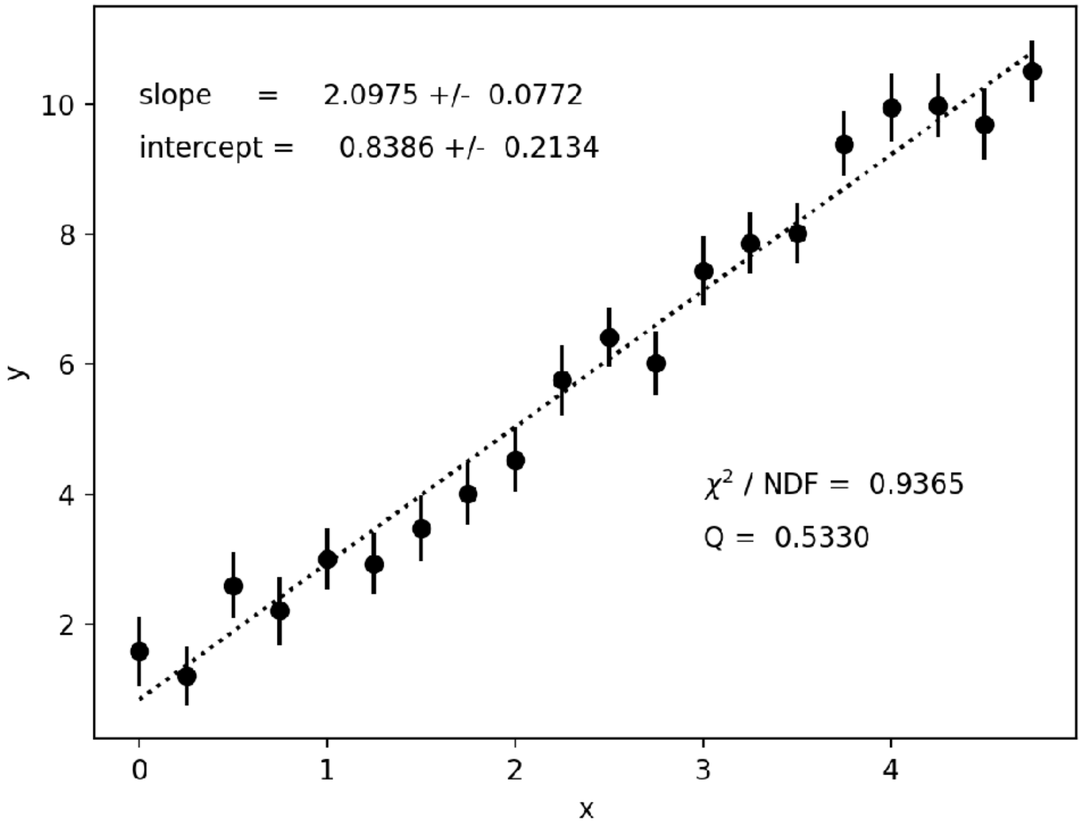
- For a raw data set $\{y_1^{(i)}, y_2^{(i)}, \ldots, y_{N_i}^{(i)}\}$ that produces each data point $y_i = \bar{y}_j^{(i)}$, one can generate a "bootstrap" data set by randomly picking up N_i element from the raw data set with replacement.

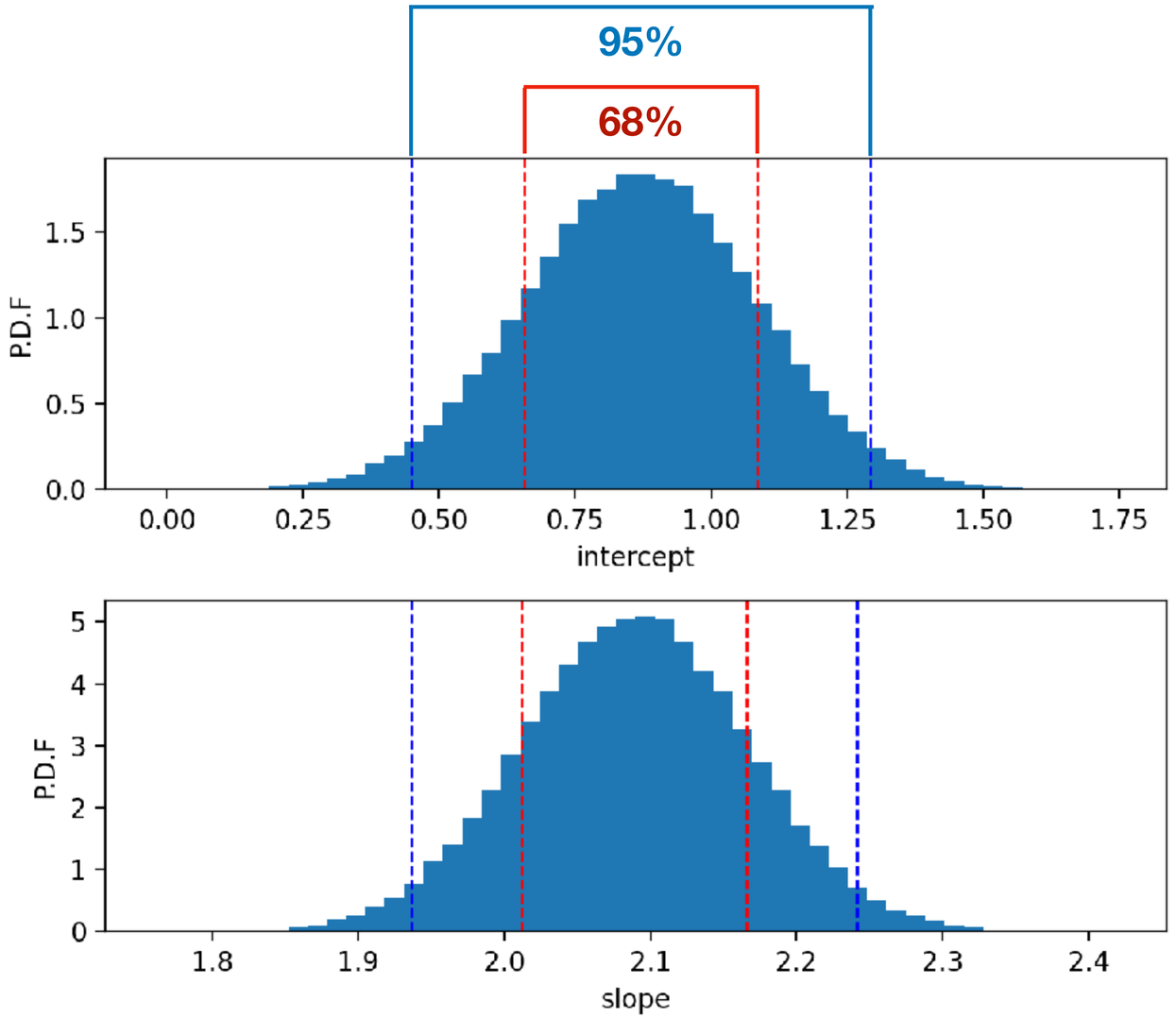
- Instead of one set of the original data points $\mathbf{y} = (y_1, y_2, \ldots, y_N)$, one can consider a "bootstrap" data points $\mathbf{y}^* = (y_1^*, y_2^*, \ldots, y_N^*)$ to make the least square fit to determine the fitting parameters $\mathbf{a}^* = (a_0^*, a_1^*, \ldots, a_{M-1}^*)$.

- Generating many such "bootstrap" data points makes many different sets of the fitting parameters, allowing us to draw a probability distribution of $\mathbf{a}^* = (a_0^*, a_1^*, \ldots, a_{M-1}^*)$.

- Choose the confidence interval, typically 95% (two-sigma), 68% (one-sigma), …

Confidence interval





Summary

- Least-square fitting works with **Gaussian error bars** and **linear models**.
- Gnuplot: be careful with “asymptotic standard error.”
- Python: make sure to choose the right “absolute_sigma” option.
- It may work for non-Gaussian error bars and nonlinear models, but use it with caution for error estimates of fitting parameters.
- You may consider the bootstrap resampling if you have raw data sets.

Monte Carlo Simulations

in “equilibrium” statistical physics

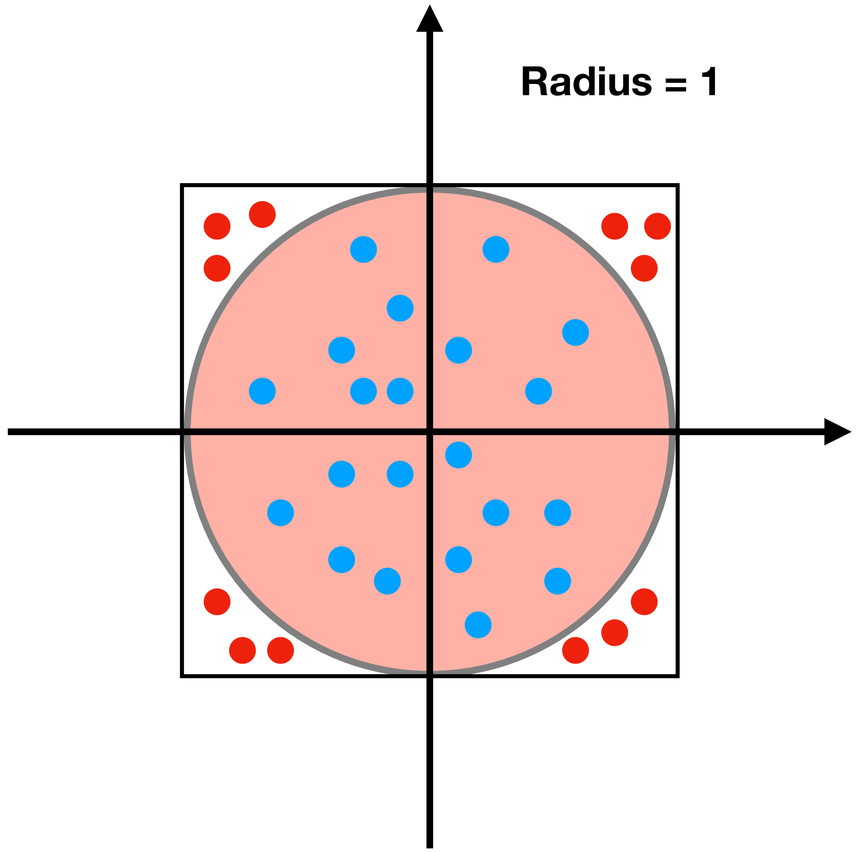
Dong-Hee Kim

Gwangju Institute of Science and Technology

References

- M. E. J. Newman and G. T. Barkema, "Monte Carlo Methods in Statistical Physics" (Oxford University Press, 1999)
- L. Böttcher and H. J. Herrmann, "Computational Statistical Physics" (Cambridge University Press, 2021)
- W. Janke, in "Computational Physics" (Springer, 1996)
- Lectures by Mattias Troyer, Bernd Berg, Kari Rummukainnen

Integration by rolling dice



Compute the area of the unit circle with random numbers.

1. Generate *two* random numbers from the uniform distribution between -1 and 1.
2. Use them as x and y coordinates.
3. See if (x,y) is inside of the circle.
4. Generate many such random points, and count how many are inside of the circle.
5. **Area** = **4** x **number of points found in the circle / total number of points**

Pseudo-random numbers

- “**P**seudo”-**R**andom **N**umber **G**enerators. That is what we’re looking for.
- There are many algorithms and implementations.
- There are notorious ones causing *systematic failures* in Monte Carlo simulations, such as **R250** and **RANDU**. (e.g. Ferrenberg *et al.* PRL 1992) See “examples of a *bad* generator” in H. G. Katzgraber, arXiv:1005.4117.
- There are many tests for “quality” check, but those do not guarantee that it is random enough. **R250** was a “good” PRNG as it passed all common quality tests at that time.

PRNG

- It is deterministic. It generates “a sequence” based on a given “seed”. The sequence is generated recursively, i.e. $x_i = f(x_{i-1}, x_{i-2}, \ldots, x_{i-k})$.
- It has a finite period.
- A simple example: a linear congruential generator, RAN0.

$$X_{i+1} = (16807\, X_i) \mod (2^{31} - 1)$$

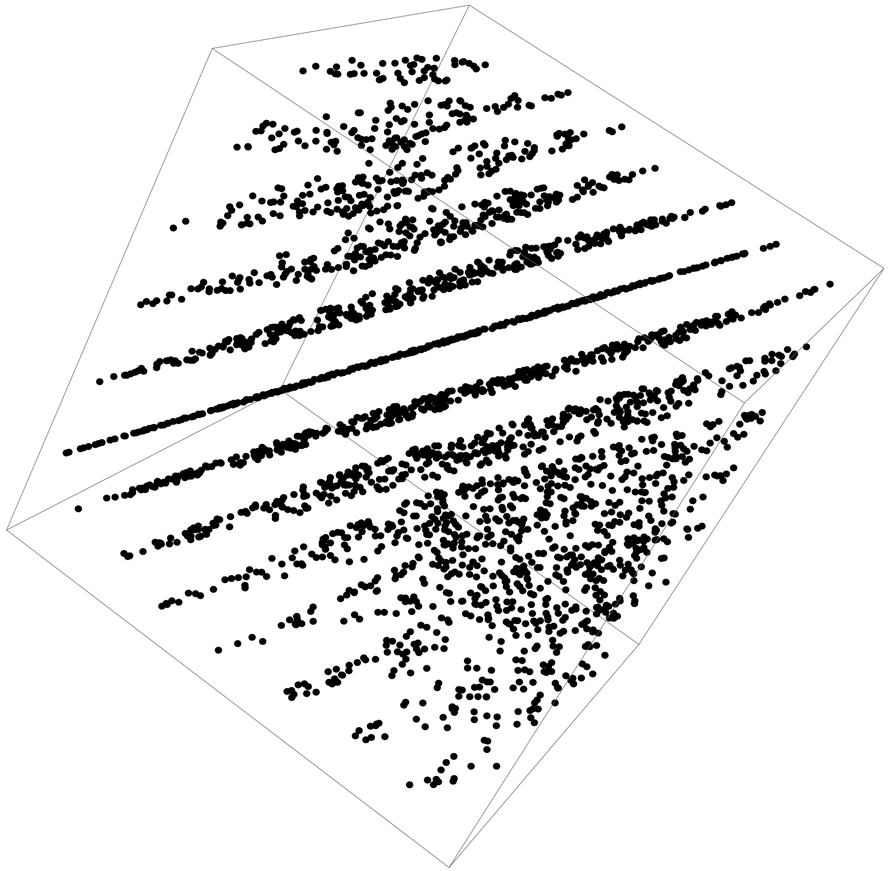
(period = 2^{31} - 1: very short)

- Good PRNG: good statistics (no correlation), long period, high performance, well-tested in your field of research.

A bad PRNG?

[From Helmut G. Katzgraber, arXiv:1005.4117]

Figure 1: 10^3 triplets of successive random numbers produced with RANDU plotted in three-dimensional space. If the random numbers were perfectly random, no planes should be visible. However, when viewed from the right angle, planes emerge, thus showing that the random numbers are strongly correlated.



Strategies for parallel computing

- Random seeding: each CPU has its own seed. *Hope for the best.*
- **Parametrization**: each CPU has different parameters for the same PRNG.
- **Block splitting**:

- **Leapfrogging**:

Popular PRNG

- **Merssene Twister** generator (MT19937, Matsumoto and Nishimura 1998)
  - Serial generator, available by default in C++11, python, etc.
  - Very fast, Very long period (2^{19937}-1), Very low correlation
- **SPRNG** (The Scalable Parallel Random Number Generators Library)
  - Parallel (parameterization), available at http://www.sprng.org/
- **TRNG** (Tina's random number generator library, fully C++)
  - Parallel (block splitting, leapfrogging), https://www.numbercrunch.de/trng/

C++ vs. Python

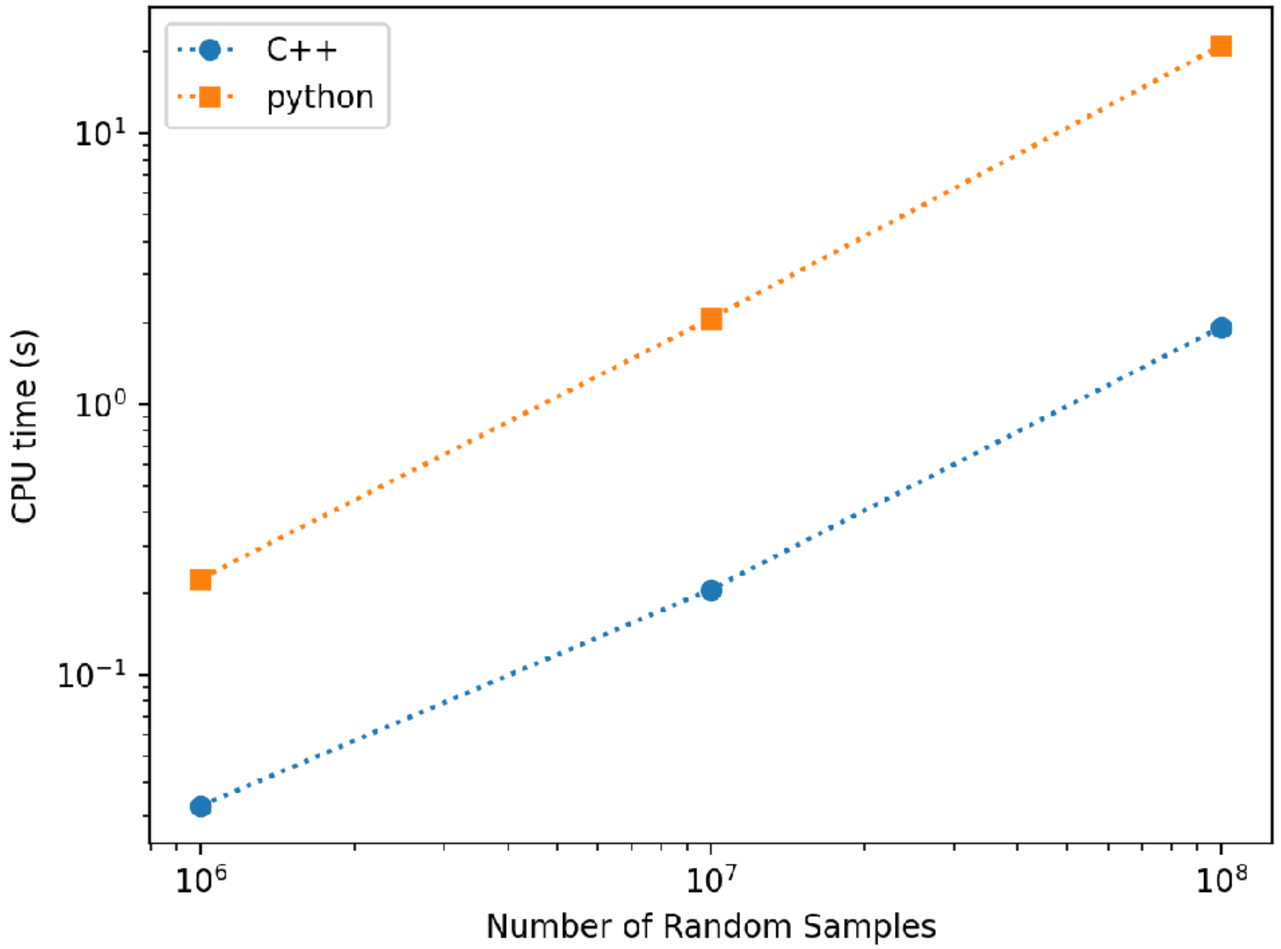
Computing π with random numbers:

```cpp
#include <iostream>
#include <random>

int main(int argc, char* argv[]) {

    const int N = atoi(argv[1]);

    std::random_device rd;
    std::mt19937 mt(rd());
    std::uniform_real_distribution<double> rng;

    int count = 0;
    for (int i = 0; i < N; ++i) {
        const double x = rng(mt);
        const double y = rng(mt);
        if (x*x + y*y < 1.0) count++;
    }

    std::cout << "AREA = "
              << 4.0 * count / N << std::endl;
}
```

```python
import random as rng
import sys

N = int(sys.argv[1])

count = 0

for i in range(N):
    x = rng.random()
    y = rng.random()
    if x**2 + y**2 < 1 :
        count = count + 1

print('AREA = %f' % (4.0*count/N))
```

```
>> ./pi.x [N]  # C++

>> python3 pi.py [N]  # python
```

C++ vs. Python



Computing π with random numbers:

C++ is 10 times faster than python.

Python's for-loop has a HUGE overhead.

Do not use python for serious Monte Carlo simulations.

Non-uniform distribution

- So far, we have been playing with a random number from a uniform distribution [0,1).
- What if you need a random number following an arbitrary distribution?
- Transformation method. e.g. Box-Muller transformation
- Rejection sampling method. (purely numerical)

Transformation

Change of variables in P.D.F

$$|p_x(x)\,dx| = |p_y(y)\,dy|$$

e.g. exponential distribution

$$p(x)\,dx = a\exp[-ax]\,dx$$
$$= d[e^{-ax}] = 1 \cdot du$$

uniform distribution

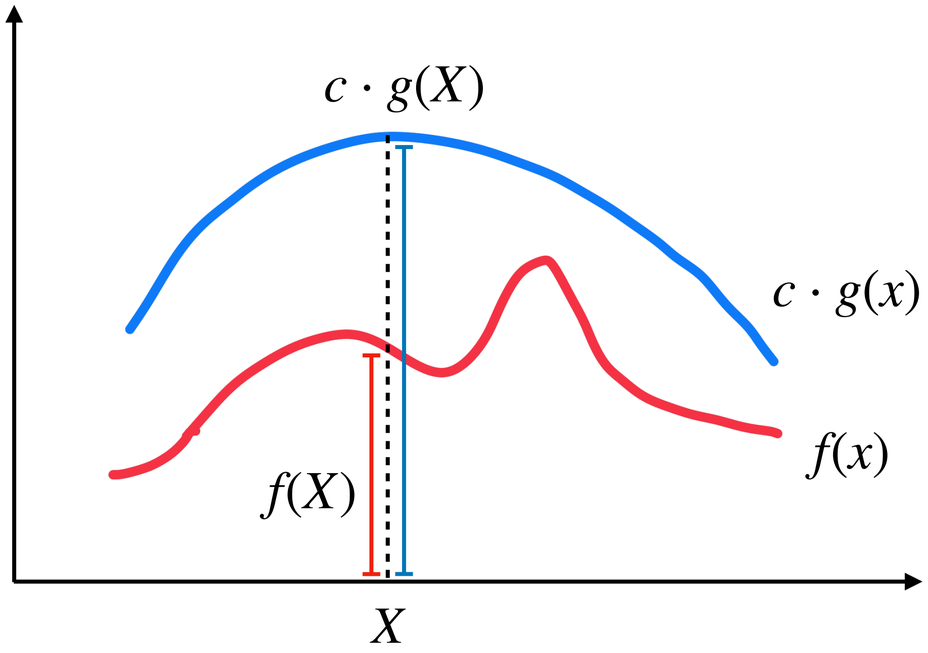
$$\rightarrow \quad x = -\frac{1}{a}\ln(1-u)$$

e.g. Box-Muller transformation for the Gaussian distribution

$$P(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right) \quad \rightarrow \quad x = \sqrt{-2\ln(1-u)}\sin(2\pi v)$$

It does not work for all probability distribution! — This method requires an inverse function.

Rejection sampling



Conditions

1. *We know how to make a random sample from **g(x)**.*
2. *There exists a constant **c** such that **f(x) < c g(x)**.*

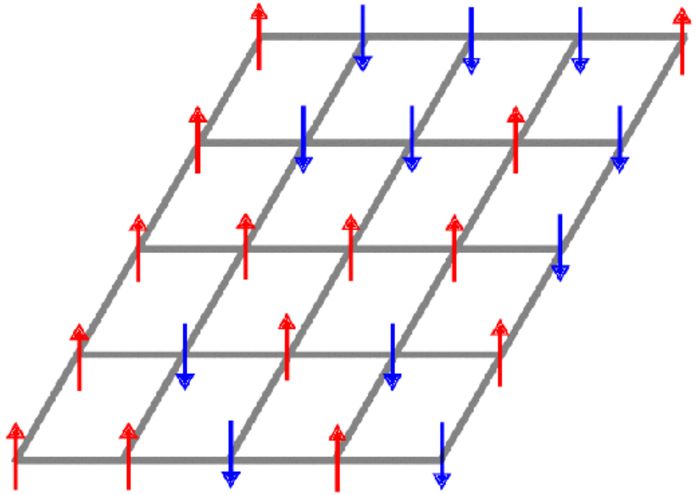
Algorithm

1. Draw a random number ***X*** from PDF ***g(x)***.
2. Accept X with a probability ***f(X) / [c g(X)]***;
   (1) Draw a uniform random number ***r***
   (2) If ***r < f(X) / [c g(X)]***, accept ***X***.
   (3) Reject, otherwise.

Ising model in two dimensions

Hamiltonian

$$E = -J \sum_{\langle i,j \rangle} s_i s_j - h \sum_i s_i \qquad h = 0$$



Canonical ensemble

$$p(\{s_1, s_2, \ldots, s_N\}) = \frac{1}{Z} \exp[-\beta E(\{s_1, s_2, \ldots, s_N\})]$$

Measurement

$$\langle O \rangle = \sum_{s_1=\pm 1} \sum_{s_2=\pm 1} \cdots \sum_{s_N=\pm 1} O(\{s_1, s_2, \ldots, s_N\})\, p(\{s_1, s_2, \ldots, s_N\}) \rightarrow \langle O \rangle_{\mathrm{MC}}$$

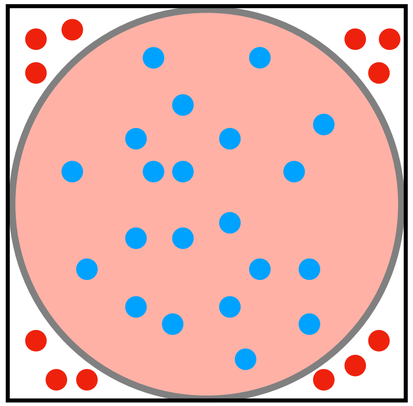
What are the hurdles?

Compute this: $$\langle O \rangle = \sum_{s_1=\pm 1} \sum_{s_2=\pm 1} \cdots \sum_{s_N=\pm 1} O(\{s_1, s_2, \ldots, s_N\})\, p(\{s_1, s_2, \ldots, s_N\})$$

where $$p(\{s_1, s_2, \ldots, s_N\}) = \frac{1}{Z} \exp[-\beta E(\{s_1, s_2, \ldots, s_N\})]$$

Number of microstates = $\mathbf{2^N}$: IT'S IMPOSSIBLY LARGE!

What about just generating a random spin sequence and computing a sum like we did before:



Importance sampling

- Uniform generation of random spin sequence would not solve the issue:
  The uniform sampling does not change the system size scaling $\sim 2^N$.

- Boltzmann weight is exponential: central region contributes much more than the tail part. Can we just sample the Boltzmann distribution directly?

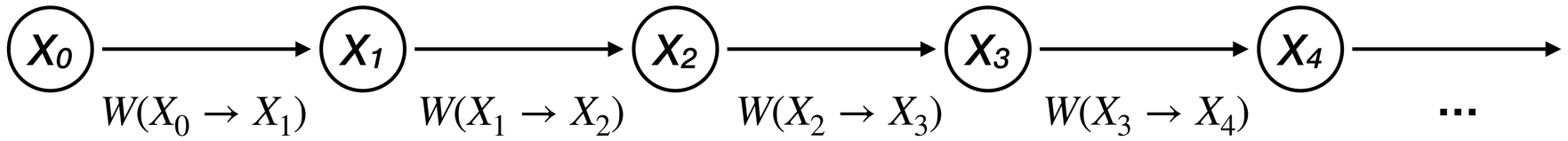
- **Importance sampling** (this is what “Monte Carlo” really means in practice)
  — a strategy to visit “important” states more often.

e.g. importance sampling of the canonical ensemble

$$\langle O \rangle = \frac{1}{Z} \sum_{\{s_i\}} O \exp(-\beta E) \approx \frac{1}{N_{\mathrm{MC}}} \sum_{i=1}^{N_{\mathrm{MC}}} O_i \equiv \langle O \rangle_{\mathrm{MC}}$$

Markov Chain Monte Carlo

Start from a state X and propose a new state Y with a **selection probability** $g(X \to Y)$.
Accept or reject the proposed state with an **acceptance probability** $A(X \to Y)$.



Transition probability $W(X \to Y) = g(X \to Y)\, A(X \to Y)$

$$\langle O \rangle_{\mathrm{MC}} = \frac{1}{N_{\mathrm{MC}}} \left[O(X_{t_o}) + O(X_{t_o+1}) + O(X_{t_o+2}) + O(X_{t_o+3}) + \cdots + O(X_{t_o+N_{\mathrm{MC}}}) \right]$$

Markov Chain Monte Carlo

- Master equation

$$\frac{dp(X,t)}{d\tau} = \sum_{Y \neq X} p(Y)W(Y \to X) - \sum_{Y \neq X} p(X)W(X \to Y) \xrightarrow{\text{Equilibrium}} 0$$

$$p(X) = p^{\text{eq}}(X)$$

equilibrium distribution

$$p^{\text{eq}}(X) \propto \exp[-\beta E(X)]$$

- Ergodicity: $\forall X, Y \in \mathcal{S},\ W(X \to Y) > 0.$
- Normalization: $\sum_Y W(X \to Y) = 1$
- Homogeneity: $\sum_Y p(Y)W(Y \to X) = p(X)$

Detailed Balance Condition

Balance Condition [stationary p(X)]

$$\sum_{Y \neq X} p(Y) W(Y \to X) = \sum_{Y \neq X} p(X) W(X \to Y)$$

"Equilibrium"

$$p(X) \to p^{\mathrm{eq}}(X)$$

Detailed Balance Condition (sufficient condition)

$$p^{\mathrm{eq}}(Y) W(Y \to X) = p^{\mathrm{eq}}(X) W(X \to Y)$$

→ **"Reversibility"**

"No net stochastic flux"

$$W(X \to Y)\, W(Y \to Z)\, W(Z \to X)$$
$$= W(X \to Z)\, W(Z \to Y)\, W(Y \to X)$$

c.f. "irreversible" MC

Canonical ensemble

$$\frac{W(X \to Y)}{W(Y \to X)} = \frac{p^{\mathrm{eq}}(Y)}{p^{\mathrm{eq}}(X)} = \exp[-\beta(E_Y - E_X)]$$

Convergence of MCMC

A physicist-friendly proof: O. Narayan and A. P. Young, PRE 64, 021104 (2001)

master equation

$$p_l(t+1) - p_l(t) = \sum_{m \neq l} \left[p_m(t)\, W(m \to l) - p_l(t)\, W(l \to m) \right] \quad \rightarrow \quad p_l(t+1) = \sum_m p_m(t)\, W(m \to l) \quad \leftarrow \quad \sum_m W(l \to m) = 1$$

detailed balance

$$p_l^{\mathrm{eq}}\, W(l \to m) = p_m^{\mathrm{eq}}\, W(m \to l) \quad \rightarrow \quad G(t) = \sum_l \frac{1}{p_l^{\mathrm{eq}}} \left(p_l(t) - p_l^{\mathrm{eq}} \right)^2 = \sum_l \frac{[p_l(t)]^2}{p_l^{\mathrm{eq}}} - 1$$

ergodicity , $W > 0$

$$\Delta G = G(t+1) - G(t) = -\frac{1}{2} \sum_{l,m,n} W(l \to m)\, W(l \to n)\, p_l^{\mathrm{eq}} \left(\frac{p_m(t)}{p_m^{\mathrm{eq}}} - \frac{p_n(t)}{p_n^{\mathrm{eq}}} \right)^2 \quad \leq 0$$

Metropolis algorithm

M(RT)² algorithm [N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller (1953)]

Canonical ensemble

$$\frac{W(X \to Y)}{W(Y \to X)} = \frac{p^{\mathrm{eq}}(Y)}{p^{\mathrm{eq}}(X)} = \exp[-\beta(E_Y - E_X)]$$

You can propose any A that satisfies this equation!

$$\frac{W(X \to Y)}{W(Y \to X)} = \frac{g(X \to Y)\,A(X \to Y)}{g(Y \to X)\,A(Y \to X)} = \frac{A(X \to Y)}{A(Y \to X)} = e^{-\beta(E_Y - E_X)}$$

symmetric

The choice of M(RT)² :

$$A(X \to Y) = \min\left[1, \exp[-\beta(E_Y - E_X)]\right]$$

Choices of Acceptance Probability

You can propose any A that satisfies this equation:

$$\frac{A(X \to Y)}{A(Y \to X)} = e^{-\beta(E_Y - E_X)}$$

M(RT)² :

$$A(X \to Y) = \min\left[1, \exp(-\beta \Delta E)\right]$$

Glauber :

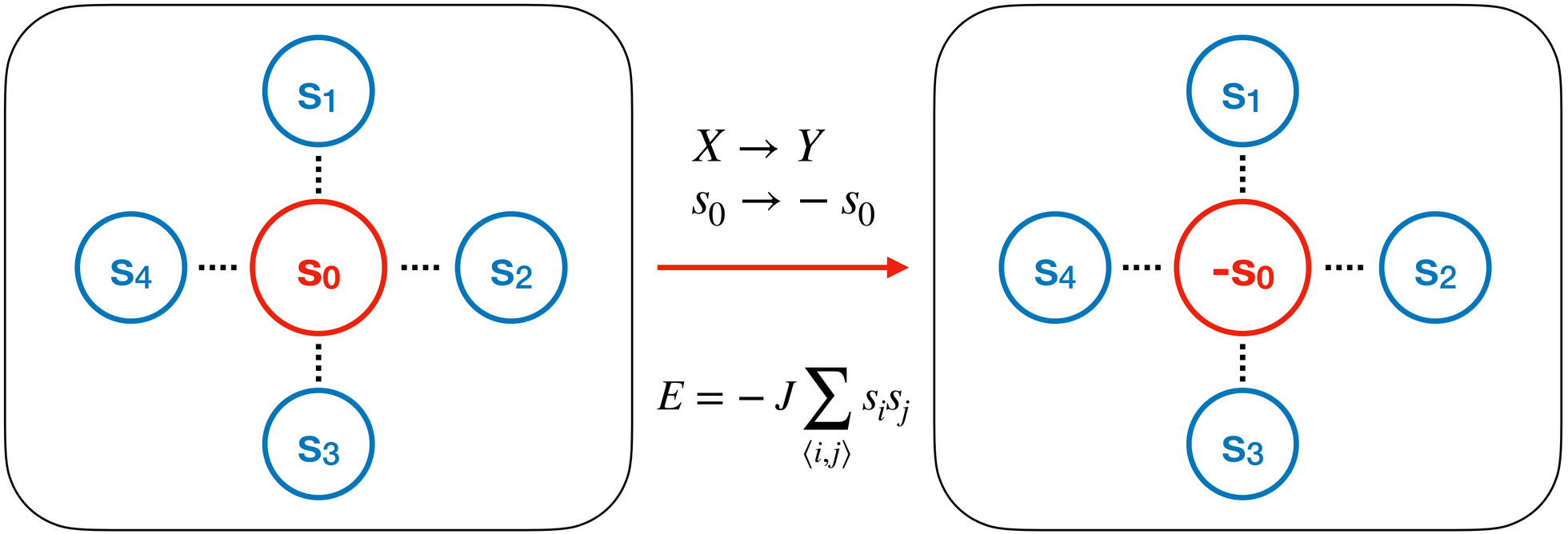
$$A(X \to Y) = \frac{\exp(-\beta \Delta E)}{1 + \exp(-\beta \Delta E)}$$

⟵ ***Heat-bath algorithm for the Ising model***

Simulating Ising model

local update with a single-spin flip

$$\Delta E \equiv E_Y - E_X = [-Js_0(s_1 + s_2 + s_3 + s_4)] - [-J(-s_0)(s_1 + s_2 + s_3 + s_4)] = \boxed{-2Js_0 \sum_{k \in \text{n.n.}} s_k}$$



Common update strategies

How do you <u>sweep</u> the lattices to update spins?

1. RANDOM

Visit lattices sites randomly. Slow.

2. SEQUENTIAL

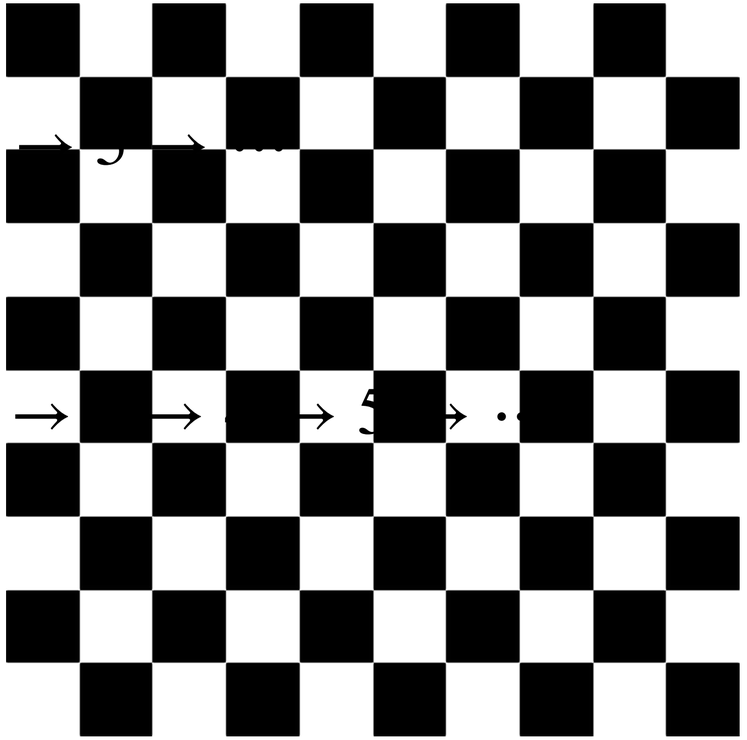
Visit every site sequentially. Weak detailed balance.

3. SUBLATTICE (CHECKERBOARD)

Visit every site in sublattices sequentially.
Weak detailed balance. Some advantages in vectorization.

$3 \to 1 \to 10 \to 14 \to$

$1 \to 2 \to$



One Monte Carlo Step (MCS)

A typical choice of unit time in Monte Carlo simulations

= One full sweep of the lattice sites

```
for mcs in range(N_mcs) :
    for i in range(L) :
        for j in range(L) :
            deltaE = (compute ΔE by s_{i,j} → − s_{i,j})
            if deltaE < 0 :
                (accept the spin flip)
            else if exp(−beta∗deltaE) < random() :
                (accept the spin flip)
```

one move

one sweep, one MCS

Weak Detailed Balance

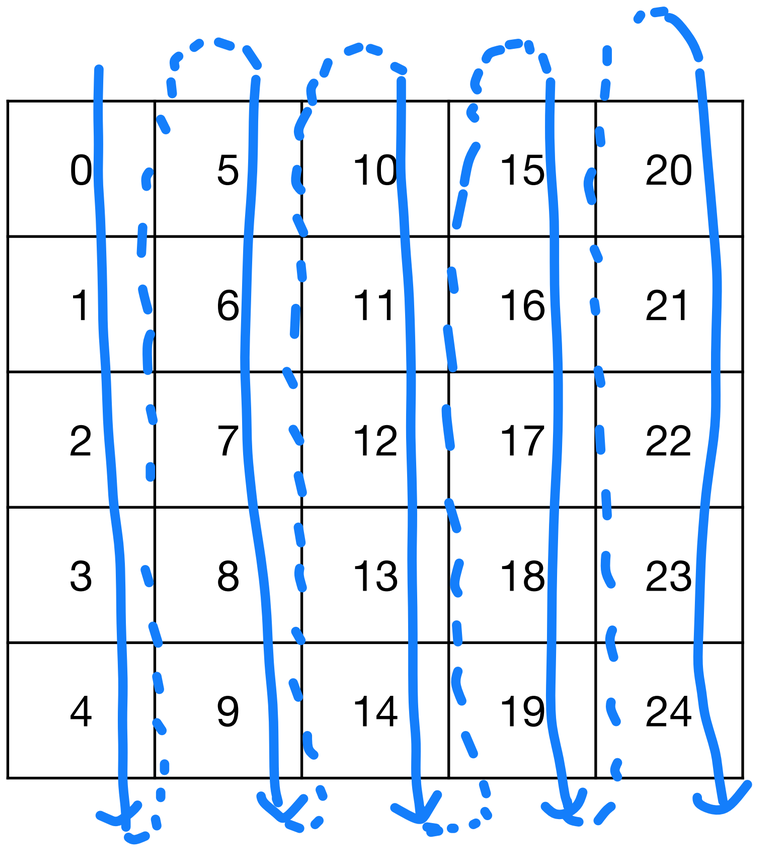
master equation $$p_l(t+1) = \sum_m p_m(t)\, W(m \to l)$$

sequential update

$$\mathbf{p}(t_{\mathrm{mcs}}+1) = \mathbf{W}_{\mathrm{sweep}}\,\mathbf{p}(t_{\mathrm{mcs}}) = \mathbf{W}_N \mathbf{W}_{N-1} \cdots \mathbf{W}_1 \mathbf{W}_0\, \mathbf{p}(t_{\mathrm{mcs}})$$

$\mathbf{W}_i$: detailed balance is valid.

$\mathbf{W}_{\mathrm{sweep}}$: detailed balance is broken.



Thus, the inequality still works in every Monte Carlo "move".

$$\Delta G = G(t+1) - G(t) = -\frac{1}{2} \sum_{l,m,n} W(l \to m)\, W(l \to n)\, p_l^{\mathrm{eq}} \left(\frac{p_m(t)}{p_m^{\mathrm{eq}}} - \frac{p_n(t)}{p_n^{\mathrm{eq}}} \right)^2 \leq 0$$

A typical structure of an MC code

0. Initialization

Ordered initial state or *disordered* (*random*) initial state?

1. Thermalization

```
sweep()
```

Allow *relaxation* time for a spin configuration to be thermalized.

Ideally, we want $p(X) \rightarrow p^{\mathrm{eq}}(X)$.

2. Measurement

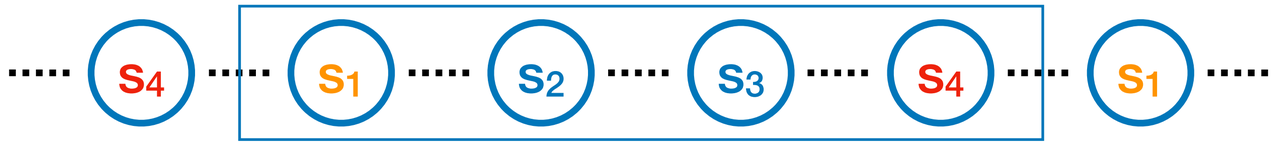
```
sweep()
measure()
```

This is a production run. Measure what you want every ***m*** MCS.

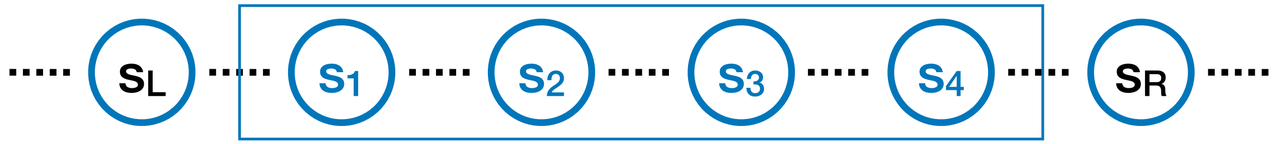
(***m = 1*** is just fine.)

Boundary Conditions

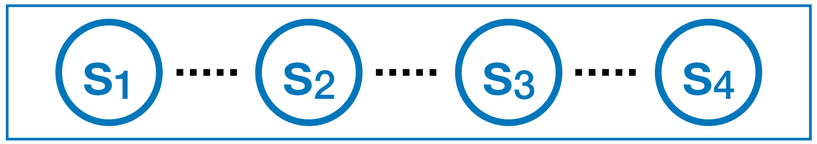
- Periodic Boundary Condition



- Fixed Boundary Condition



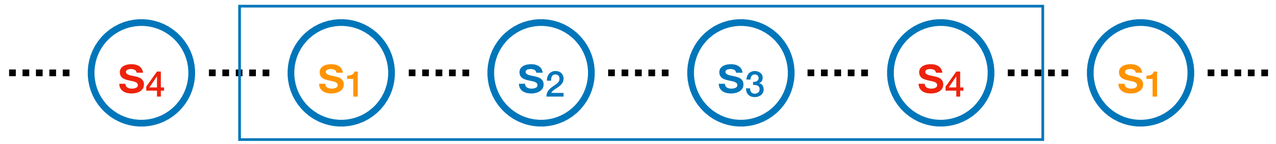
- Open Boundary Condition



- Helical Boundary Condition

Boundary Conditions

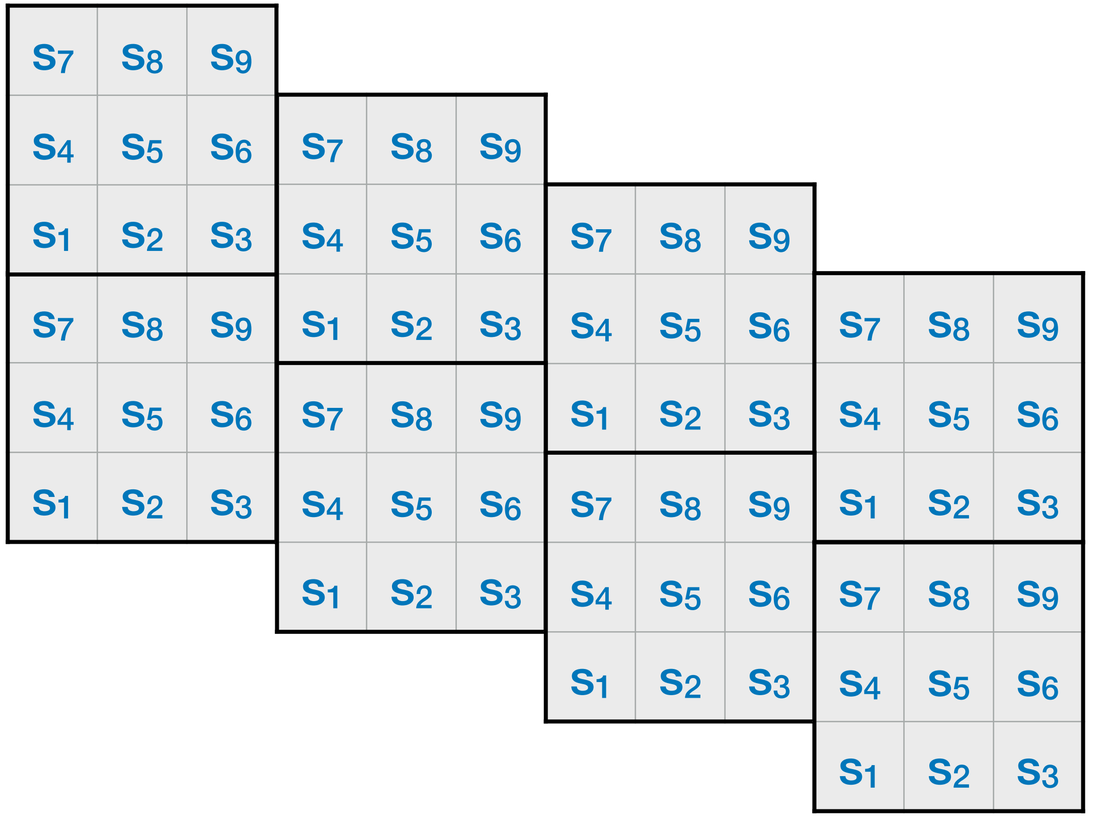
- **Periodic Boundary Condition**



- Fixed Boundary Condition

- Open Boundary Condition

- **Helical Boundary Condition**



Measurements

- magnetization (order parameter) $\langle m \rangle = \frac{1}{N} \left\langle \left| \sum_{i=1}^{N} s_i \right| \right\rangle$
- magnetic susceptibility $\chi = \beta N \left(\langle m^2 \rangle - \langle m^2 \rangle \right)$
- specific heat $c = \frac{\beta^2}{N} \left(\langle E^2 \rangle - \langle E^2 \rangle \right)$
- Binder's (fourth-order) cumulant $U_4 = 1 - \frac{\langle m^4 \rangle}{3 \langle m^2 \rangle^2}$
- correlation function $G_c^{(2)}(i,j) = \langle s_i s_j \rangle - \langle s_i \rangle \langle s_j \rangle$

Error estimate

You have done M *uncorrelated* measurements on A : $\{A_1, A_2, \ldots, A_M\}$. → sample mean $\bar{A} = \frac{1}{M}\sum_{i=0}^{M} A_i$.

$$\langle \bar{A} \rangle = \frac{1}{M}\sum_i \langle A_i \rangle = \langle A \rangle$$ best estimate of a sample mean = true expectation value

Standard error : standard deviation of a *true* estimate

$$\sigma_A^2 = \langle A^2 \rangle - \langle A \rangle^2$$ → ***We do not know it.***

Our error estimate : deviation of $\bar{A}$ from $\langle A \rangle$

We know it for the measurements!

$$\Delta_A^2 = \left\langle \left(\bar{A} - \langle A \rangle\right)^2 \right\rangle = \frac{\sigma_A^2}{M} \approx \frac{s_A^2}{M-1}$$

$$s_A^2 = \overline{(A^2)} - (\bar{A})^2 = \frac{1}{M}\sum_i A_i^2 - \left(\frac{1}{M}\sum_i A_i\right)^2$$

$$\langle s_A^2 \rangle = \frac{M-1}{M}\left[\langle A^2 \rangle - \langle A \rangle^2\right] = \frac{M-1}{M}\sigma_A^2$$

Equilibration time

- Monte Carlo measurements should be started after the Markov chain provides a distribution sufficiently close to the asymptotic distribution.

- The number of equilibration (thermalization) steps should be several (>10) times of an autocorrelation time measured from the very beginning.

$$\tau_{\mathrm{eq}} = \frac{\sum_{t=1}^{\infty} \left(\langle A_0 A_t \rangle - \langle A \rangle^2 \right)}{\langle A_0 \rangle \langle A \rangle - \langle A \rangle^2}$$

- This is one of the bottleneck for parallelization with massive independent MC runs across many CPUs. Every cpu needs its own thermalization.

Autocorrelation time

- A measure of how long it takes from one state to a significantly different another state. *Ideally*, it should be a time interval between different measurements, which is practically not possible.

- Usually the equilibration time is much longer than the autocorrelation time.

- Autocorrelation function $C(t) = \langle A(t_i)A(t_i + t)\rangle - \langle A\rangle^2 \propto \exp(-t/\tau_{\text{exp}})$

- Integrated autocorrelation time

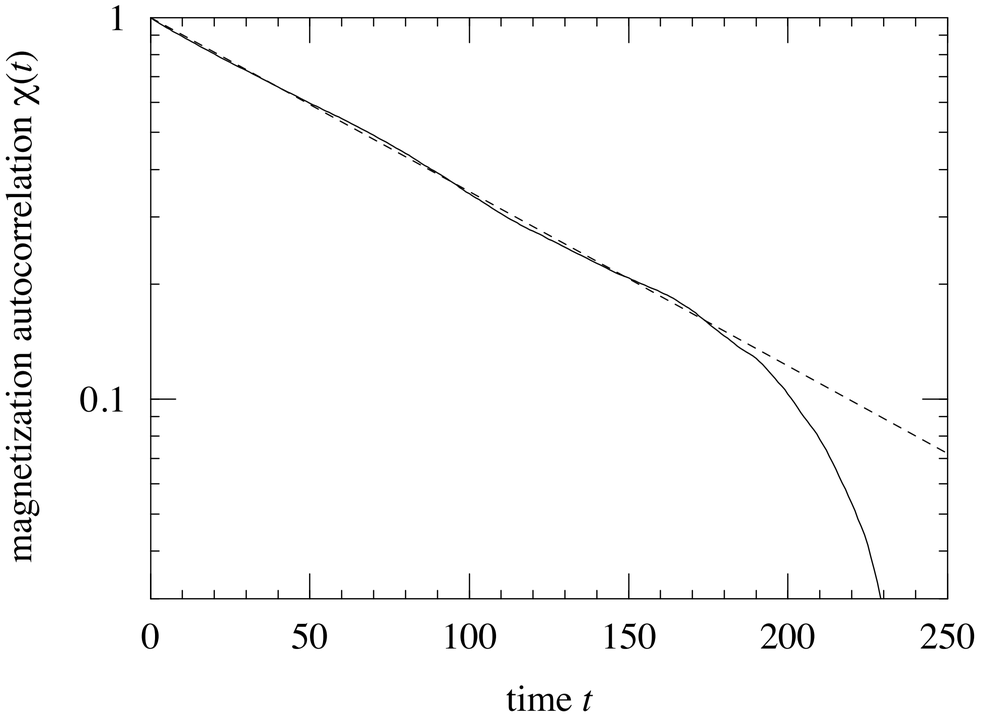
$$\tau_{\text{int}} \equiv \sum_{t=1}^{\infty} \frac{C(t)}{C(0)} = \frac{\sum_{i=1}^{\infty}\left(\langle A_1 A_{1+t} - \langle A\rangle^2\right)}{\langle A^2\rangle - \langle A\rangle^2} \approx \int_0^{\infty} \frac{C(t)}{C(0)} dt = \tau_{\text{exp}}$$

if purely exponential

Exponentially decaying autocorrelation function

$$C(t) = \langle A(t'+t)A(t')\rangle - \langle A\rangle^2 = \frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t} A(t'+t)A(t') - \frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t} A(t'+t)\cdot\frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t} A(t')$$

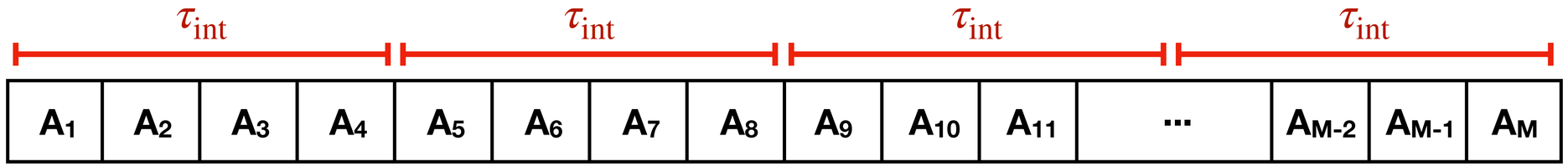
(Newman and Barkema)



Trouble happens when *t* gets close to t_{max} :

- *very large statistical fluctuations*
- *poor accuracy at its tail*
- *Thus, you have to run it for a very long time (as long as many correlation times)!*

Binning analysis



Error estimate for correlated samples:

$$\Delta_A^2 \equiv \left\langle (\bar{A} - \langle A \rangle)^2 \right\rangle = \frac{\sigma_A^2}{M} + \frac{1}{M^2} \sum_{i \neq j} \left(\langle A_i A_j \rangle - \langle A \rangle^2 \right)$$

Assumption: correlation decay rapidly as $|i - j| \to \infty$.

$$(\text{second term}) = \frac{2}{M^2} \sum_{i=1}^{M} \sum_{t=1}^{M-i} \left(\langle A_i A_{i+t} \rangle - \langle A \rangle^2 \right) \approx \frac{2}{M} \sum_{t=1}^{\infty} \left(\langle A_1 A_{1+t} \rangle - \langle A \rangle^2 \right)$$

$$\frac{2\sigma_A^2}{M} \tau_{\text{int}}^A$$

$$\Delta_A = \sigma_A \sqrt{\frac{1 + 2\tau_{\text{int}}^A}{M}}$$

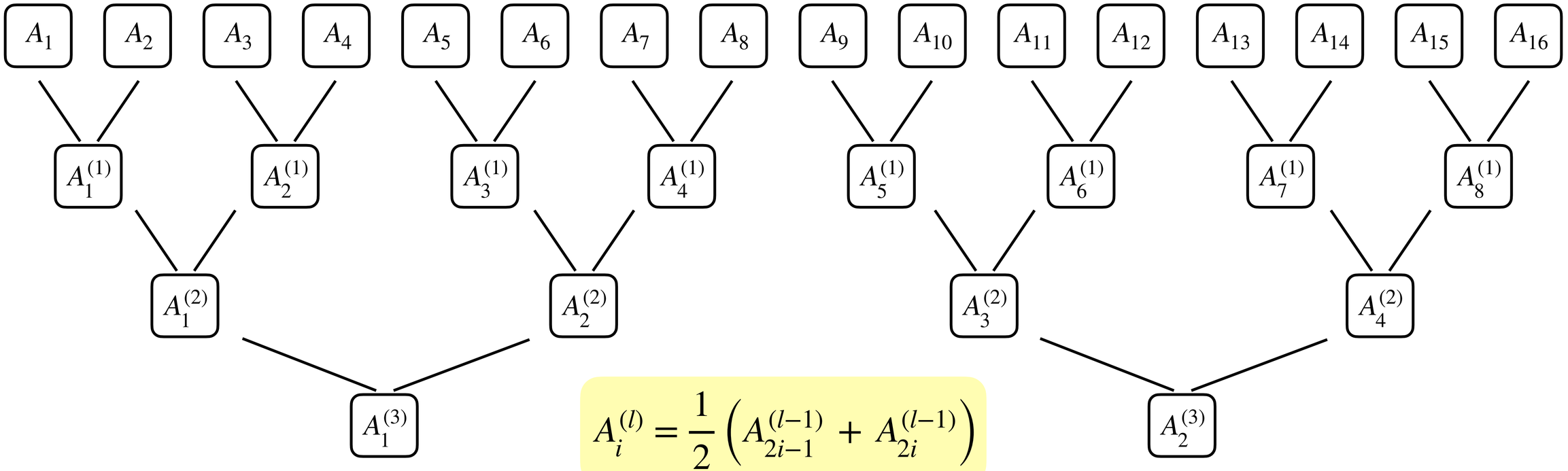
← ***corrected error***

V. Ambegaokar and M. Troyer, Am. J. Phys. 78, 150 (2010)

Binning analysis

V. Ambegaokar and M. Troyer, Am. J. Phys. 78, 150 (2010) [arXiv:0906.0943]

Measuring error estimate and integrated autocorrelation time at once.



$$A_i^{(l)} = \frac{1}{2}\left(A_{2i-1}^{(l-1)} + A_{2i}^{(l-1)}\right)$$

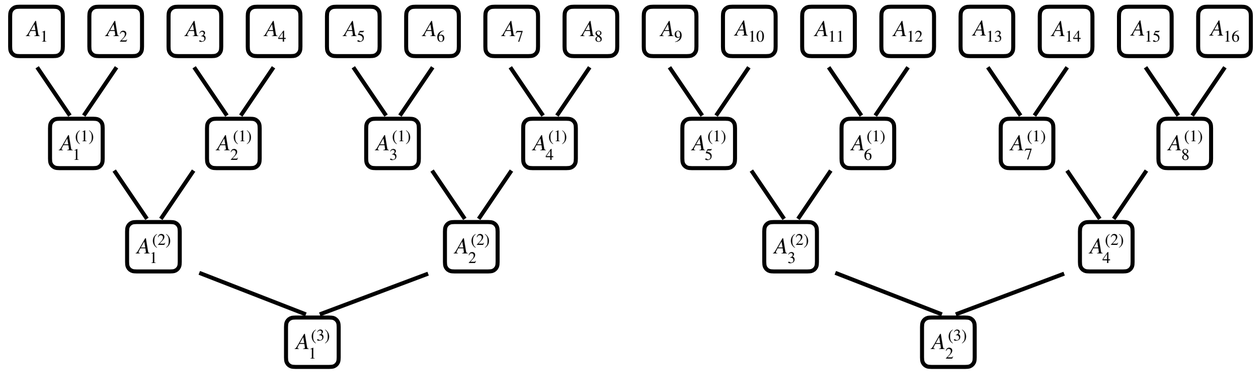
Less correlated as it proceeds to the next layer.

Binning analysis

V. Ambegaokar and M. Troyer, Am. J. Phys. 78, 150 (2010) [arXiv:0906.0943]

$$A_i^{(l)} = \frac{1}{2}\left(A_{2i-1}^{(l-1)} + A_{2i}^{(l-1)}\right)$$

$$i = 1,\ldots,M_l \equiv M/2^l$$



Error estimate at the *l*-th layer if uncorrelated

$$\Delta_A^{(l)} = \left[\frac{1}{M_l(M_l-1)}\sum_{i=1}^{M_l}\left(A_i^{(l)} - \overline{A^{(l)}}\right)^2\right]^{1/2} \xrightarrow{l\to\infty} \Delta_A = \sigma_A\sqrt{\frac{1+2\tau_{\text{int}}^A}{M}}$$

Integrated autocorrelation time

$$\tau_{\text{int}}^A = \frac{1}{2}\lim_{l\to\infty}\left[\left(\frac{\Delta_A^{(l)}}{\Delta_A^{(0)}}\right)^2 - 1\right]$$

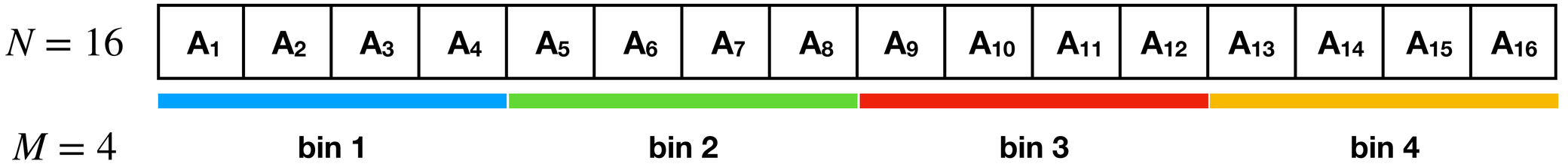
Test if your M is large enough to make $\Delta_A^{(l)}$ converged.

Jackknife method

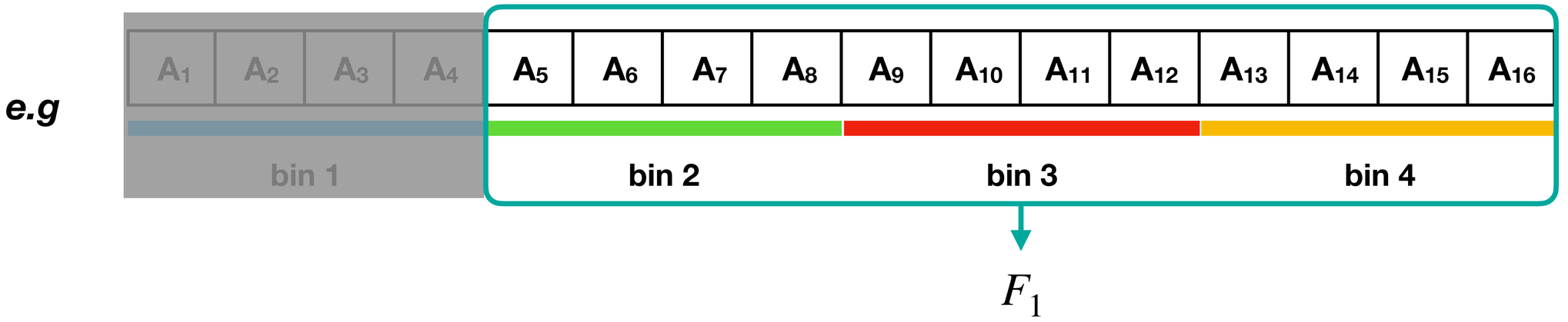
- What if your quantity is a function of other estimates, *e.g. specific heat, magnetic susceptibility, and Binder's cumulant.*
- Magnetic susceptibility $\chi = \beta N \left(\langle m^2 \rangle - \langle m^2 \rangle \right)$: how can you estimate the error of this quantity? The binning analysis is not applicable.
- Blocking method, bootstrap resampling method, **jackknife method**.
- Jackknife method: *easy*, *fast*, but *independent samples* needed.

Jackknife method

1. Split the measurements into M bins of size $N/M \gg \tau_{\mathrm{int}}$.



2. Try to discard each bin to prepare a measurement of F_i with remaining data



Jackknife method

3. Jackknife error estimate

Jackknife measurements: $F_0, F_1, F_2, \ldots, F_M$ *(*F_0 *: computation with a full data set)*

Expectation value

$$\langle F \rangle \approx \bar{F} = F_0$$

Error estimate

$$\Delta_F \approx \sqrt{\sum_{i=1}^{M} \left(F_i - \bar{F}\right)^2}$$

Jackknife or Bootstrap ?

Jackknife

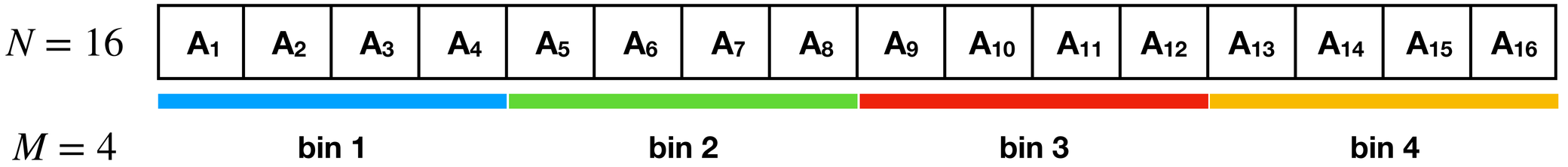
+ quick and easy
+ more suitable for small samples
- poor for non-smooth estimator

Bootstrap

+ all purposes, no assumption
+ more precise
- computationally more intensive

Bootstrap resampling

1. Split the measurements into M bins of size $N/M \gg \tau_{\mathrm{int}}$.



2. Pick randomly M bins with replacement. Compute the quantity of interest.

3. Repeat Step 1-2 (a large number) N_{b} times. Now you have done N_{b} measurements.

4. Compute the error estimate at a given confidence interval with the distribution.

5. Often, the bootstrap estimate of the error is measured as:

$$\Delta_F = \sqrt{\frac{1}{N_b - 1} \sum_{i=0}^{N_b} \left(F_i - \langle F \rangle \right)^2}$$

Finite-size-scaling analysis

- Correlation length is always *finite* in a finite-size system.
- Strictly speaking, a phase transition does not exist in a finite-size system.
- Although, you can still define a *pseudo-critical point* which is getting closer and closer to a true critical point as the system size increases.
- Critical behaviors can be also studied with the *finite-size-scaling ansatz* of macroscopic observables.

Pseudo-critical point & Finite-size-scaling ansatz

At the critical point, the correlation length ξ diverges as $\xi \sim |T - T_c|^{-\nu}$ in the thermodynamic limit.

However, in a finite-size system with length scale L, it has to be like $\xi_L \sim L$ at the largest.

A "pseudo"-critical point : $\xi_L \propto |T_L^* - T_c|^{-\nu} \propto L \longrightarrow T_L^* = T_c + aL^{-1/\nu}$

$|T_L^* - T_c| L^{1/\nu} = O(1)$

How does an observable showing criticality like, $X \sim |T - T_c|^{-x}$, behave in a finite-size system?

Finite-Size-Scaling ansatz : $X_L(T) = L^y X_o(L/\xi) \longrightarrow X_L(T) = L^{x/\nu} \tilde{X}_o\left(|T - T_c| L^{1/\nu}\right)$

$y = x/\nu$

$X_L(T_L^*) = L^y \tilde{X}_o(L/\xi_L) \propto L^y \longleftrightarrow X_L(T_L^*) \propto |T_L^* - T_c|^{-x} \propto \xi_L^{x/\nu} \propto L^{x/\nu}$

Finite-size-scaling ansatz

leading-order behavior in $t \equiv (T - T_c)/T_c$

- magnetization

$$\langle m \rangle = \frac{1}{L^d} \left\langle \left| \sum_{i=1}^{L^d} s_i \right| \right\rangle \longrightarrow m_L = L^{-\beta/\nu} \mathscr{M}_o \left(tL^{1/\nu} \right)$$

- magnetic susceptibility

$$\chi = \beta L^d \left(\langle m^2 \rangle - \langle m^2 \rangle \right) \longrightarrow \chi_L = L^{-\gamma/\nu} \mathscr{X}_o \left(tL^{1/\nu} \right)$$

- specific heat

$$c = \frac{\beta^2}{L^d} \left(\langle E^2 \rangle - \langle E^2 \rangle \right) \longrightarrow c_L = L^{-\alpha/\nu} \mathscr{C}_o \left(tL^{1/\nu} \right)$$

$$c_L^* \sim \log L \ \ (\alpha = 0)$$

***2D Ising model ($\alpha = 0$) : $c(t) \sim -\log|t|$

The fourth-order cumulant

a.k.a. Binder cumulant, Binder parameter

$$U_L = 1 - \frac{\langle m^4 \rangle_L}{3\langle m^2 \rangle_L^2} \quad \xrightarrow{p_L(m,t) = L^{\beta/\nu}\tilde{p}_o(|m|L^{\beta/\nu}, tL^{1/\nu})} \quad U_L = \mathcal{U}_o\left(tL^{1/\nu}\right)$$

i.e. a true critical point = a system-size-invariant crossing point in U_L

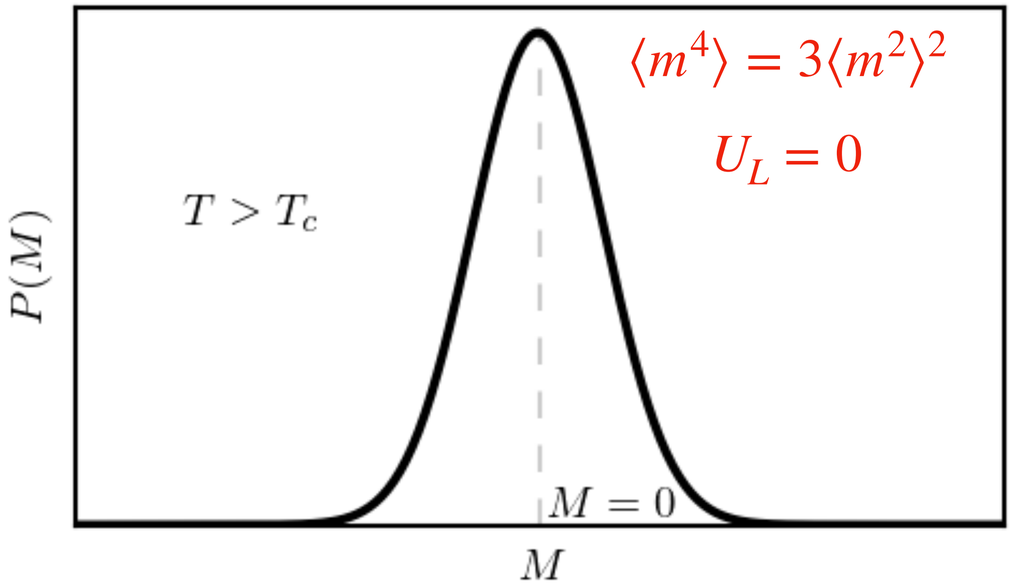
$$U_L(T) = \begin{cases} \frac{2}{3} & \text{for } T < T_c \\ \text{const.} & \text{for } T = T_c \\ 0 & \text{for } T > T_c \end{cases}$$

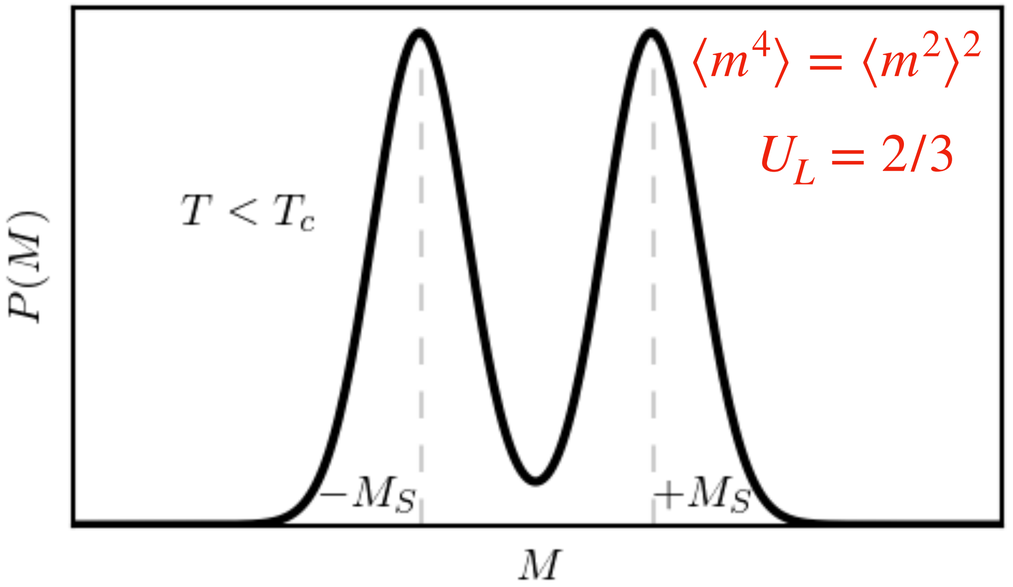
*figures taken from L. Böttcher's lecture note

The fourth-order cumulant

a.k.a. Binder cumulant, Binder parameter

$$U_L = 1 - \frac{\langle m^4 \rangle_L}{3\langle m^2 \rangle_L^2}$$





$$p_L(m) = \sqrt{\frac{L^d}{2\pi\sigma_L^2}} \exp\left(-\frac{L^d m^2}{2\sigma_L^2}\right)$$

$$p_L(m) = \frac{1}{2}\sqrt{\frac{L^d}{2\pi\sigma_L^2}} \left\{ \exp\left[-\frac{L^d(m-m_s)^2}{\sigma_L^2}\right] + \exp\left[-\frac{L^d(m-m_s)^2}{\sigma_L^2}\right] \right\}$$

Critical slowing down

- At the critical point, the *correlation time* diverges, too!

$$\tau \sim |t|^{-z\nu} \qquad z : \textit{dynamical exponent}$$

- It is getting worse as the system size increases!

$$\xi \sim |t|^{-\nu} \longrightarrow \tau \sim \xi^z \sim L^z$$

- This is a major issue with the single-spin update scheme.
  - It would need extremely long MC sweeps to simulate a large-size system.

⟶ ***What about updating many spins instead of a single spin?***

Cluster algorithms

- Why not updating many spins at once, instead of a single spin?
- Cluster update algorithms ➔ much shorter autocorrelation time
  ➔ significantly reduces the critical slowing down
- Popular algorithms in classical spin systems: Swendsen-Wang, Wolff cluster updates
- Although, the cluster update schemes are model-dependent. *There are cases where a cluster update strategy is not available.*

Fortuin-Kasteleyn representation

$$p = 1 - e^{-2\beta J}$$

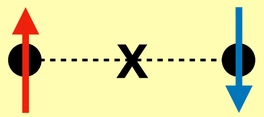
$$Z = \sum_{\{s_i\}} e^{\beta J \sum_{\langle i,j \rangle} s_i s_j} = \sum_{\{s_i\}} \prod_{\langle i,j \rangle} e^{\beta J} \left[(1-p) + p\delta_{s_i,s_j} \right] = \sum_{\{s_i\}} \sum_{n_{ij}} \prod_{\langle i,j \rangle} e^{\beta J} \left[(1-p)\delta_{n_{ij},0} + p\delta_{s_i,s_j}\delta_{n_{ij},1} \right]$$

$$= \sum_{\{s_i\}} \sum_{n_{ij}} \prod_{\langle i,j \rangle} \left[e^{-\beta J}(1-\delta_{s_i,s_j})\delta_{n_{ij},0} + e^{\beta J}\delta_{s_i,s_j} \left\{ (1-p)\delta_{n_{ij},0} + p\delta_{n_{ij},1} \right\} \right] \equiv \sum P(\boldsymbol{\sigma}, \mathbf{n})$$

→ ***bond percolation at a given spin configuration***

different neighboring spins

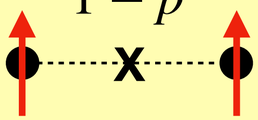
deleted (n=0)



same neighboring spins

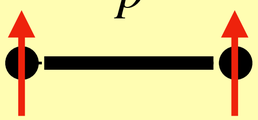
deleted (n=0)

$1-p$



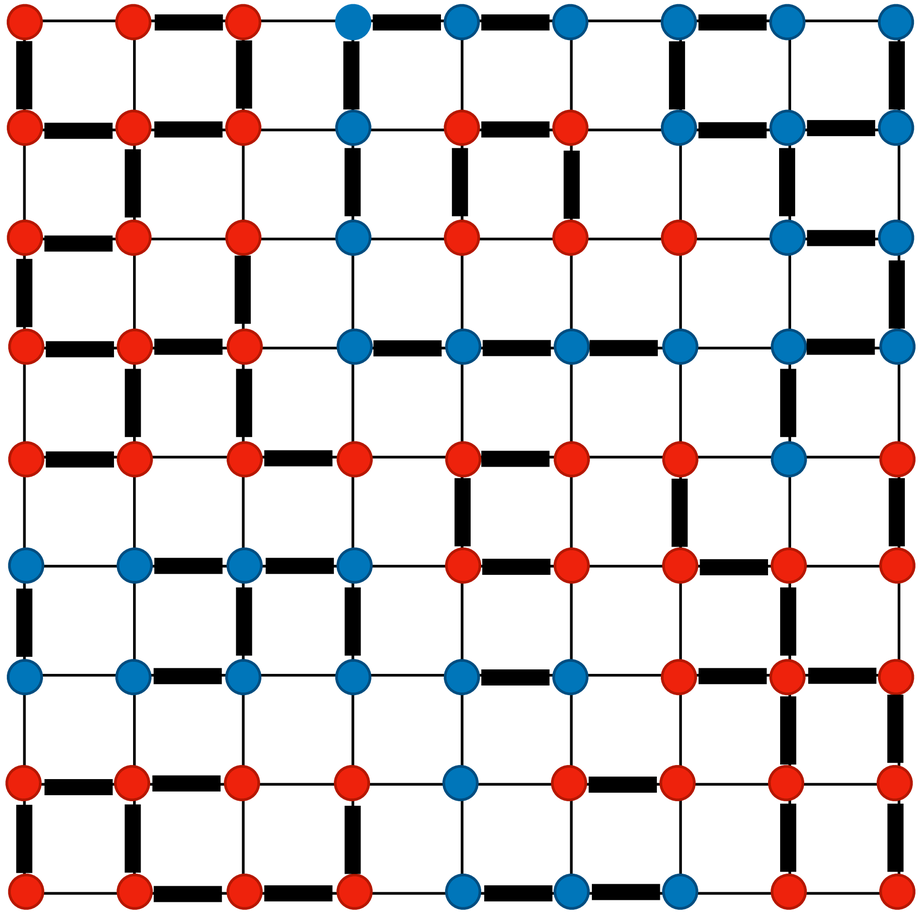
active (n=1)

p



Swendsen-Wang algorithm

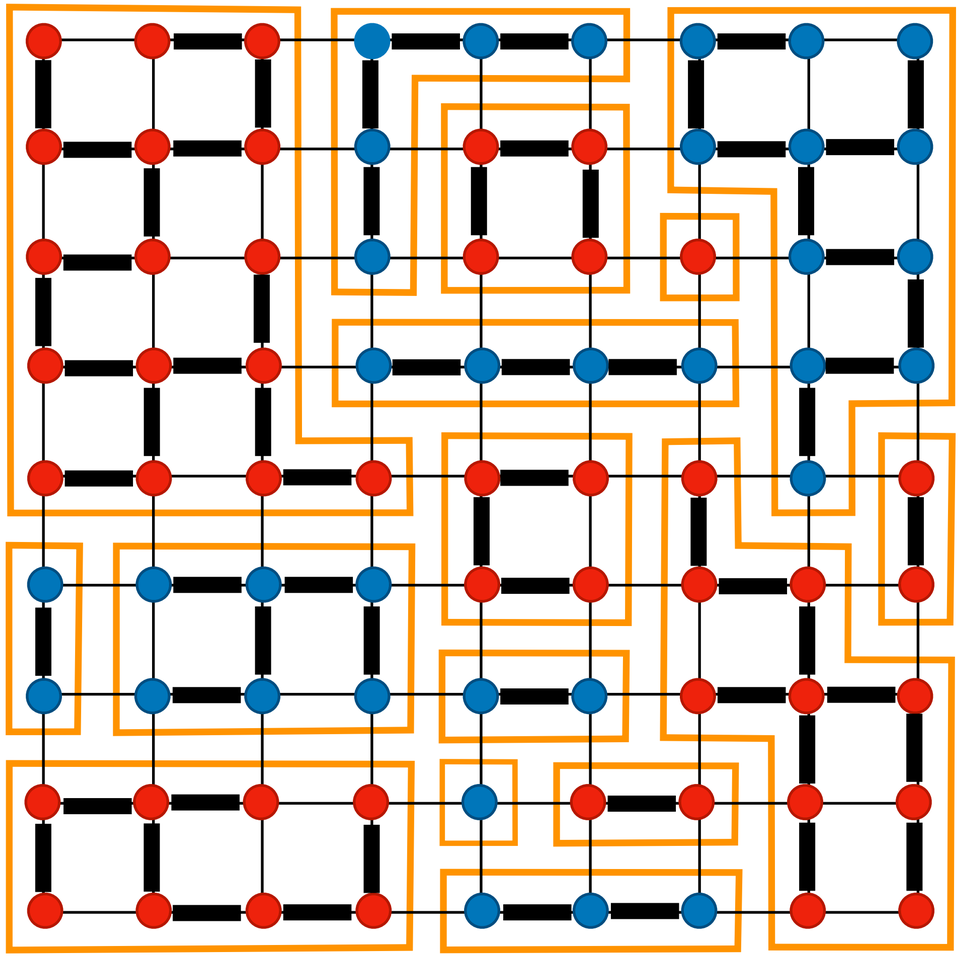
1. Mark active bonds with probability ***p*** for neighboring bonds between the same spins. $p = 1 - \exp[-2\beta J]$



R. H. Swendsen and J.-S. Wang, PRL 58, 86 (1987).
J.-S. Wang and R. H. Swendsen, Physica A 167, 565 (1990).

Swendsen-Wang algorithm

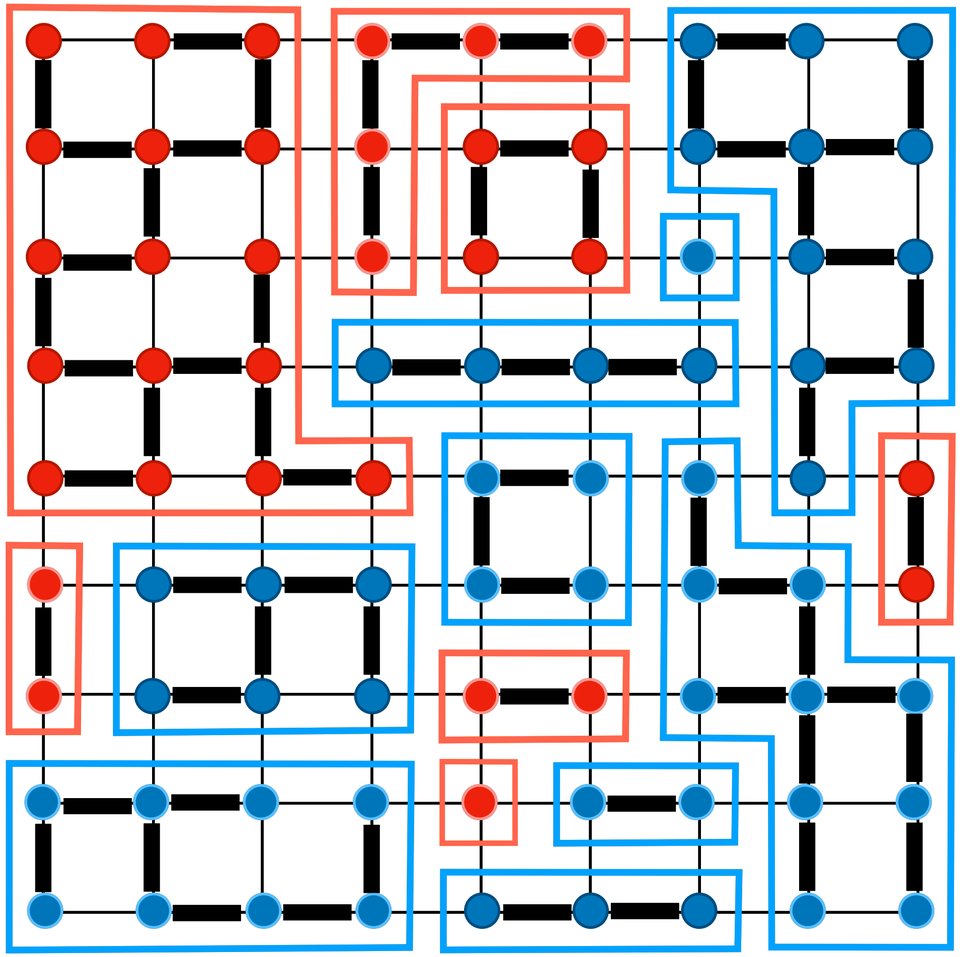
1. Mark active bonds with probability ***p*** for neighboring bonds between the same spins. $p = 1 - \exp[-2\beta J]$
2. Identify ALL clusters connected through the active bonds.



R. H. Swendsen and J.-S. Wang, PRL 58, 86 (1987).
J.-S. Wang and R. H. Swendsen, Physica A 167, 565 (1990).

Swendsen-Wang algorithm

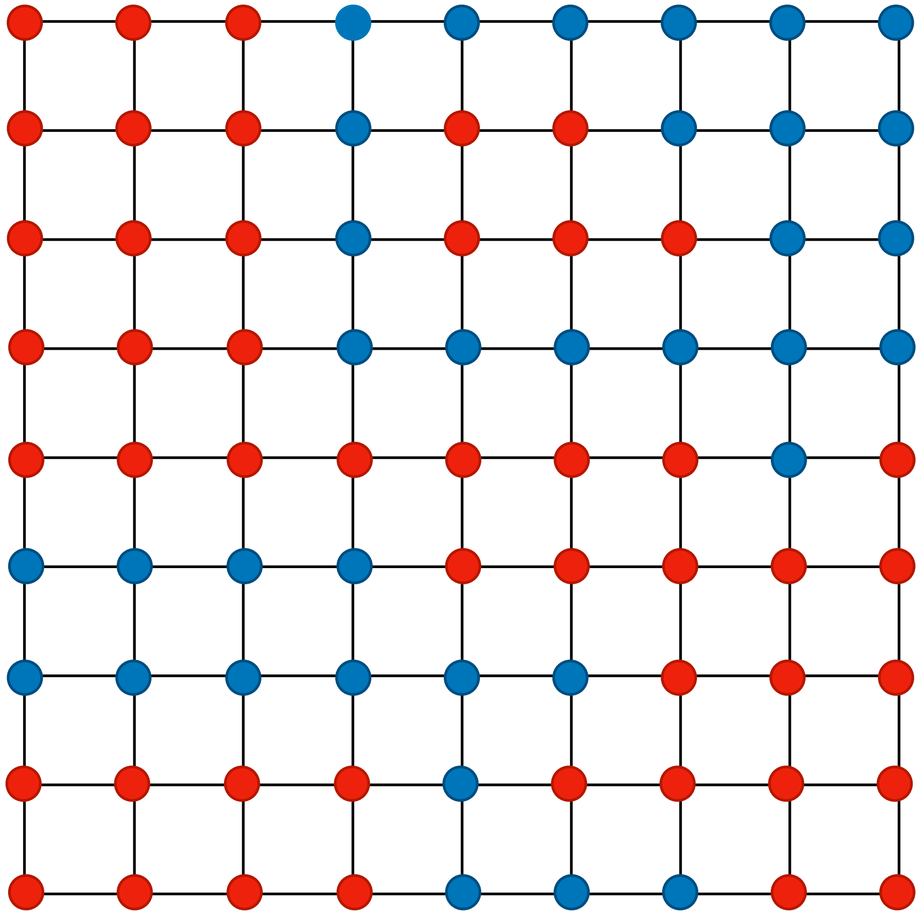
1. Mark active bonds with probability ***p*** for neighboring bonds between the same spins. $p = 1 - \exp[-2\beta J]$
2. Identify ALL clusters connected through the active bonds.
3. Assign a new spin +1 or -1 randomly for each cluster.



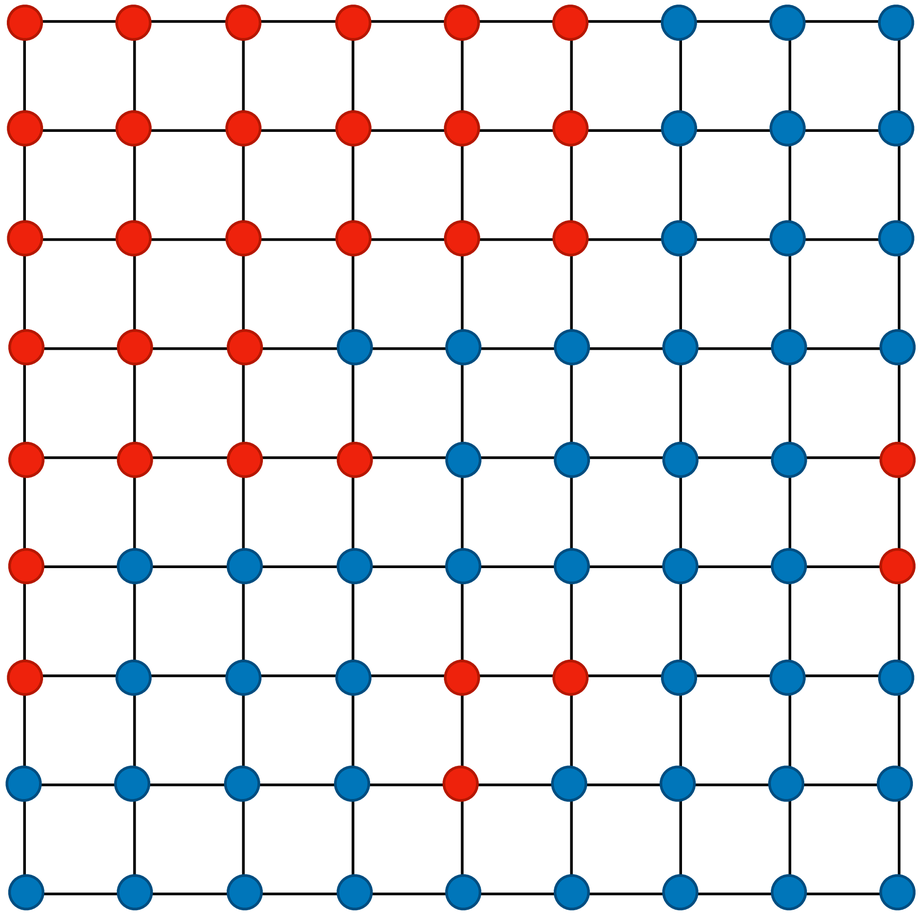
R. H. Swendsen and J.-S. Wang, PRL 58, 86 (1987).
J.-S. Wang and R. H. Swendsen, Physica A 167, 565 (1990).

Swendsen-Wang algorithm

OLD



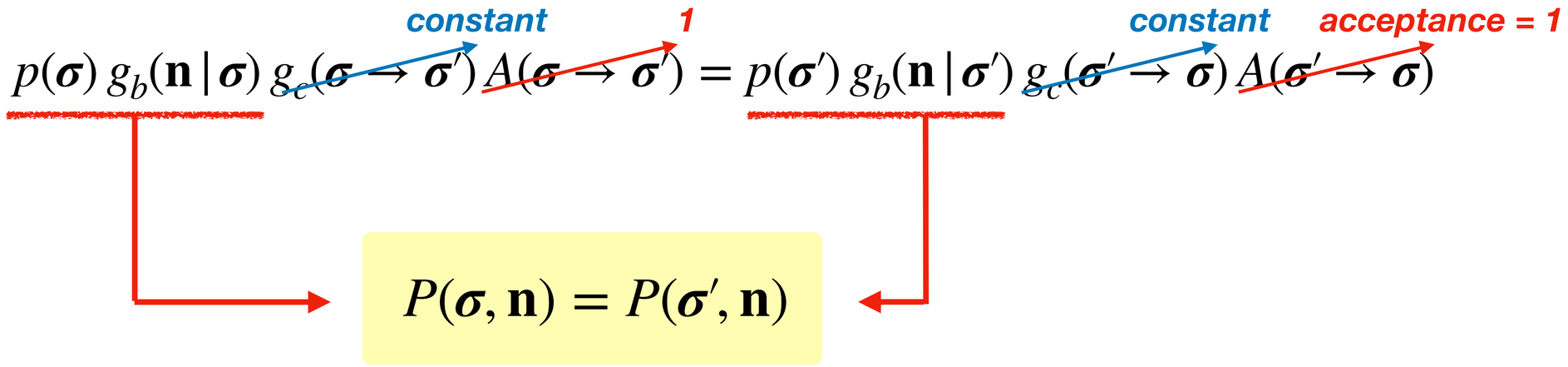
NEW



Detailed Balance

detailed balance condition $$p(\boldsymbol{\sigma})W(\boldsymbol{\sigma} \to \boldsymbol{\sigma}') = p(\boldsymbol{\sigma}')W(\boldsymbol{\sigma}' \to \boldsymbol{\sigma})$$

SW update

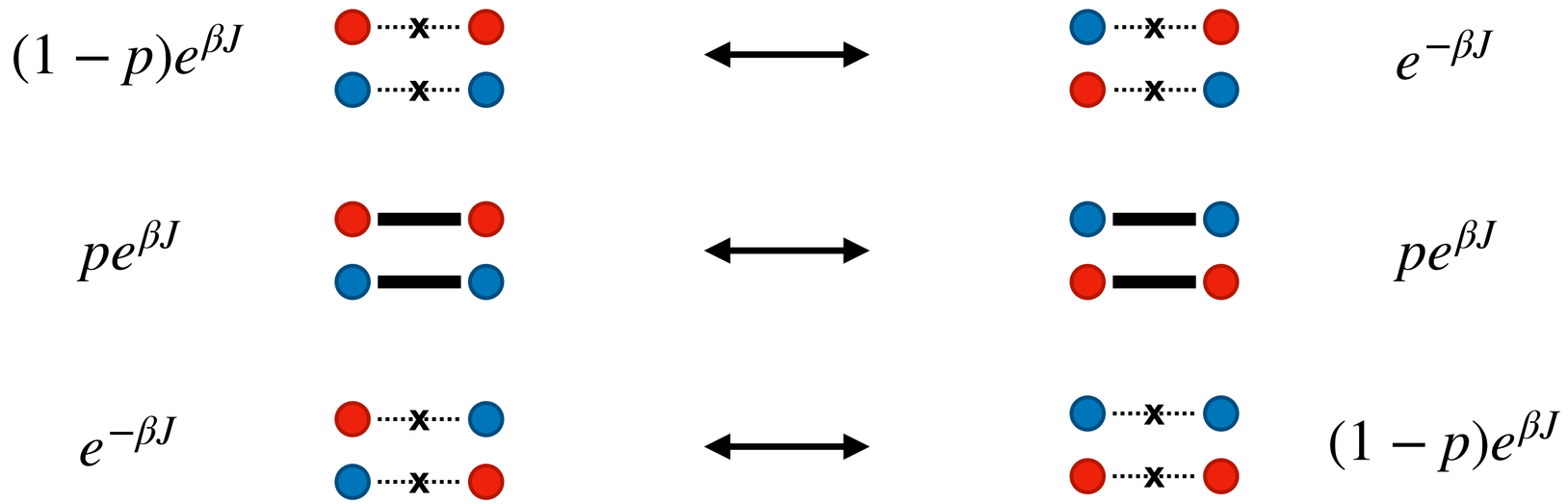


$$Z = \sum_{\{s_i\}} \sum_{n_{ij}} \prod_{\langle i,j \rangle} \left[e^{-\beta J}(1 - \delta_{s_i,s_j})\delta_{n_{ij},0} + e^{\beta J}\delta_{s_i,s_j} \left\{ (1-p)\delta_{n_{ij},0} + p\delta_{n_{ij},1} \right\} \right] \equiv \sum P(\boldsymbol{\sigma}, \mathbf{n})$$

Detailed Balance

$$Z = \sum_{\{s_i\}} \sum_{n_{ij}} \prod_{\langle i,j \rangle} \left[e^{-\beta J}(1 - \delta_{s_i,s_j})\delta_{n_{ij},0} + e^{\beta J}\delta_{s_i,s_j} \left\{ (1-p)\delta_{n_{ij},0} + p\delta_{n_{ij},1} \right\} \right] \equiv \sum P(\boldsymbol{\sigma}, \mathbf{n})$$

$$P(\boldsymbol{\sigma}, \mathbf{n}) = P(\boldsymbol{\sigma}', \mathbf{n}) \quad \longleftarrow \quad \textbf{\textit{Try}}\ p = 1 - e^{-2\beta J}\ \textbf{!}$$



Numerical Bottleneck

- In the Swendsen-Wang algorithm, "all" clusters must be identified.
- Cluster identification can be a major bottleneck.
- Hoshen-Kopelmann algorithm for cluster labelling
- Newman-Ziff algorithm
- Wolff update is preferred if available because of its simplicity.

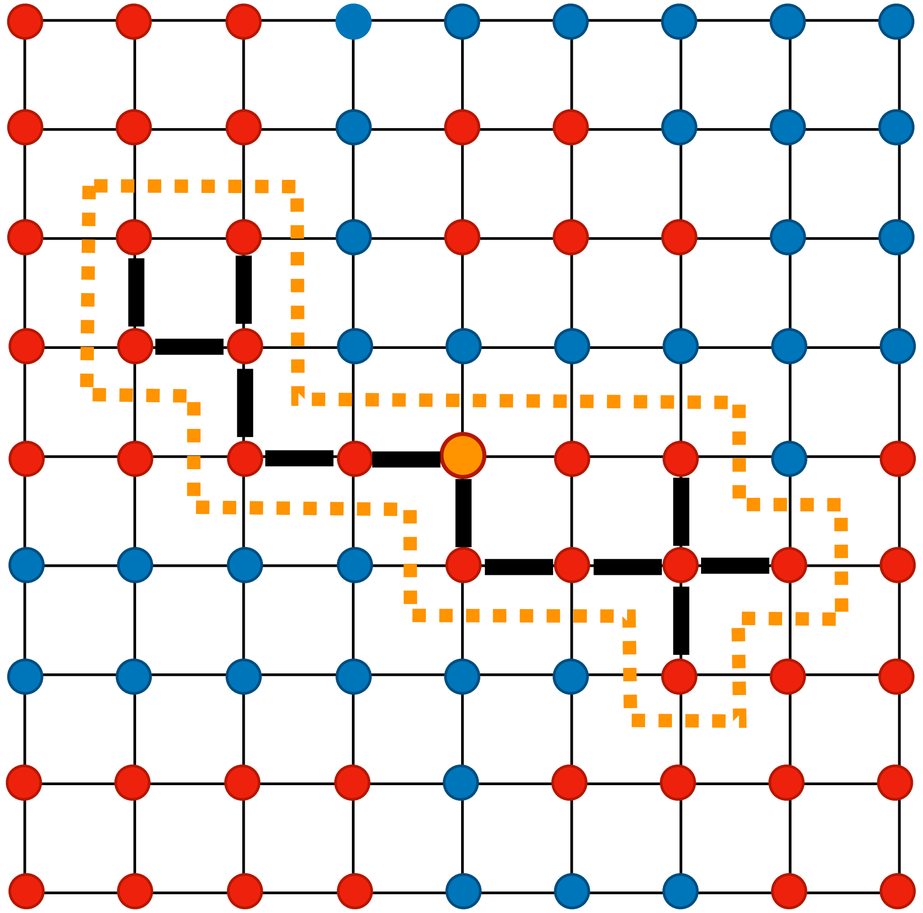
Wolff algorithm

1. Choose a random site to be a starting site.

2. From the starting site, make a cluster with the neighboring sites of the same spin through active bonds chosen with probability
   $p = 1 - \exp(-2\beta J)$.

3. Expand the cluster by inspecting the neighbors of newly joined sites until it stops growing.

4. Flip the spin of the cluster.



U. Wolff, PRL 62, 361 (1989).

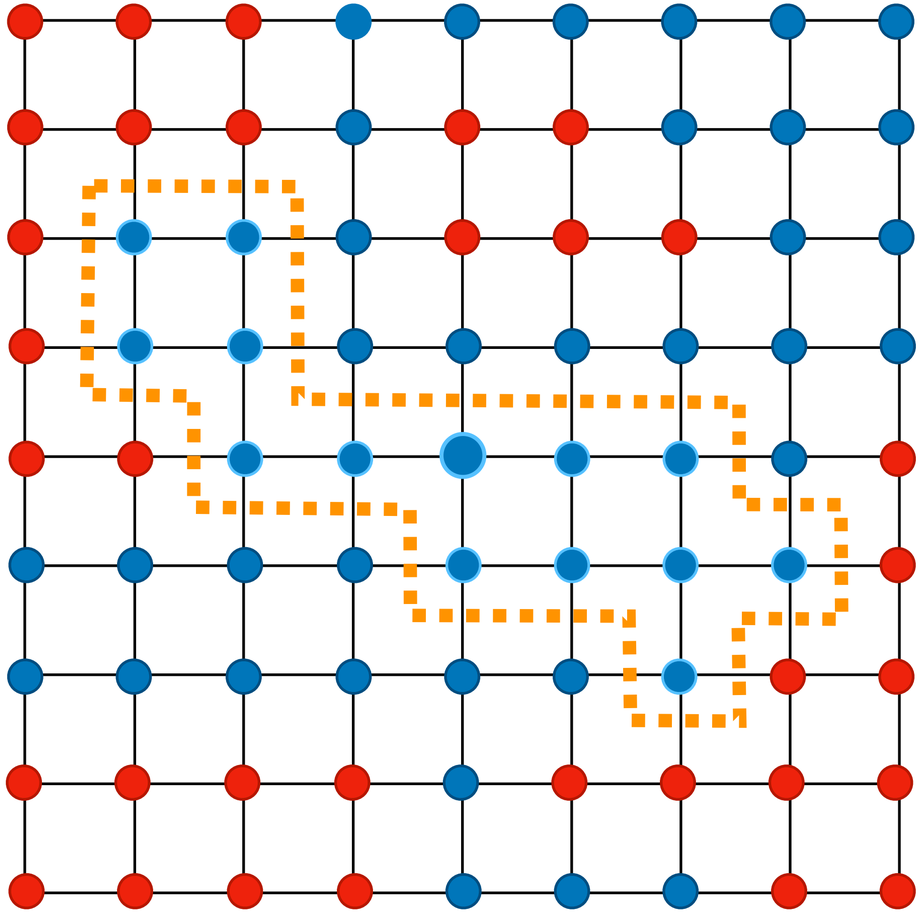
Wolff algorithm

1. Choose a random site to be a starting site.

2. From the starting site, make a cluster with the neighboring sites of the same spin through active bonds chosen with probability
   $p = 1 - \exp(-2\beta J)$.

3. Expand the cluster by inspecting the neighbors of newly joined sites until it stops growing.

4. Flip the spin of the cluster.



U. Wolff, PRL 62, 361 (1989).

Detailed Balance

OLD STATE (A)

stopping dead bonds (red-red) = d_A

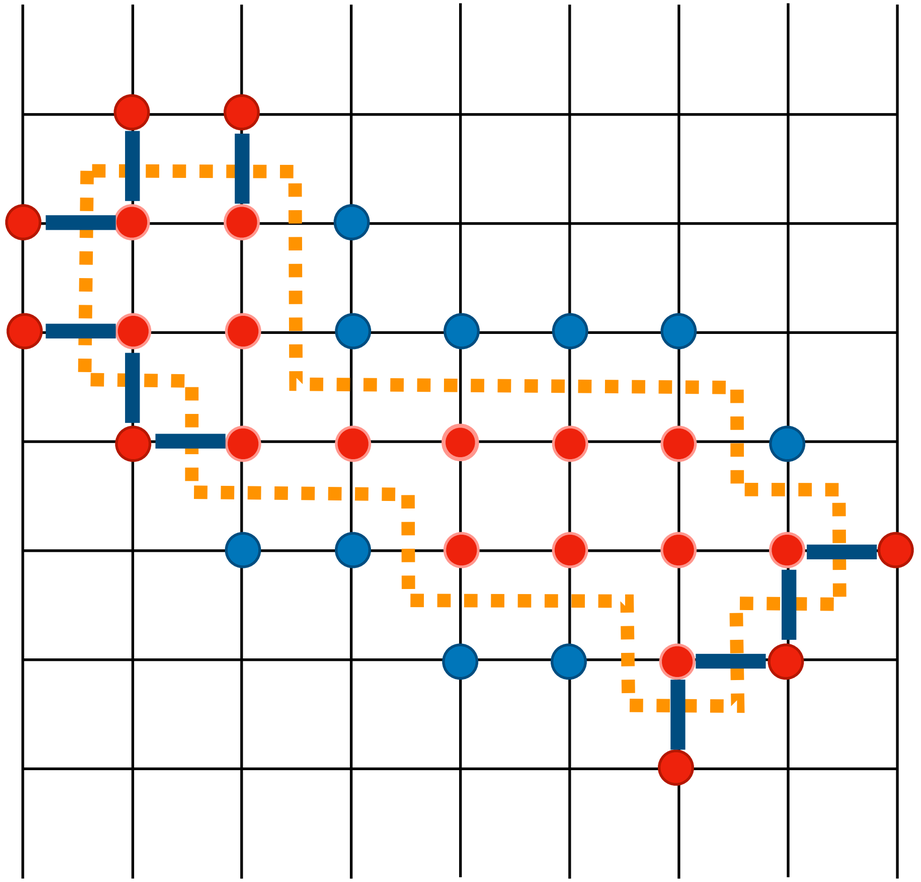
other dead bonds (red-blue) = d_B

transition probability

$$W(\boldsymbol{\sigma}_A \to \boldsymbol{\sigma}_B) \propto (1-p)^{d_A}$$

energy

$$E_A = -Jd_A + Jd_B + (\text{irrelevant})$$



Detailed Balance

NEW STATE (B)

stopping dead bonds (blue-blue) = d_B

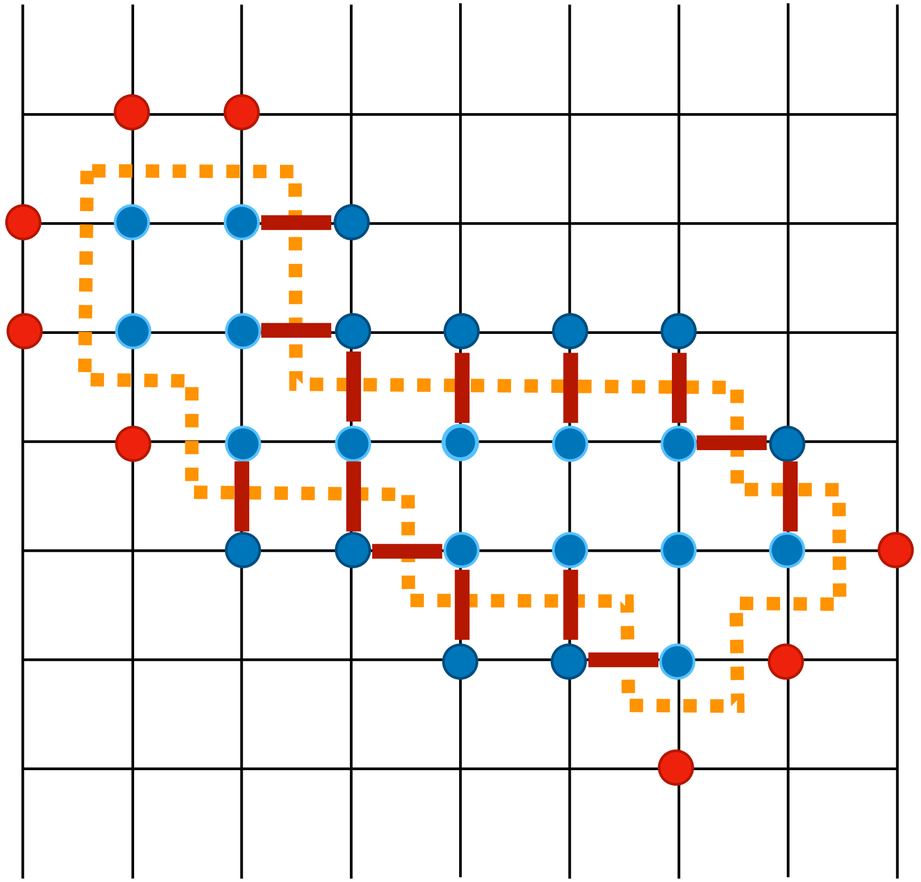
other dead bonds (blue-red) = d_A

transition probability

$$W(\boldsymbol{\sigma}_B \rightarrow \boldsymbol{\sigma}_A) \propto (1-p)^{d_B}$$

energy

$$E_B = -Jd_B + Jd_A + (\text{irrelevant})$$



Detailed Balance

transition probability

$$W(\boldsymbol{\sigma}_A \to \boldsymbol{\sigma}_B) \propto (1-p)^{d_A} \qquad W(\boldsymbol{\sigma}_B \to \boldsymbol{\sigma}_A) \propto (1-p)^{d_B}$$

energy

$$E_A = -Jd_A + Jd_B + (\text{irrelevant}) \qquad E_B = -Jd_B + Jd_A + (\text{irrelevant})$$

detailed balance

$$\frac{W(\boldsymbol{\sigma}_A \to \boldsymbol{\sigma}_B)}{W(\boldsymbol{\sigma}_B \to \boldsymbol{\sigma}_A)} = \exp[-\beta(E_B - E_A)]$$

→ $(1-p)^{d_A - d_B} = \exp[-2\beta J(d_A - d_B)]$ ↔ $p = 1 - e^{-2\beta J}$

Improved estimators

Swendsen-Wang

two spin term: $s_i s_j = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are on the same cluster} \\ 0 & \text{otherwise} \end{cases}$

$\langle m \rangle = 0\,,\ \langle m^2 \rangle = \frac{1}{N^2} \sum_{ij} \langle s_i s_j \rangle = \frac{1}{N^2} \left\langle \sum_k n_k^2 \right\rangle$ n_k : number of spins in cluster *k*

Wolff

$\langle n \rangle = \left\langle \sum_k p_k n_k \right\rangle = \frac{1}{N} \left\langle \sum_k n_k^2 \right\rangle = N \langle m^2 \rangle \longrightarrow \chi = \beta \langle n \rangle$

$p_k = \frac{n_k}{N}$: probability of choosing a cluster *k* of size n_k

Dynamical exponent

Wolff single-cluster update algorithm

$$\tau = \tau_{\text{steps}} \frac{\langle n \rangle}{L^d} = \tau_{\text{steps}} \frac{\chi}{\beta L^d} \xrightarrow{\text{at } T = T_c} L^z \sim L^{z_{\text{steps}}} \cdot L^{\gamma/\nu} \cdot L^{-d}$$

$$\chi = \beta \langle n \rangle$$

$$z = z_{\text{steps}} + \frac{\gamma}{\nu} - d$$

Dimension	M(RT)²	Wolff	Swendsen-Wang
d = 2	z = 2.167(1)	z = 0.25(1)	z = 0.25(1)
d = 3	z = 2.02(2)	z = 0.33(1)	z = 0.54(2)

Remarks

- Pay enough attention to the equilibration and autocorrelation time.
- Be careful with issues like critical slowing down and impenetrable barrier.
- No one accepts Monte Carlo data without the error estimates.
- Finite-size-scaling analysis is sometimes very delicate. Practice.
- We have only covered "classic" techniques of Monte Carlo simulations. There are many "modern" developments. A lot to study 😀