Metropolis Monte Carlo Simulation: Q & A

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California

Email: anakano@usc.edu

Good question $\xrightarrow[\text{help}]{}$ Operational understanding of MPMC!

USC Discovery Cluster

Q: Why my ising.c doesn't compile on USC Discovery cluster* (discovery.usc.edu)?

A: Please load the standard software module and use Gnu C compiler (gcc) instead of cc.

```
[anakano@discovery1]$ module purge
[anakano@discovery1]$ module load usc
[anakano@discovery1]$ gcc -o ising ising.c -lm
[anakano@discovery1]$ ./ising
Input JdivT HdivT Sta_step
0.2 0.0 2000000
avgM & sigM = -6.847660e-01 3.389275e+01
```

Remember to plot the absolute value of magnetization

*We are not providing a class account on Discovery for this course, but you are welcome to use it if you already have an account.

Metropolis Inequalities

Q: How to handle *exp_val* = $\exp(-\delta V / k_{\mathrm{B}}T) = 1$?

A: Either accept it unconditionally or conditionally with probability 1; let us (arbitrarily) pick: `if (exp_val > 1.0) {}`

Q: How to accept an attempt with probability *exp_val*?

A: Let us use

```
else if ((rand()/(double)RAND_MAX) <= exp_val) {}
```

Always *true* for *exp_val*=1.0, and correct probability if *exp_val* is rational with denominator *RAND_MAX* and *rand*() ∈ [1,*RAND_MAX*].[*]

```
// Our pick for assignment 3
if (exp_val > 1.0) {
  s[i][j] = s_new;
  runM += 2.0*s_new;
}
else if (rand()/(double)RAND_MAX <= exp_val) {
  s[i][j] = s_new;
  runM += 2.0*s_new;
}
```

[*] Linear-congruential random-number generator would return an integer in the range [1,*RANDMAX-1*], while certain library returns [0,*RANDMAX*], introducing 10^{-9} discretization error (which we have in general *exp_val* values anyways).

Metropolis Inequalities (2)

Q: Could we get over with just one if statement (no else)?

A: Yes we can, though with slightly more computation.

```
// Not our pick for assignment 3
~~if (exp_val > 1.0) {~~
~~  s[i][j] = s_new;~~
~~  runM += 2.0*s_new;~~
~~}~~
~~else ~~if (rand()/(double)RAND_MAX <= exp_val) {
  s[i][j] = s_new;
  runM += 2.0*s_new;
}
```

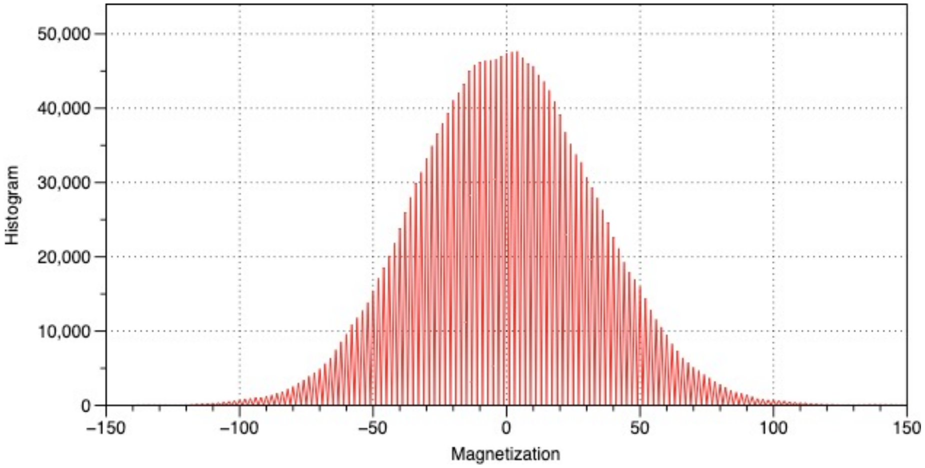
Magnetization Histogram

Q: Why so many zero entries in my histogram?

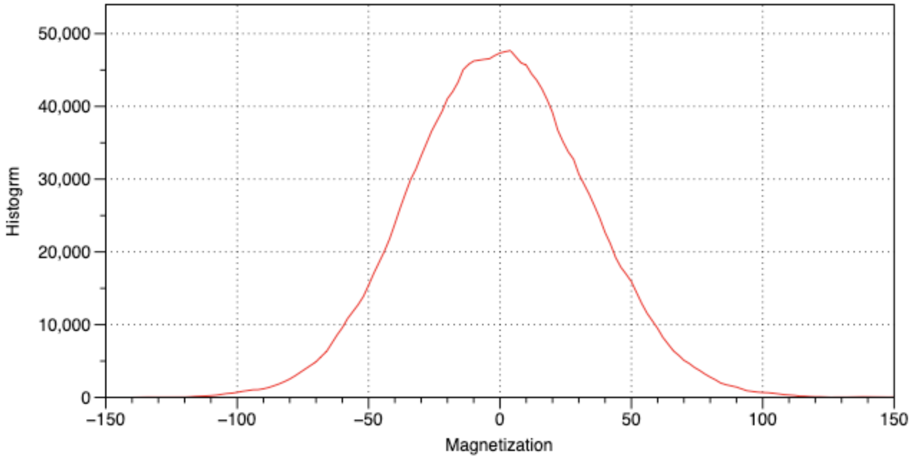
A: Spin flip conserves the parity of the total magnetization, thus no occurrence of odd magnetization.

$runM \leftarrow L^2 = 400\ (L = 20)$ // Initialization (cold start)

$runM \mathrel{+}= 2\times s\_new$ // At each spin flip



This is perfectly fine



Or eliminate all zero entries

Proving Metropolis Algorithm

Q: **How much detail is required in assignment 3, part 1?**

A: **Just show that Metropolis transition-probability matrix: (1) satisfies the detailed-balanced condition; and consequently (2) fixed-point property, *i.e.*, the desired probability is its eigenvector with eigenvalue 1.**

Metropolis Transition-probability matrix

$$\pi_{m,n} = \overbrace{\min(\frac{\rho_m}{\rho_n}, 1)}^{\text{accept/reject}} \quad \overbrace{\alpha_{m,n}}^{\text{symmetric attempt}}$$

↓

Detailed-balance condition

$$\pi_{mn}\rho_n = \pi_{nm}\rho_m$$

Equal population flux

↓

Fixed-point

$$\boldsymbol{\Pi\rho} = 1 \bullet \boldsymbol{\rho}$$
$$\sum_n \pi_{mn}\rho_n = \rho_m$$

Once you get there, stuck forever

(Filtering) Since all other eigenvalues are less than 1 in absolute value (Perron-Frobenius theorem), we get there no matter what is the initial probability

Proof Sketch

$$\pi_{mn} = \begin{cases} \alpha_{mn} & \rho_m \geq \rho_n \quad m \neq n \\ (\rho_m/\rho_n)\alpha_{mn} & \rho_m < \rho_n \quad m \neq n \\ 1 - \sum_{m' \neq n} \pi_{m'n} & \qquad\qquad m = n \end{cases}$$

1. Use to prove detailed balance
   $\pi_{mn}\rho_n = \pi_{nm}\rho_m$

2. This ensures normalization
   $\sum_m \pi_{mn} = 1$

3. Combine 1 & 2 to prove fixed-point
   $\sum_n \pi_{mn}\rho_n = \rho_m$

Q: What Is α_{mn} in Ising MC?

$N = 20 \times 20 = 400$ in assignment 3

States: $m, n \in \left\{ s^N = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{pmatrix} \middle| \, s_i = \uparrow, \downarrow; i = 1, \ldots, N \right\}$

$$\pi_{m,n} = \overbrace{\min\left(\frac{\rho_m}{\rho_n}, 1\right)}^{\text{accept/reject}} \overbrace{\alpha_{m,n}}^{\text{attempt}}$$

i.e., one spin-flip away

Attempt matrix: $\alpha_{m,n} = \begin{cases} 1/N & Hamming\_distance(m,n) = 1 \\ 0 & \text{else} \end{cases}$

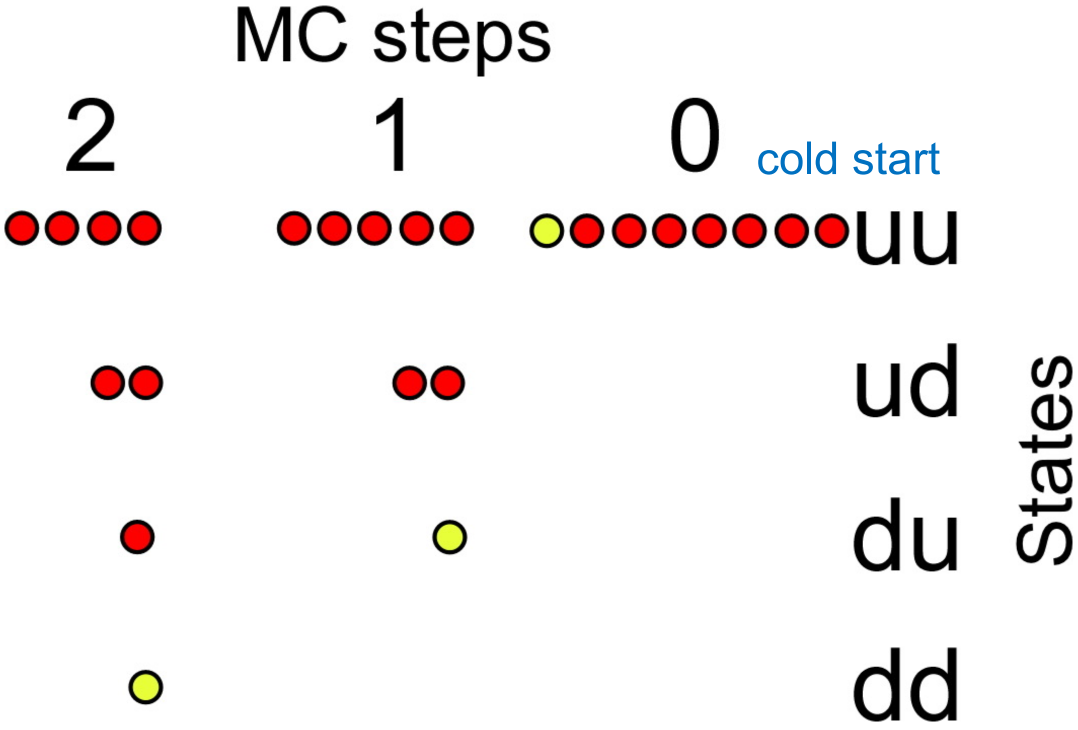
Example: $N = 3$ ($2^N = 8$ states)

$$\begin{pmatrix} \uparrow\uparrow\uparrow \\ \uparrow\uparrow\downarrow \\ \uparrow\downarrow\uparrow \\ \uparrow\downarrow\downarrow \\ \downarrow\uparrow\uparrow \\ \downarrow\uparrow\downarrow \\ \downarrow\downarrow\uparrow \\ \downarrow\downarrow\downarrow \end{pmatrix} = \begin{pmatrix} & 1/3 & 1/3 & & 1/3 & & & \\ 1/3 & & & 1/3 & & 1/3 & & \\ 1/3 & & & 1/3 & & & 1/3 & \\ & 1/3 & 1/3 & & & & & 1/3 \\ 1/3 & & & & & 1/3 & 1/3 & \\ & 1/3 & & & 1/3 & & & 1/3 \\ & & 1/3 & & 1/3 & & & 1/3 \\ & & & 1/3 & & 1/3 & 1/3 & \end{pmatrix} \begin{pmatrix} \uparrow\uparrow\uparrow \\ \uparrow\uparrow\downarrow \\ \uparrow\downarrow\uparrow \\ \uparrow\downarrow\downarrow \\ \downarrow\uparrow\uparrow \\ \downarrow\uparrow\downarrow \\ \downarrow\downarrow\uparrow \\ \downarrow\downarrow\downarrow \end{pmatrix}$$

Q: Where Is Matrix-Vector Multiplication?

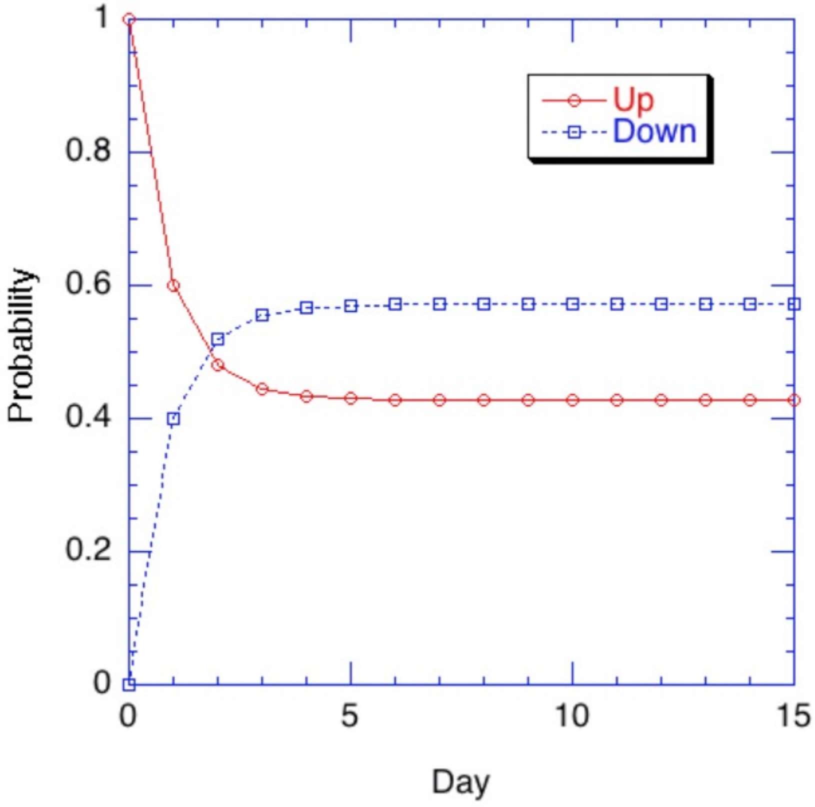
A: The probability density vector is replaced by an ensemble of individual MC sequences in Markov-chain MC; the ensemble average is then replaced by time average.

$$\boldsymbol{\rho}^{(t+1)} = \boldsymbol{\Pi}\boldsymbol{\rho}^{(t)}$$



Example: Two-Level System

$$\Pi = \begin{array}{c|cc} & \uparrow & \downarrow \\ \hline \uparrow & a & 1-b \\ \downarrow & 1-a & b \end{array} = \begin{pmatrix} 0.6 & 0.3 \\ 0.4 & 0.7 \end{pmatrix} \quad (a = 0.6, b = 0.7)$$



$$\begin{pmatrix} p_{\uparrow}^{(t)} \\ p_{\downarrow}^{(t)} \end{pmatrix} = \begin{pmatrix} 0.6 & 0.3 \\ 0.4 & 0.7 \end{pmatrix}^{t} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \xrightarrow[t \to \infty]{} \begin{pmatrix} 0.4286 \\ 0.5714 \end{pmatrix}$$

3/7
4/7

Equilibrium probability

A Metropolis Monte Carlo

Your only knowledge = equilibrium probability distribution

$$\boldsymbol{\rho} = \begin{pmatrix} 3/7 \\ 4/7 \end{pmatrix}$$

A choice of attempt matrix

$$\alpha_{\uparrow\downarrow} = \alpha_{\downarrow\uparrow} = 1$$

Detailed-balanced transition-probability matrix

$$\mathbf{\Pi} = \begin{pmatrix} \pi_{\uparrow\uparrow} & \pi_{\uparrow\downarrow} \\ \pi_{\downarrow\uparrow} & \pi_{\downarrow\downarrow} \end{pmatrix} = \begin{pmatrix} 1 - \alpha_{\downarrow\uparrow} & \alpha_{\uparrow\downarrow}(\rho_{\uparrow}/\rho_{\downarrow}) \\ \alpha_{\downarrow\uparrow} & 1 - \alpha_{\uparrow\downarrow}(\rho_{\uparrow}/\rho_{\downarrow}) \end{pmatrix}$$

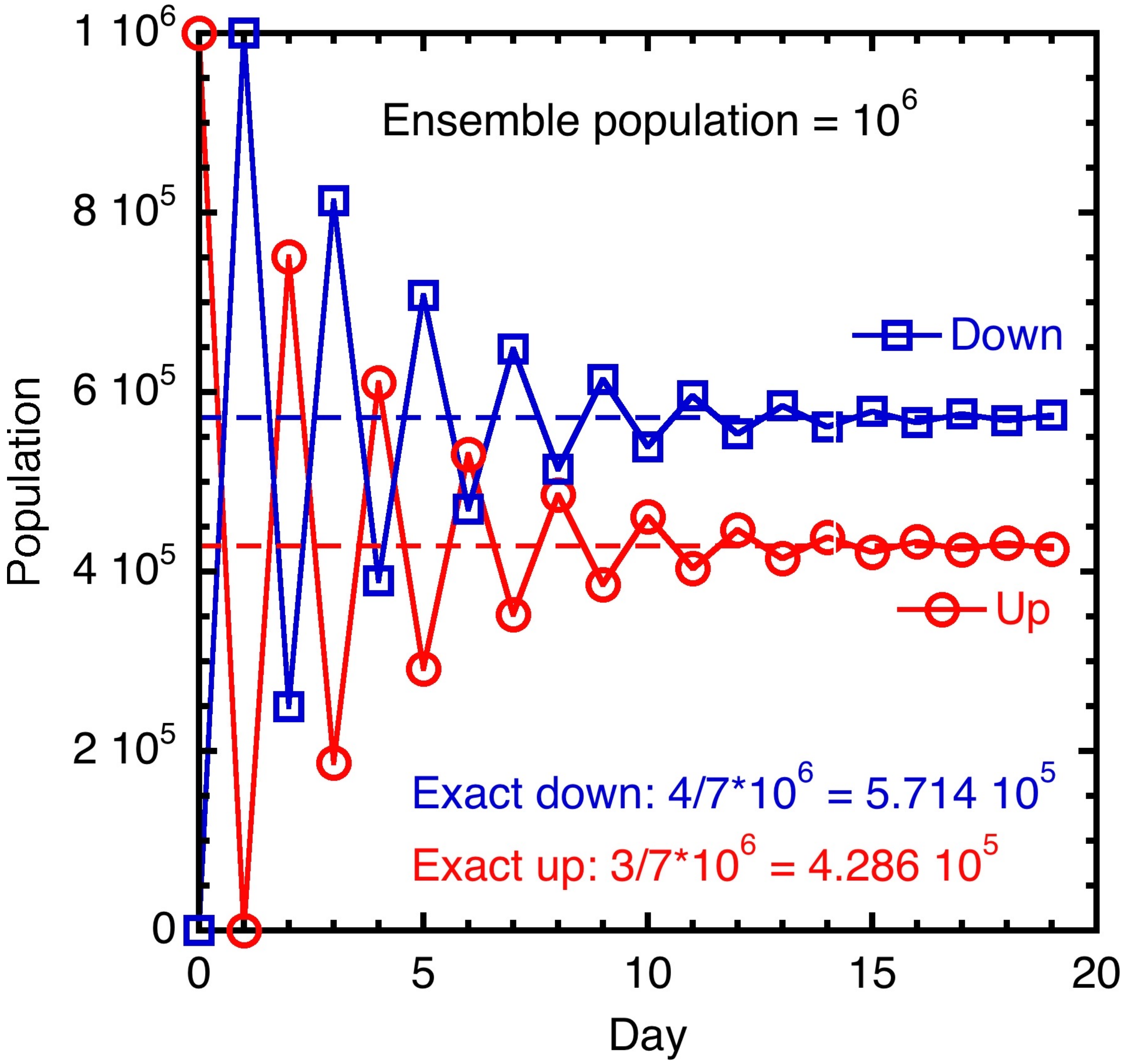
$$= \begin{pmatrix} 1 - 1 & 1 \bullet 3/4 \\ 1 & 1 - 1 \bullet 3/4 \end{pmatrix} = \begin{pmatrix} 0 & 3/4 \\ 1 & 1/4 \end{pmatrix}$$

Q: How to represent the probability distribution?
A: An ensemble of many samples

Ensemble-Average MC

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define NTRY 20  // # of MC trials
#define NENS 1000000  // ensemble size
#define TRNS 3.0/4.0  // up-to-down conditional probability

int main() {
  int s;  // spin state: 0 = up; 1 = down
  int hist[NTRY][2];  // histogram
  int try,walker;

  srand((unsigned)time((long *)0));
  for (try=0; try<NTRY; try++) for (s=0; s<2; s++) hist[try][s] = 0;

  for (walker=0; walker<NENS; walker++) {
    s = 0;  // up on day 0
    ++(hist[0][s]);
    for (try=1; try<NTRY; try++) {
      if (s == 0) s = 1;  // unconditional down move
      else if (rand()/(double)RAND_MAX < TRNS) s = 0;  // conditional up move
      ++(hist[try][s]);  // accumulate the average
    }
  }
  for (try=0; try<NTRY; try++) printf("%d %d %d\n",try,hist[try][0],hist[try][1]);
  return 0;
}
```

Ensemble-Average MC Result



Time-Average MC

Replace ensemble average by time average of one walker

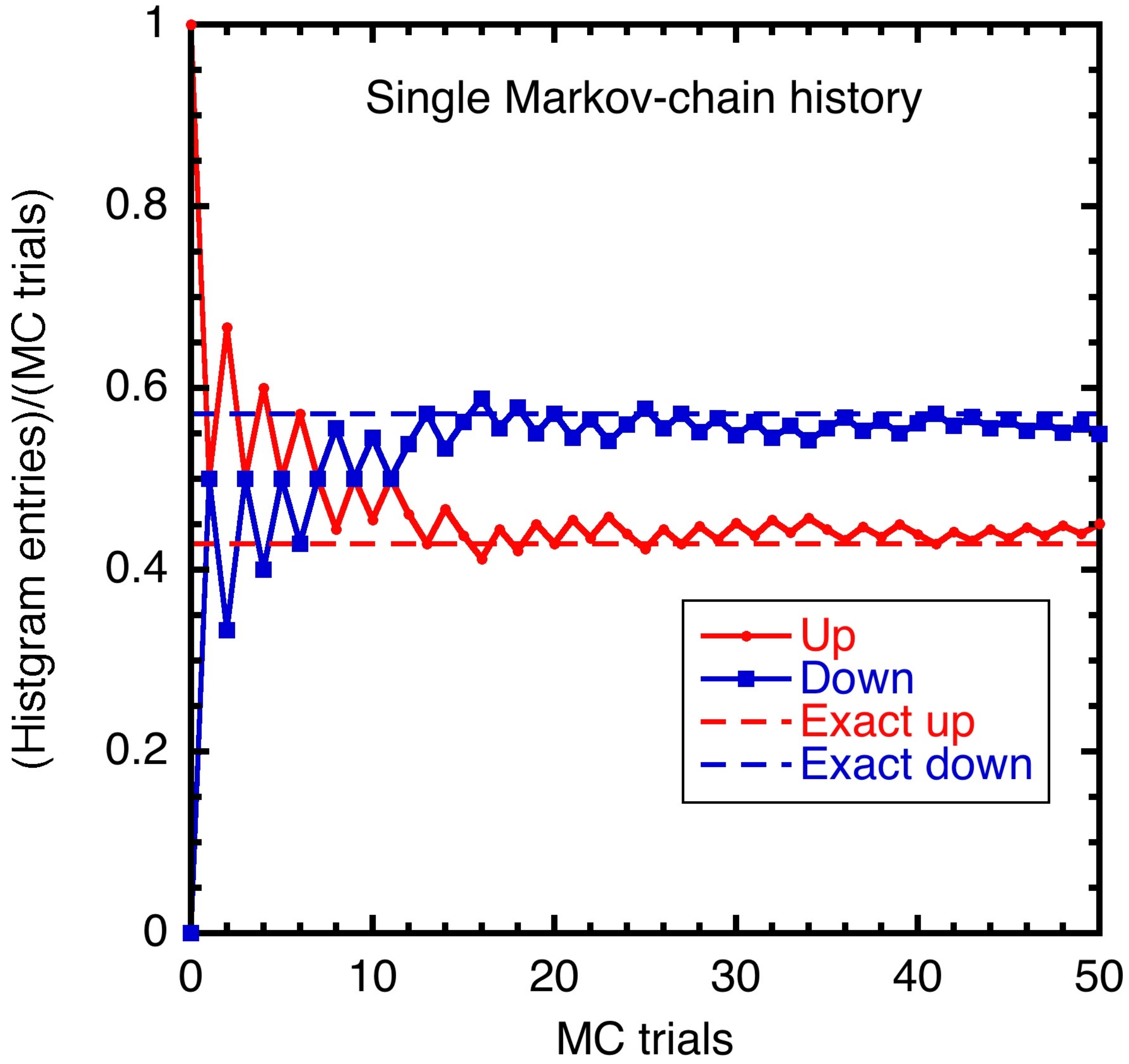
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define NTRY 1000  // ensemble size
#define TRNS 3.0/4.0  // up-to-down conditional probability

int main() {
  int s;  // spin state: 0 = up; 1 = down
  int hist[NTRY][2];  // histogram
  int try,i;

  srand((unsigned)time((long *)0));
  for (try=0; try<NTRY; try++) for (s=0; s<2; s++) hist[try][s] = 0;

  s = 0;  // up on day 0
  ++(hist[0][s]);
  for (try=1; try<NTRY; try++) {
    if (s == 0) s = 1;  // unconditional down move
    else if (rand()/(double)RAND_MAX < TRNS) s = 0;  // conditional up move
    for (i=0; i<2; i++) hist[try][i] = hist[try-1][i];
    ++(hist[try][s]);  // accumulate the average
  }

  for (try=0; try<NTRY; try++)
    printf("%d %d %d\n",try,hist[try][0],hist[try][1]);

  return 0;
}
```

Ergodic hypothesis

Time-Average MC Result

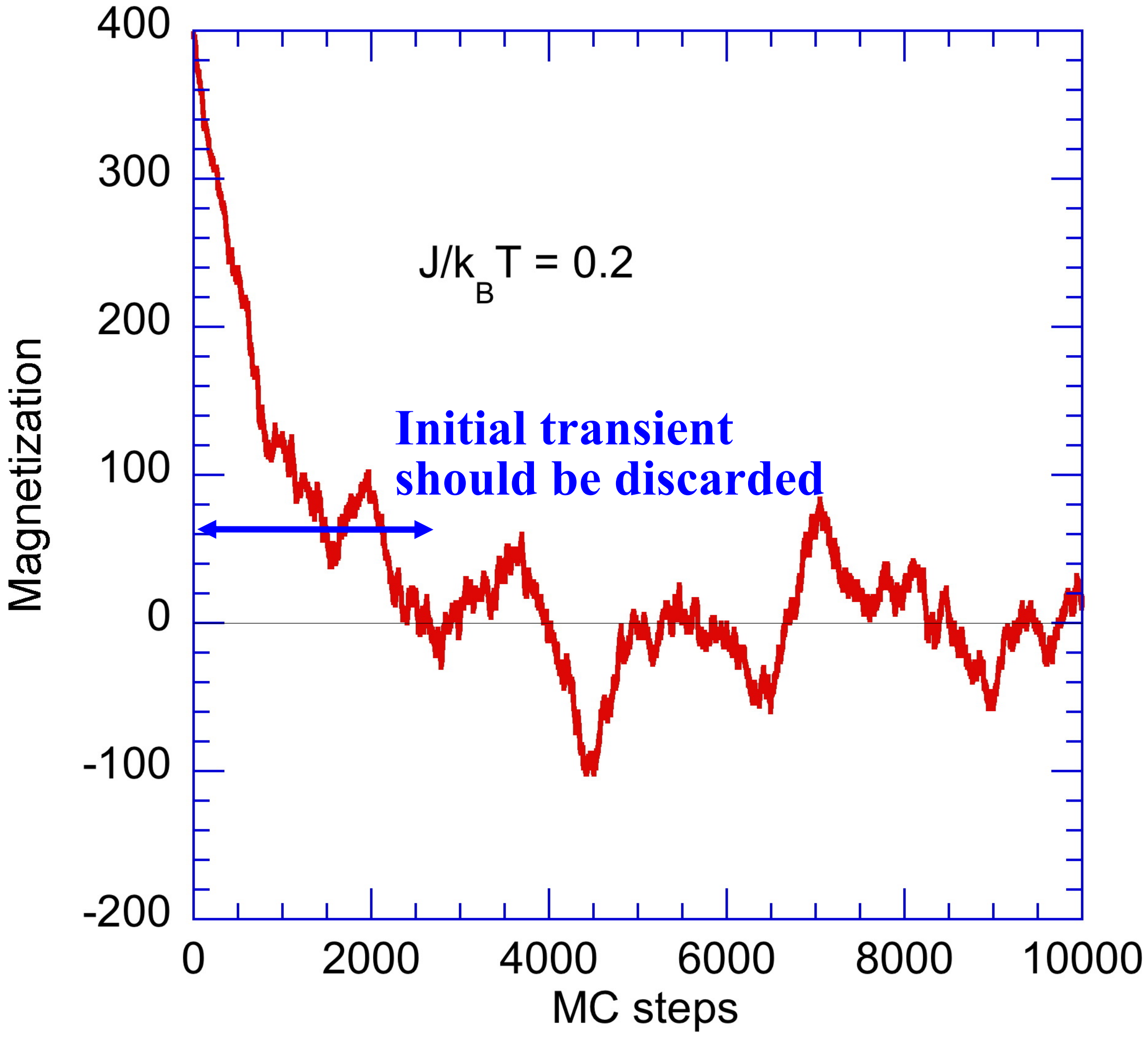
Try	Up	Down
0	1	0
1	1	1
2	2	1
3	2	2
4	3	2
5	3	3
6	4	3
7	4	4
8	4	5
9	5	5
10	5	6
11	6	6
12	6	7
13	6	8
14	7	8
15	7	9
16	7	10
17	8	10
18	8	11
19	9	11
20	9	12

Cumulative histogram



Q: Need Equilibration Steps?

A: Yes, statistics should be taken after the memory of the initial configuration is lost

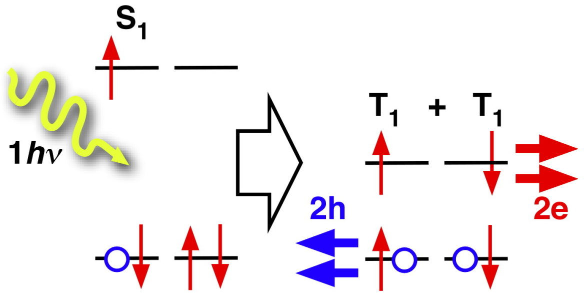


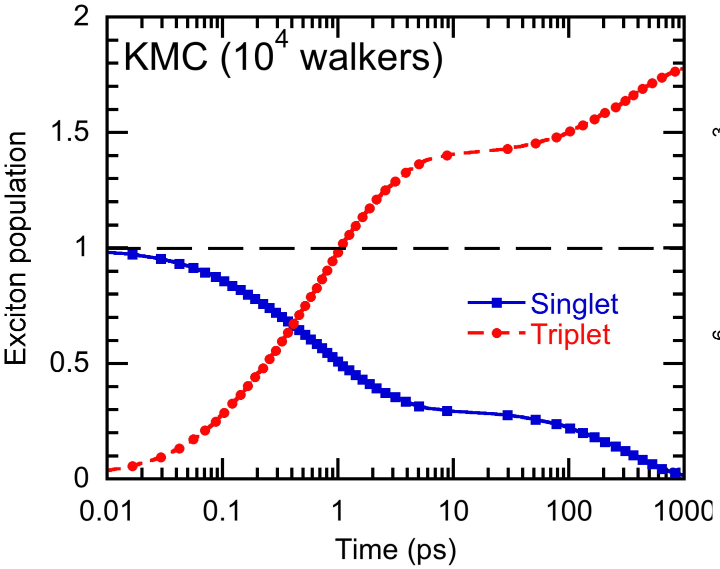
Initial transient should be discarded

Metropolis MC *vs.* Kinetic MC

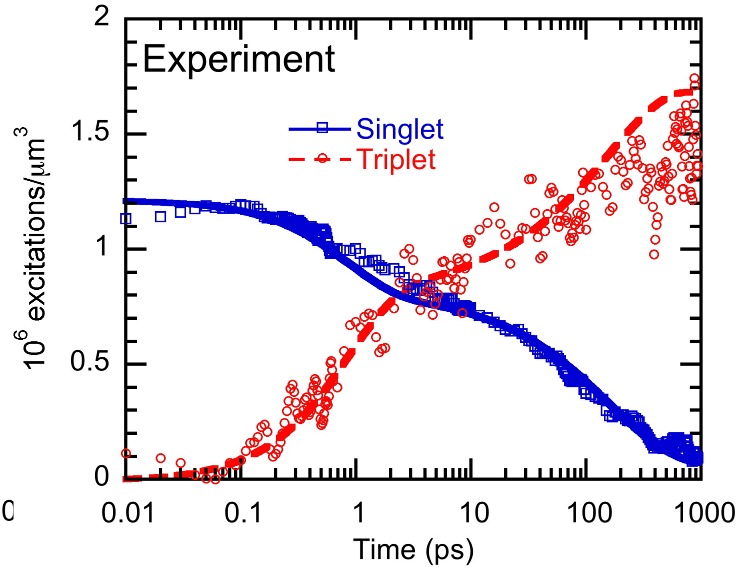
- **Metropolis MC:** Given probability density ρ_α ($\alpha = 1, \ldots, N_{\mathrm{states}}$) calculate statistical average of a physical quantity as $\langle A \rangle = \Sigma_\alpha \rho_\alpha A_\alpha$ where the transition-probability matrix $\pi_{\alpha\beta}$ is an artifact for importance sampling

- **Kinetic MC:** Given transition-rate matrix $\pi_{\alpha\beta}$ (calculated, *e.g.*, based on the transition state theory) & initial distribution $\rho_\alpha(t = 0)$, obtain the time variation of $\rho_\alpha(t)$ by solving the master equation represented by an ensemble of state samples, $d\rho_\alpha/dt = -\Sigma_\beta \pi_{\beta\alpha}\rho_\alpha + \Sigma_\beta \pi_{\alpha\beta}\rho_\beta$



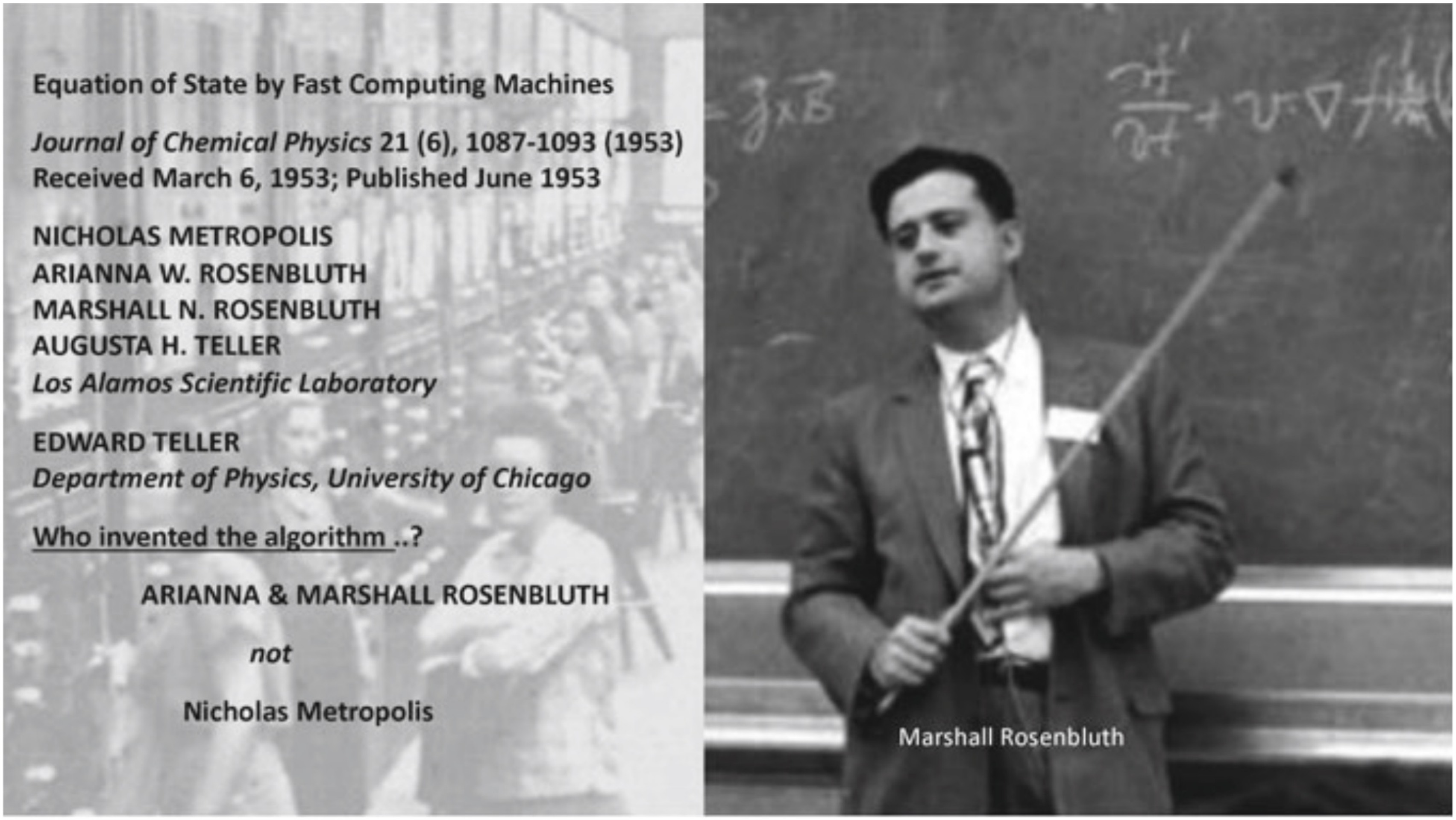


W. Mou *et al.*
APL **102,** 173301 ('13)



S. T. Roberts *et al.*
JACS **134**, 6388 ('12)

Metropolis Algorithm?



Equation of State by Fast Computing Machines

Journal of Chemical Physics 21 (6), 1087-1093 (1953)
Received March 6, 1953; Published June 1953

NICHOLAS METROPOLIS
ARIANNA W. ROSENBLUTH
MARSHALL N. ROSENBLUTH
AUGUSTA H. TELLER
Los Alamos Scientific Laboratory

EDWARD TELLER
Department of Physics, University of Chicago

Who invented the algorithm ..?

ARIANNA & MARSHALL ROSENBLUTH

not

Nicholas Metropolis

Marshall Rosenbluth

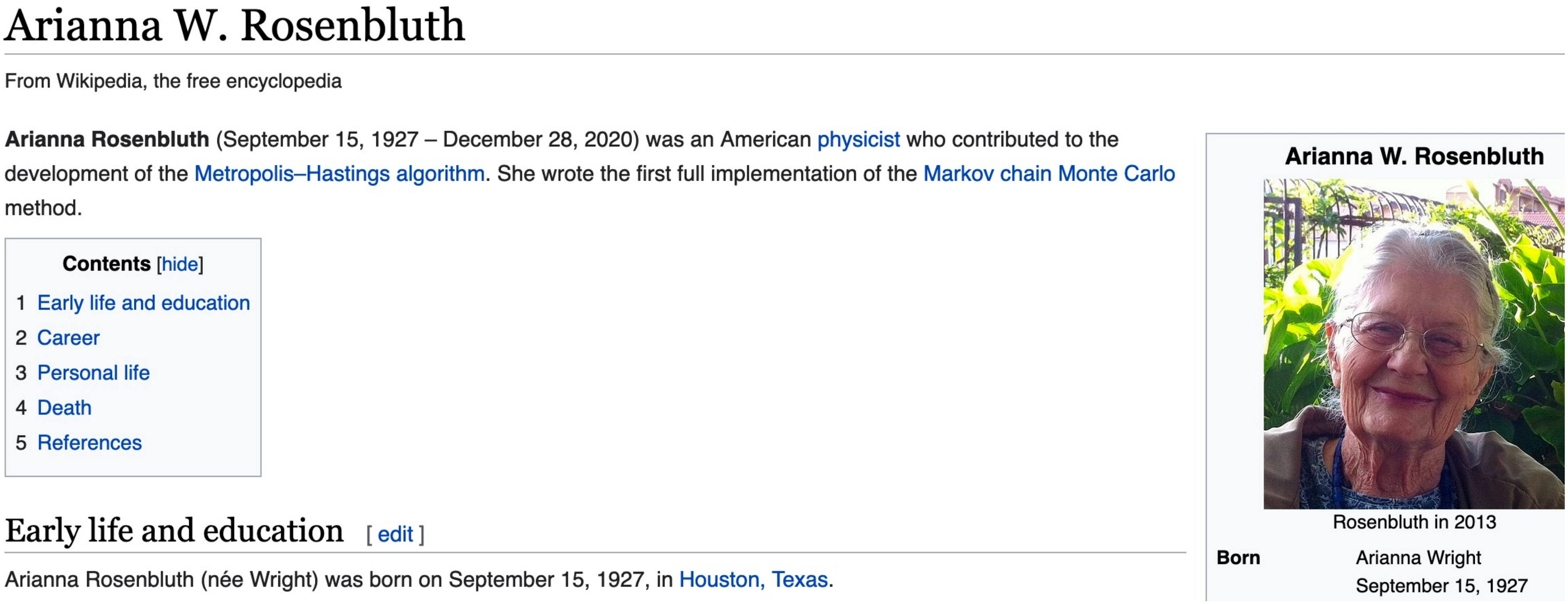
https://www.youtube.com/watch?v=MaSGXhsMk1U

A slide taken from a recent presentation by Michael Klein, giving proper credit to the creators of the “Metropolis algorithm” (M. Klein)

https://aiichironakano.github.io/phys516/Battimelli-ComputerMeetsPhysics-Springer20.pdf, p. 29

See N. Metropolis *et al*., *J. Chem. Phys*. **21**, 1087 (’53)

RIP Arianna



Arianna W. Rosenbluth

From Wikipedia, the free encyclopedia

Arianna Rosenbluth (September 15, 1927 – December 28, 2020) was an American physicist who contributed to the development of the Metropolis–Hastings algorithm. She wrote the first full implementation of the Markov chain Monte Carlo method.

Contents [hide]

Arianna W. Rosenbluth

Rosenbluth in 2013

Born	Arianna Wright September 15, 1927

Early life and education [edit]

Arianna Rosenbluth (née Wright) was born on September 15, 1927, in Houston, Texas.

Death [edit]

Arianna died on December 28, 2020 in the greater Los Angeles, California area.

https://en.wikipedia.org/wiki/Arianna_W._Rosenbluth

See APS New article:

https://aps.org/publications/apsnews/202203/history.cfm

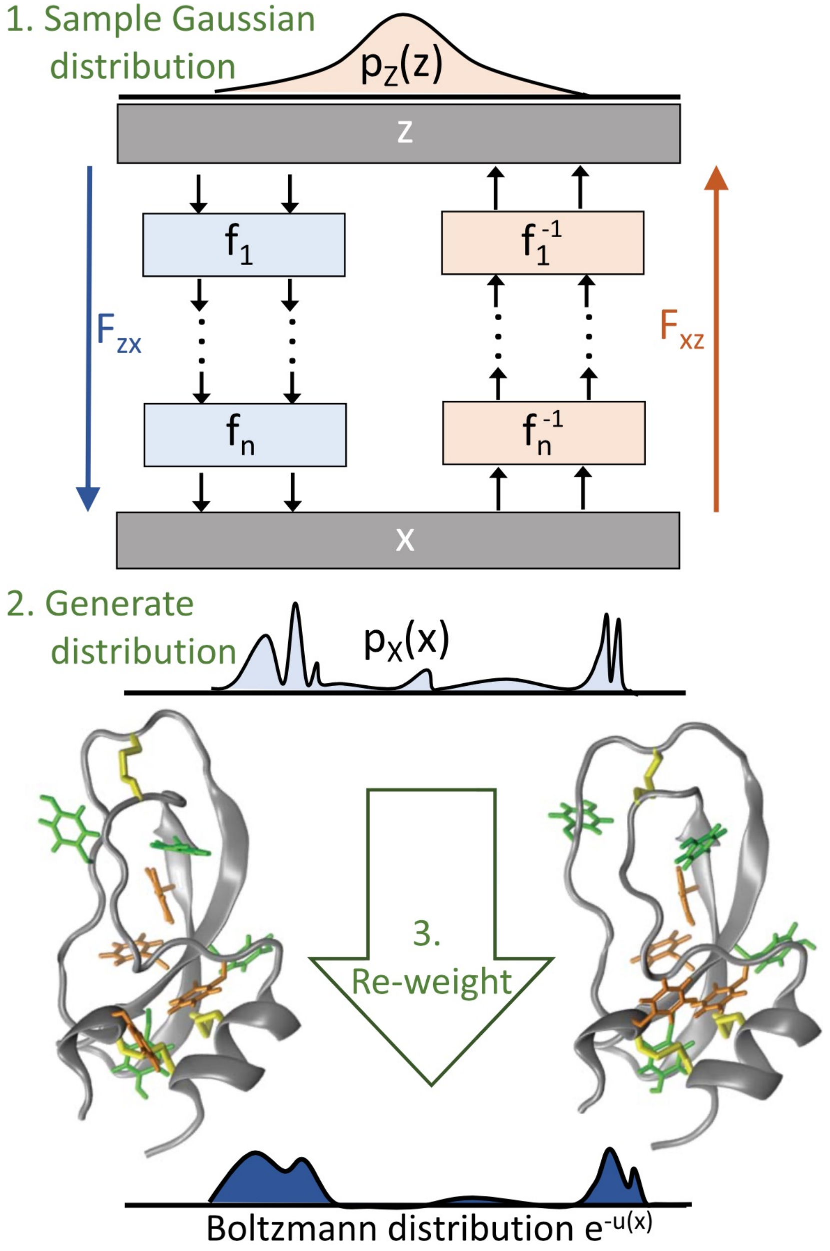
And, a brief history:

https://aiichironakano.github.io/phys516/Georgescu-EarlyMC-NRevPhys23.pdf

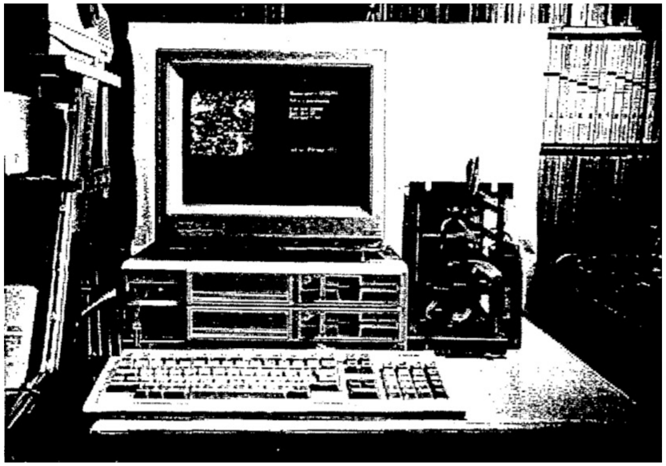
Coordinate Transformation?

- **Box-Muller algorithm:** For a harmonic oscillator, $u(x) = Kx^2/2$, Boltzmann probability density (which is Gaussian $p(x) \propto \exp(-u(x)/k_\mathrm{B}T) = \exp(-Kx^2/2k_\mathrm{B}T)$ can be generated by coordinate transformation
- **Boltzmann generator:** Machine learning of coordinate transformation such that the probability density is Gaussian in the transformed coordinate system, $z(x)$, for complex, multidimensional $u(x)$

F. Noe *et al.*
Science **365,** 1001 ('19)



Ising Machine



1bit の世界の専用計算機
——イジング・マシーン——

泰　地　真弘人
（東京大学教養学部）

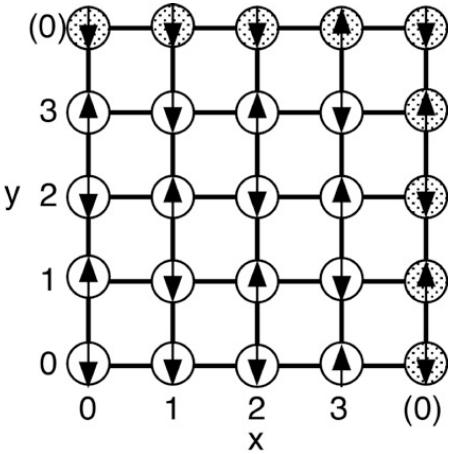
（1994年 3 月 2 日受理）

Ising Machine: A Special Purpose Computer for 1-bit Worlds

TAIJI Makoto

(Received 3 March 1994)

$$V(s^N) = -J\sum_{(k,l)} s_k s_l - H\sum_k s_k, \quad s_k = \pm 1$$



https://aiichironakano.github.io/phys516-lecture.html

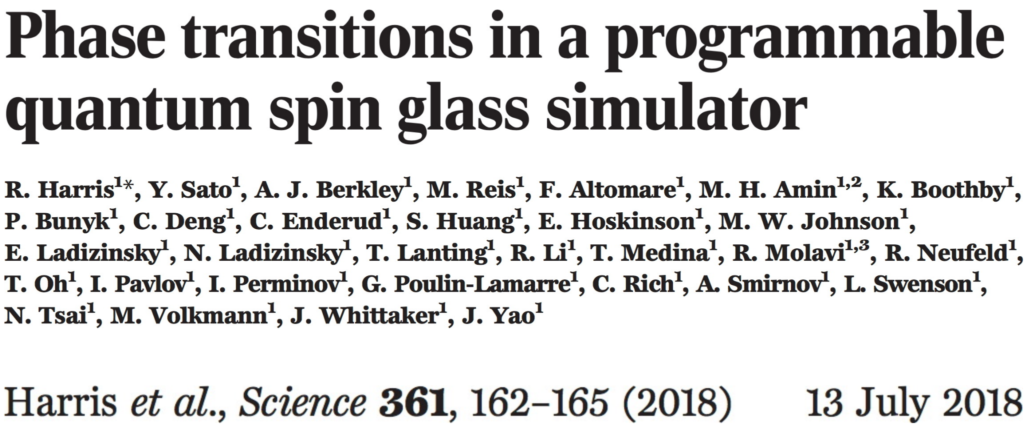
Abstract

This paper describes the development of special-purpose computer systems for Ising models, "Ising Machine" m-TIS 1 and 2. The first two sections explain Ising models and their Monte Carlo simulations. In section 3 and 4, I describe my motivation to build a special-purpose computer and the development of m-TIS 1. In section 5 and 6, the use of field-programmable gate arrays in a special-purpose computer is discussed. In the last two sections I discuss the potential abilities and future prospects of both Ising machine and a special-purpose computer in general.

J. Plasma Fusion Res. **70**, 332 ('94)

USC Quantum Computation Center

- D-Wave 2X system with 1,098-quantum bits (qubits)

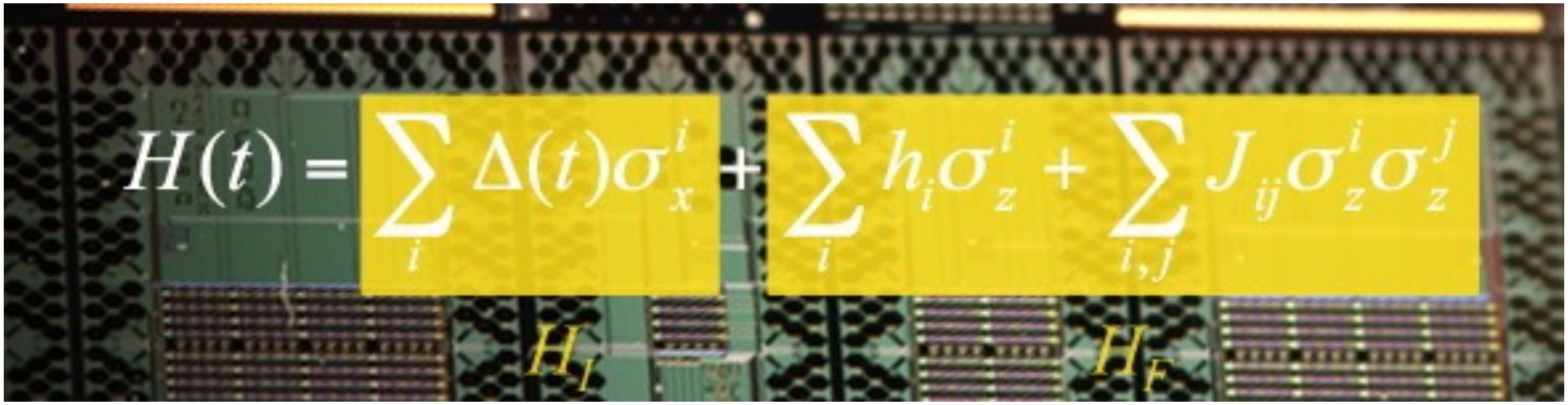


Phase transitions in a programmable quantum spin glass simulator

R. Harris[1*], Y. Sato[1], A. J. Berkley[1], M. Reis[1], F. Altomare[1], M. H. Amin[1,2], K. Boothby[1], P. Bunyk[1], C. Deng[1], C. Enderud[1], S. Huang[1], E. Hoskinson[1], M. W. Johnson[1], E. Ladizinsky[1], N. Ladizinsky[1], T. Lanting[1], R. Li[1], T. Medina[1], R. Molavi[1,3], R. Neufeld[1], T. Oh[1], I. Pavlov[1], I. Perminov[1], G. Poulin-Lamarre[1], C. Rich[1], A. Smirnov[1], L. Swenson[1], N. Tsai[1], M. Volkmann[1], J. Whittaker[1], J. Yao[1]

Harris *et al.*, *Science* **361**, 162–165 (2018)   13 July 2018

- Adiabatic quantum optimization



$$H(t)=\underbrace{\sum_i \Delta(t)\sigma_x^i}_{H_I}+\underbrace{\sum_i h_i\sigma_z^i+\sum_{i,j}J_{ij}\sigma_z^i\sigma_z^j}_{H_F}$$

http://www.isi.edu/research_groups/quantum_computing/home

Machine Learning on D-Wave

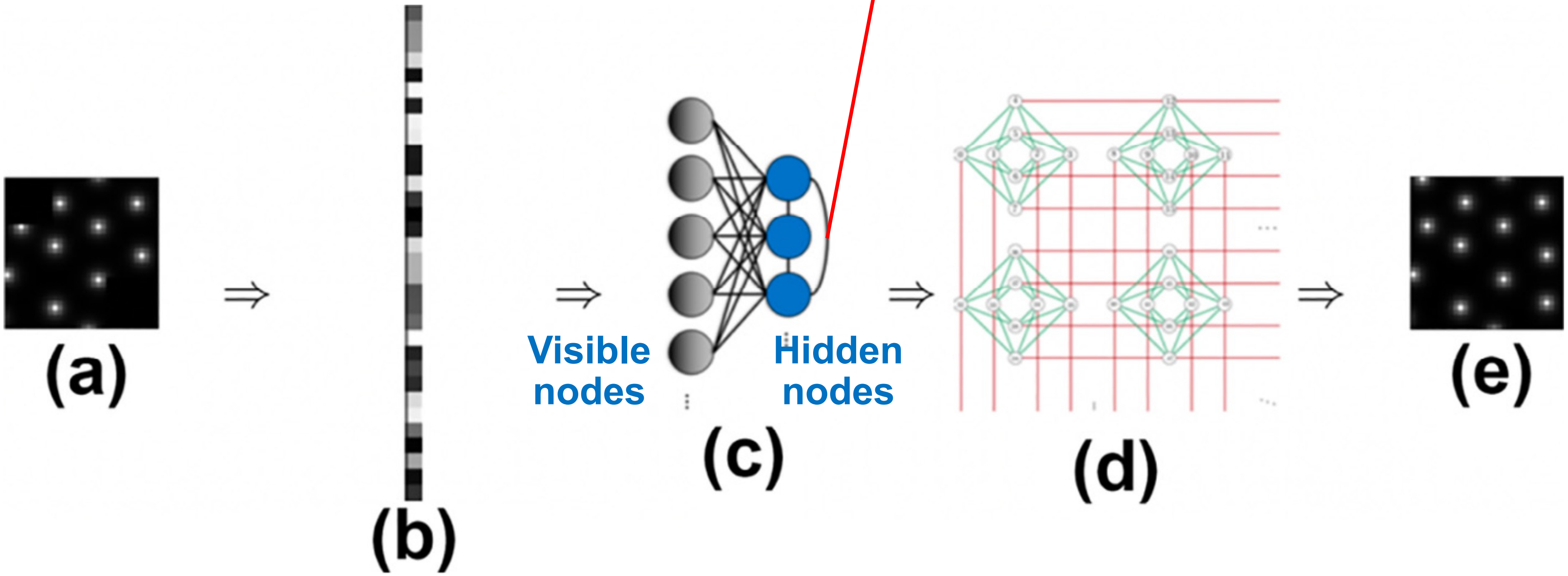
Boltzmann machine modeling of layered MoS_2 synthesis on a quantum annealer

J. Liu, A. Mohan, R. K. Kalia, A. Nakano, K. Nomura, P. Vashishta, and K.T. Yao

Comput. Mater. Sci. **173**, 109429 ('20)

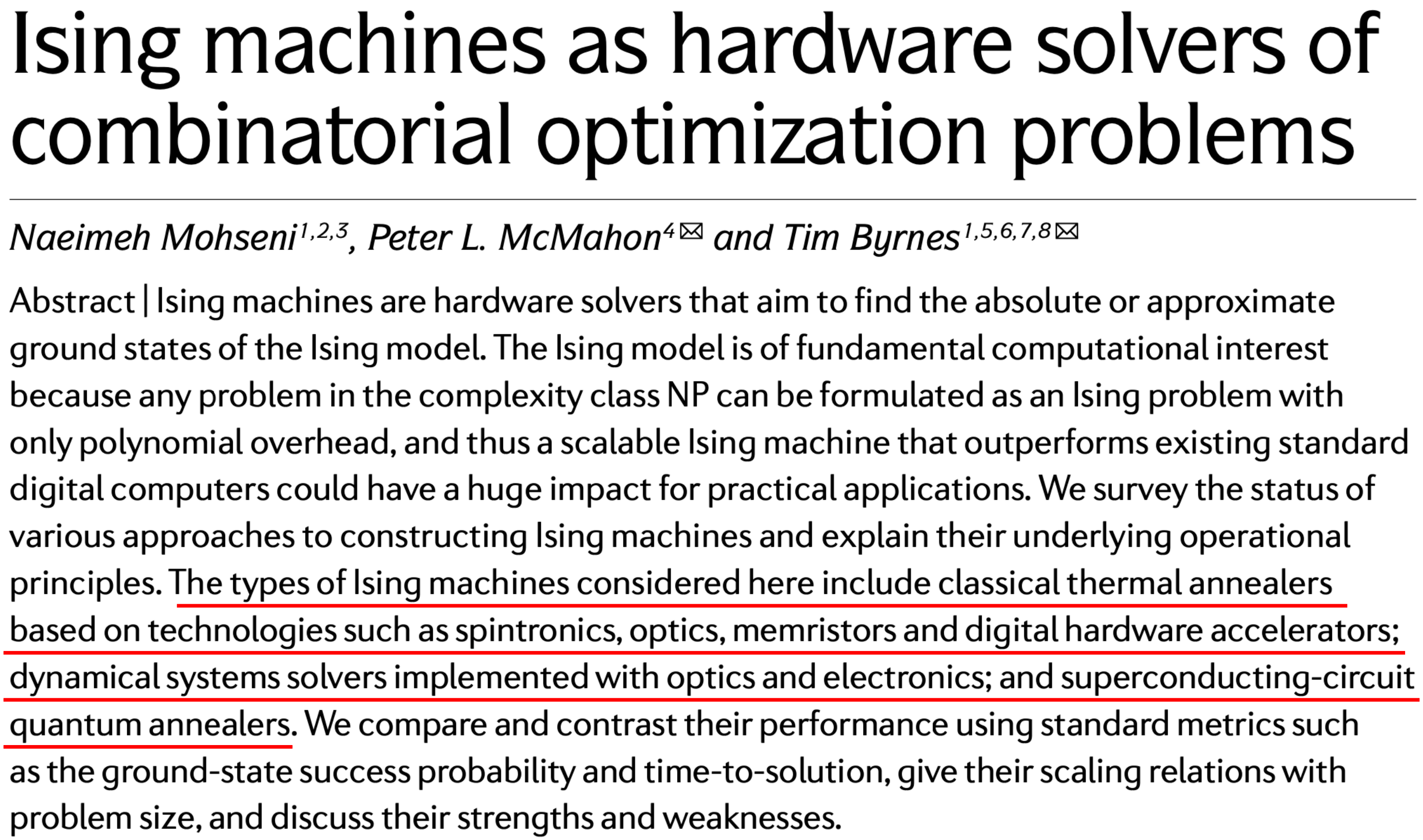
- **Computing power of D-Wave allows unrestricted* Boltzmann Machine to enhance machine learning performance**

* Nobel physics prize to Geoffrey Hinton in 2024 cited restricted Boltzmann machine



Final project by Ankith Mohan (MSCS) with Jeremy Liu (PhD-CS)

More Ising Machines



Ising machines as hardware solvers of combinatorial optimization problems

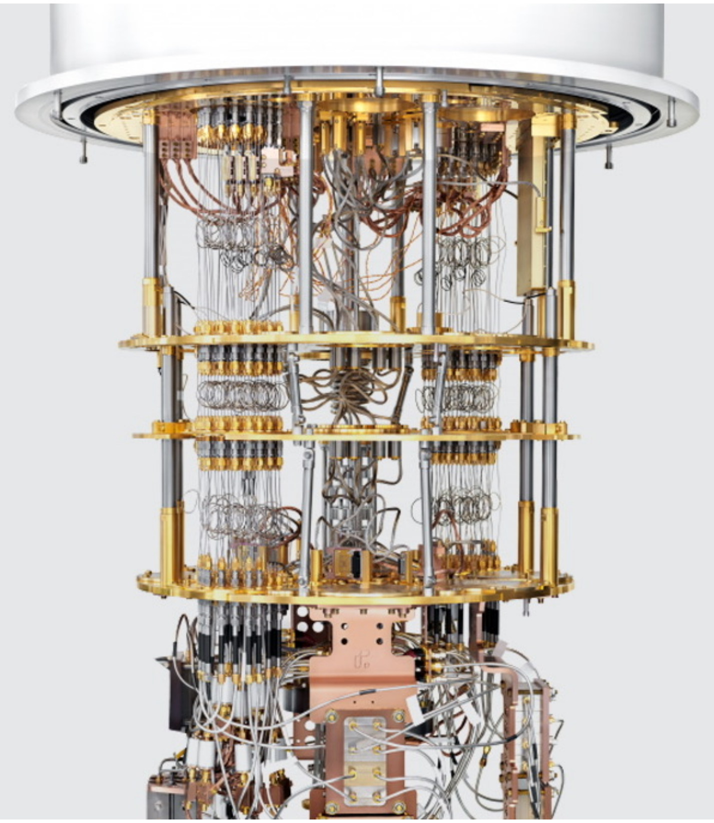
Naeimeh Mohseni[1,2,3], Peter L. McMahon[4]✉ and Tim Byrnes[1,5,6,7,8]✉

Abstract | Ising machines are hardware solvers that aim to find the absolute or approximate ground states of the Ising model. The Ising model is of fundamental computational interest because any problem in the complexity class NP can be formulated as an Ising problem with only polynomial overhead, and thus a scalable Ising machine that outperforms existing standard digital computers could have a huge impact for practical applications. We survey the status of various approaches to constructing Ising machines and explain their underlying operational principles. The types of Ising machines considered here include classical thermal annealers based on technologies such as spintronics, optics, memristors and digital hardware accelerators; dynamical systems solvers implemented with optics and electronics; and superconducting-circuit quantum annealers. We compare and contrast their performance using standard metrics such as the ground-state success probability and time-to-solution, give their scaling relations with problem size, and discuss their strengths and weaknesses.

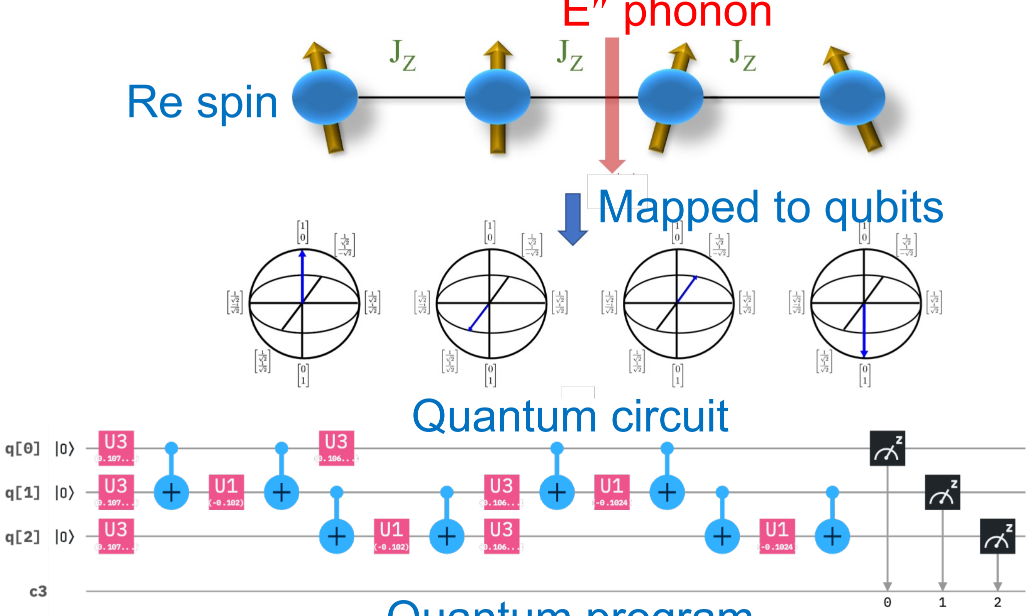
N. Mohseni *et al.*
Nat. Rev. Phys. **4,** 363 ('22)

Quantum Computing of Magnetism

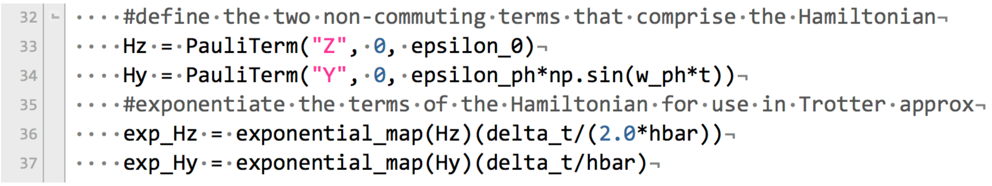
- Simulated quantum many-body (transverse-field Ising) dynamics on IBM's Q & Rigetti's Aspen quantum processors
- Electromagnetic-field control of quantum states in a chain of rhenium-magnets in MoSe$_2$ monolayer to realize desired material properties on demand, thereby pushing the envelope of "quantum materials science"



$$H(t) = -J_z \sum_{i=1}^{N-1} \boldsymbol{\sigma}_z^i \boldsymbol{\sigma}_z^{i+1} - \varepsilon_{ph}\sin(\omega_{ph}t) \sum_{i=1}^{N} \boldsymbol{\sigma}_x^i$$



Quantum program



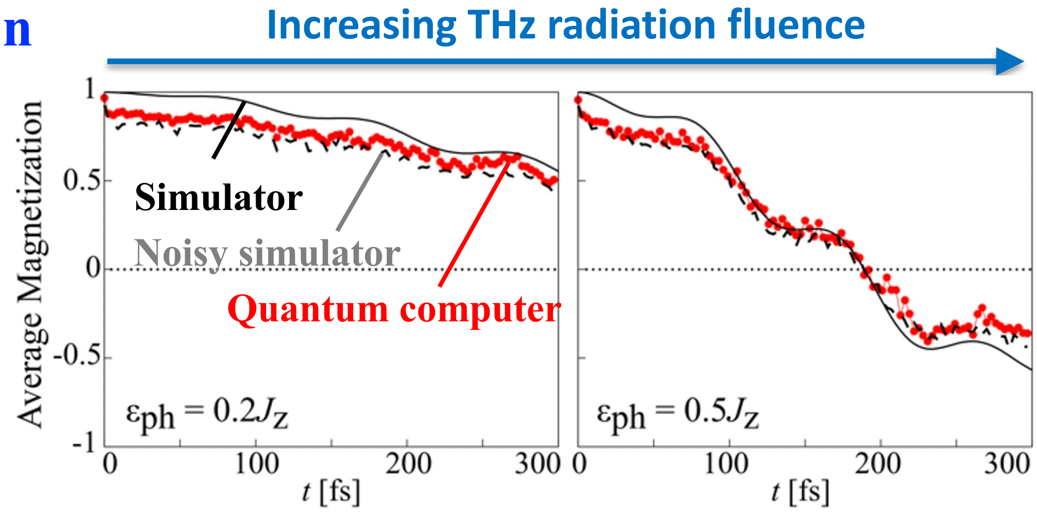
```
    #define the two non-commuting terms that comprise the Hamiltonian
    Hz = PauliTerm("Z", 0, epsilon_0)
    Hy = PauliTerm("Y", 0, epsilon_ph*np.sin(w_ph*t))
    #exponentiate the terms of the Hamiltonian for use in Trotter approx
    exp_Hz = exponential_map(Hz)(delta_t/(2.0*hbar))
    exp_Hy = exponential_map(Hy)(delta_t/hbar)
```

Do it yourself at https://quantum-computing.ibm.com

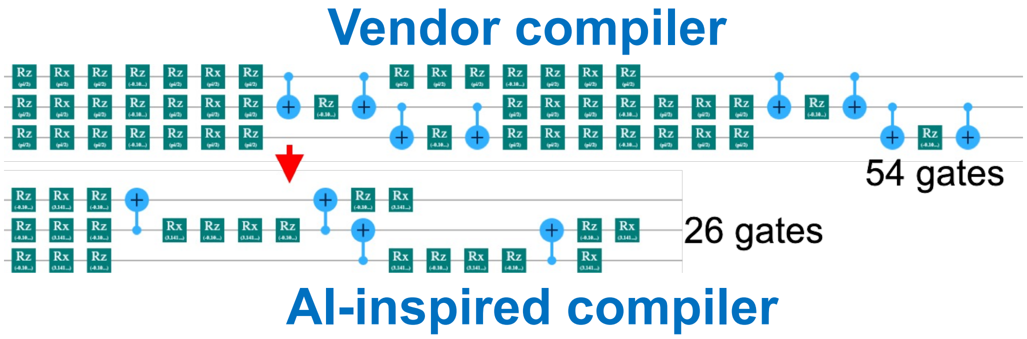
Quantum Dynamics on NISQ Computers

- **Quantum-dynamics simulations on NISQ computers show dynamic suppression of magnetization by THz radiation & symmetry protection in topological matter**

L. Bassman *et al.*,
Phys. Rev. B **101**, 184305 ('20);
M. Mercadoet *al.*,
Ibid. **110**, 075116 ('24)



- **AI-inspired quantum compiler reduced the circuit size by 30% to mitigate environmental noise**

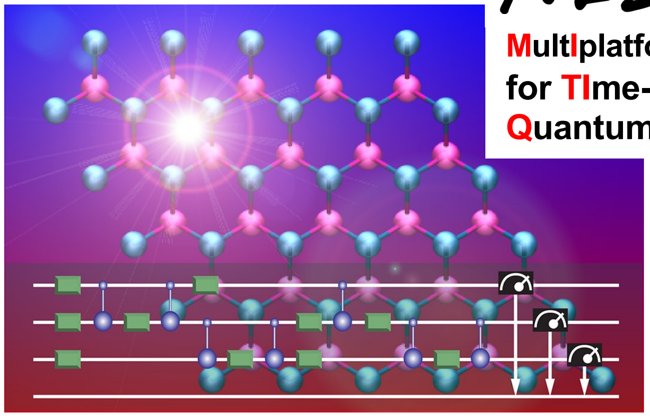


L. Bassman *et al.*,
Quantum Sci. Tech. **6**, 014007 ('21)

- **Full-stack, cross-platform software for quantum dynamics simulations on NISQ computers**

MISTIQS

MultIplatform Software
for TIme-dependent
Quantum Simulation



C. Powers *et al.*, *SoftwareX* **14**, 100696 ('21)
https://github.com/USCCACS/MISTIQS