Simulations in Statistical Physics

Course for MSc physics students

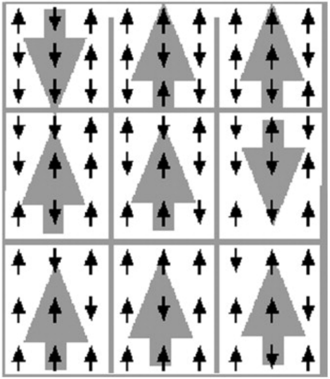
Janos Török

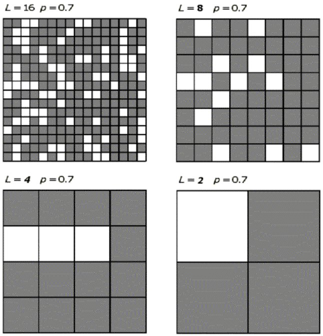
Department of Theoretical Physics

October 13, 2015

Real space numerical renormalization group

- At the critical point the system is self similar (scale-free)
- It does not matter on which scale we are looking at it.

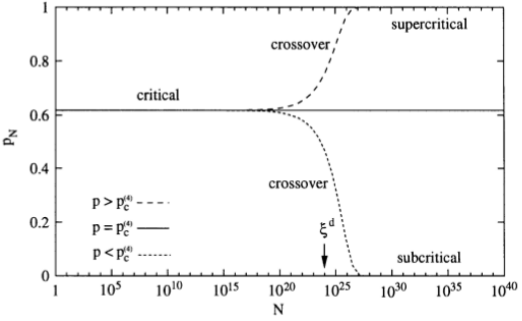




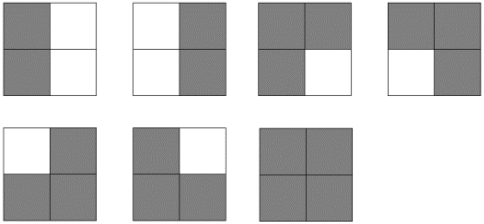
Real space numerical renormalization group

- As the system gets larger it converges into a fixed point

$$\lim_{n\to\infty} R_n(p) = \begin{cases} 0 & \text{for } 0 \leqslant p < p_c \,, \\ c & \text{for } p = p_c \,, \\ 1 & \text{for } p_c < p \leqslant 1 \end{cases}$$



Numerical renormalization group, percolation

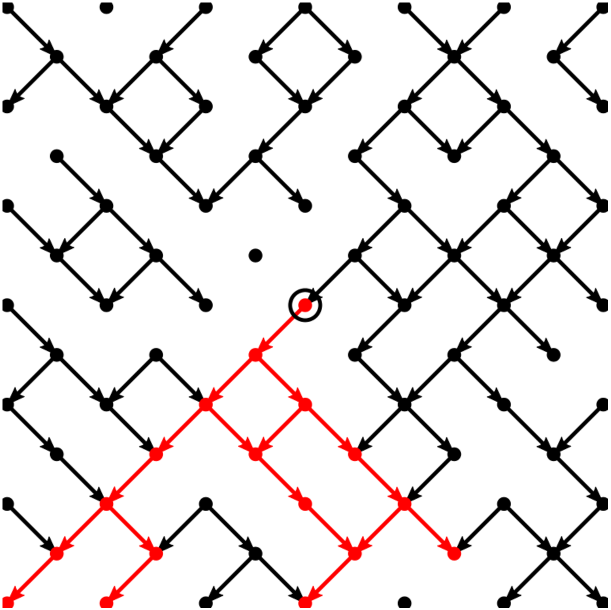


- probability that the cell is spanned:

$$p' = R(p) = 2p^2(1-p)^2 + 4p^3(1-p) + p^4$$

- In the critical point $p' = p$.
- Three solutions $p_0 = 0$, $p_1 = 1$, and $p_* = 0.6180$
- Theoretical value $p_c = 0.5927$
- Larger blocks (only numerically possible) give better estimates

Directed percolation



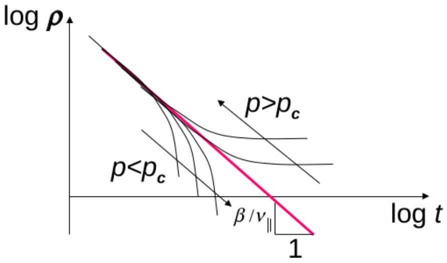
Directed percolation

- More complicated than percolation
- 3 exponents (correlation lengths in two directions) $\nu_\perp$, $\nu_{||}$ and (order parameter) β

$$\rho(\Delta p, t, L) \sim b^{-\beta/\nu_\perp} \rho(b^{1/\nu_\perp} \Delta p, t/b^z, L/b),$$

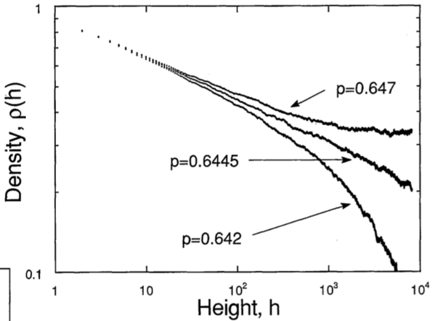
with $z = \nu_{||}/\nu_\perp$.

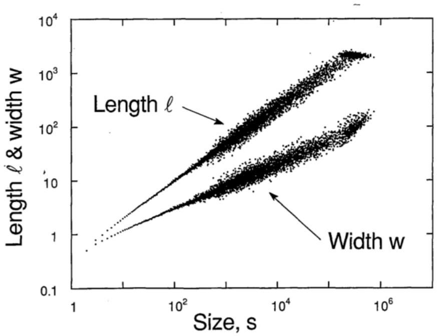
- $\beta/\nu_{||}$ as on figure
- z in a large sample
- Critical scaling of finite clusters



Directed percolation

- Density versus time





- Length/width versus size
- Clusters are fractal

Random numbers

- Why?
  - Ensemble average:

    $$\langle A \rangle = \sum_i A_i P_i^{\mathrm{eq}}$$

    Random initial configurations
  - Model: e.g. Monte-Carlo
  - Fluctuations
  - Sample
- How?

Generate random numbers

- We need good randomness:
  - Correlations of random numbers appear in the results
  - Must be fast
  - Long cycle
  - Cryptography

Random number generators

- True (Physical phenomena):
  - Shot noise (circuit)
  - Nuclear decay
  - Amplification of noise
    - Atmospheric noise (`random.org`)
    - Thermal noise of resistor
    - Reverse biased transistor
  - Limited speed
  - Needed for cryptography
- Pseudo (algorithm):
  - Deterministic
    - Good for debugging!
  - Fast
  - Can be made reliable

Language provided random numbers

It is good to know what the computer does!

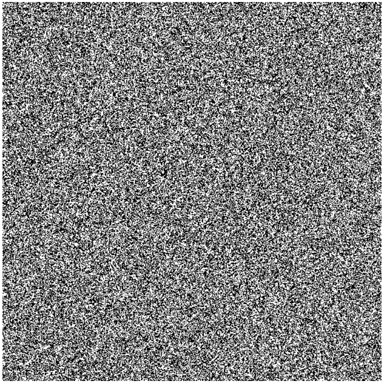
- Algorithm
  - Performance
  - Precision
  - Limit cycle
  - Historically(?) a catastrophe

DILBERT By Scott Adams
TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
NINE NINE NINE NINE NINE NINE
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.
www.dilbert.com
scottadams@aol.com
© 2001 United Feature Syndicate, Inc.

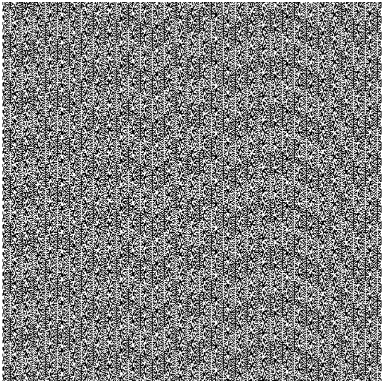
Language provided random numbers

It is good to know what the computer does!

Random



php rand() on Windows



Language provided random numbers

It is good to know what the computer does!

- Algorithm
  - Performance
  - Precision
  - Limit cycle
  - Historically a catastrophe
- Seed
  - From true random source
  - Time
  - Manual
    - Allows debugging
    - Ensures difference

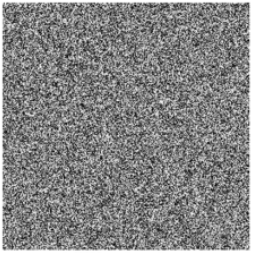
First only uniform random numbers

Seed

- From true random source
- Time
- Manual

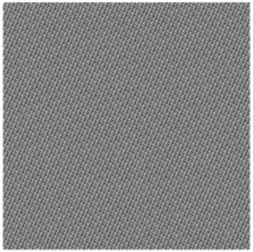
Random number generator of Python with different seeds:

System.Random
numbers 0...n of seed 0



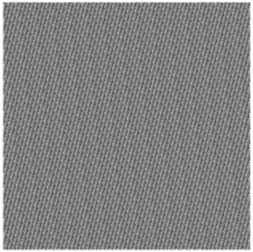
Sequence of 65536 random values.

System.Random
0th number of seed 0...n



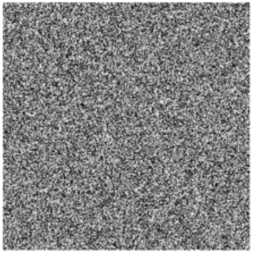
Sequence of 65536 random values.

Linear function i * 19969 / 207
numbers 0...n

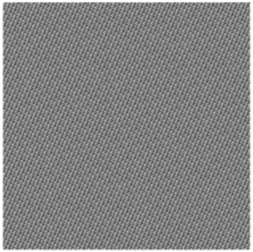


Sequence of 65536 random values.

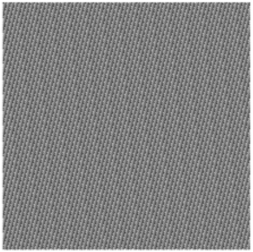
System.Random
numbers 0...n of seed 0


Sequence of 65536 random values.

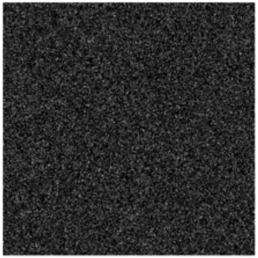
System.Random
0th number of seed 0...n


Sequence of 65536 random values.

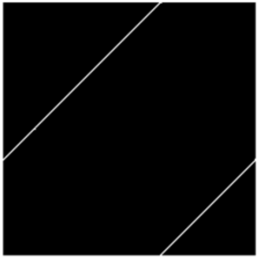
Linear function i * 19969 / 207
numbers 0...n


Sequence of 65536 random values.

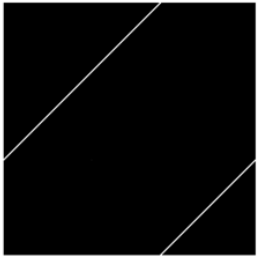
System.Random
numbers 0...n of seed 0


Plot of 500000 random coordinates.

System.Random
0th number of seed 0...n

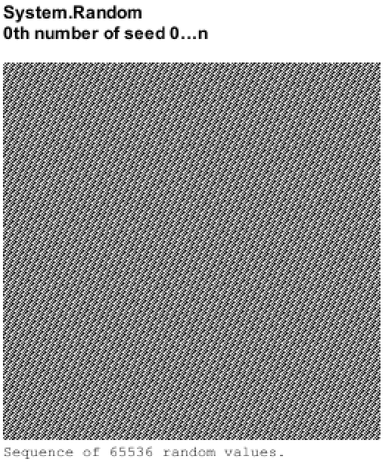

Plot of 500000 random coordinates.

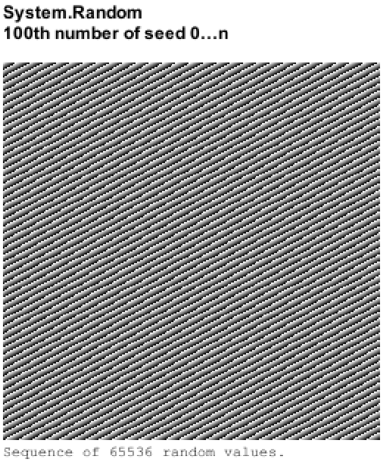
Linear function i * 19969 / 207
numbers 0...n


Plot of 500000 random coordinates.

Seed

- Ensemble average: Include in the code if possible instead of restarting it with different seeds!





Multiplicative congruential algorithm

- Let r_j be an integer number, the next is generated by

$$r_{j+1} = (ar_j + c)\mathrm{mod}(m),$$

- Sometimes only k bits are used
- Values between 0 and $m-1$ or $2^k - 1$
- Three parameters (a, c, m).
- If $m = 2^X$ is fast. Use AND (&) instead of modulo (%).
- Good:
  - Historical choice:
    $a = 7^5 = 16807,\ m = 2^{31} - 1 = 2147483647,\ c = 0$
  - gcc built-in ($k = 31$):
    $a = 1103515245,\ m = 2^{31} = 2147483648,\ c = 12345$
- Bad:
  - RANDU: $a = 65539,\ m = 2^{31} = 2147483648,\ c = 0$

Tausworth, Kirkpatrick-Stoll generator

- Fill an array of 256 integers with random numbers

$$J[k] = J[(k-250)\&255]\hat{\ }J[(k-103)\&255]$$

- Return $J[k]$, increase k by one

- Can be 64 bit number
- Extremely fast, but short cycles for certain seeds

XOR function

^	1	0
1	0	1
0	1	0

Tausworth, Kirkpatrick-Stoll generator corrected by Zipf

The one the lecturer uses

- Fill an array of 256 integers with random numbers

$$J[k] = J[(k-250)\&255]\hat{}J[(k-103)\&255]$$

  Increase k by one

$$J[k] = J[(k-30)\&255]\hat{}J[(k-127)\&255]$$

- Return $J[k]$, increase k by one
- Extremely fast, reliable also on bit level

General transformation $x \in [0:1[$

$$x = r/(RAND\_MAX + 1)$$

Tests

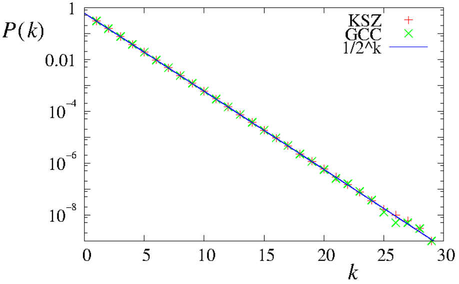
- General: e.g. TESTU01
- Diehard tests:
  - Birthday spacings (spacing is exponential)
  - Monkey tests (random typewriter problem)
  - Parking lot test
  - Moments: $m = \int_0^1 \frac{1}{n+1}$
  - Correlation
    $$C_{q,q'}(t) = \int_0^1 \int_0^1 x^q x'^{q'} P[x, x'(t)] dx dx' = \frac{1}{(q+1)(q'+1)}$$
  - Fourier-spectra
  - Fill of d dimensional lattice
  - Random walks

Red ones are not always fulfilled!

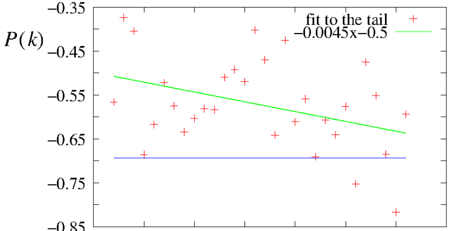
- Certain Multiplicative congruential generators are bad on bit series distribution, not completely position independent.

Bit series distribution

Probability of having k times the same bit

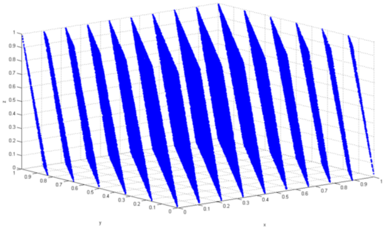


Fit to the tail for different bit positions show (gcc)



Fill of d dimensional lattice

- Generate d random numbers $c_i \in [0, L]$
- Set $x[c_1, c_2, \ldots, c_d] = 1$
- The Marsaglia effect is that for all congruential multiplicative generators there will be unavailable points (on hyperplanes) if d is large enough.
- For RANDU $d = 3$



Solution for Marsaglia effect

- Instead of d random numbers only 1 (x)
- Divide it int d parts

  ```
  c_1=x%d, x/=d
  c_2=x%d, x/=d
  ...
  ```
- Better to have $L = 2^k$.
- In this case much faster!

General advice: Save time by generating less random numbers

Random numbers with different distributions

- Let us have a good random number $r \in [0, 1]$.
- The probability density function is $P(x)$
- The cumulative distribution is

$$D(x) = \int_{-\infty}^{x} P(x')dx'$$

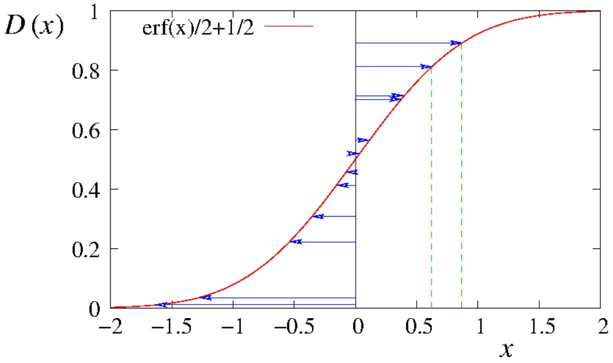
- Obviously:

$$P(x) = D'(x)$$

- The numbers $D^{-1}(x)$ will be distributed according to $P(x)$
- $D^{-1}(x)$ is the inverse function of $D(x)$ not always easy to get!

Random numbers with different distributions

Graphical representation



Box-Müller method

Gaussian distributed random numbers

$$P(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

- Generate independent uniform $r_1, r_2 \in (0, 1)$
- r_1, r_2 cannot be zero!
- Two independent normally distributed random numbers:

$$x_1 = \sqrt{-2 \log r_1} \cos 2\pi r_2$$

$$x_2 = \sqrt{-2 \log r_1} \sin 2\pi r_2$$

- It uses radial symmetry:

$$P(x, y) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} = \frac{1}{\sqrt{2\pi}} e^{-(x^2+y^2)/2}$$

Power law distributed random numbers

Let $P(y)$ have uniform distribution in $[0, 1]$. We generate $P(x)$ such as

$$P(x) = Cx^n$$

for $x \in [x_0, x_1]$.

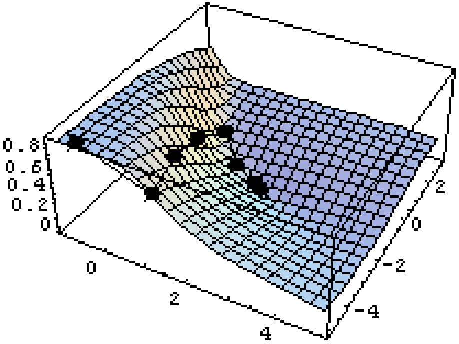
$$D(x) = \int_{x_0}^{x} P(x')dx' = \frac{C}{n+1}\left(x^{n+1} - x_0^{n+1}\right)$$

The inverse function is simple:

$$x = \left[\left(x_1^{n+1} - x_0^{n+1}\right) y + x_0^{n+1}\right]^{1/(n+1)}$$

Optimization

- General problem of finding the ground state
- Phase-space:
- Arbitrary number of dimensions
- Methods:
  - Steepest Descent
  - Stimulated Annealing
  - Genetic algorithm



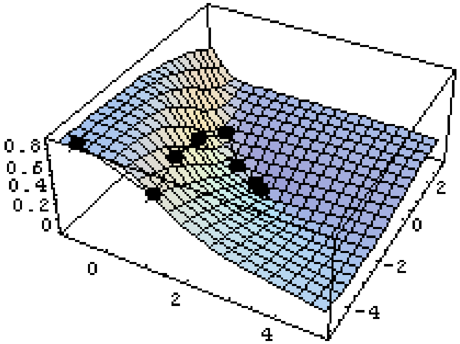
Gradient based optimization

- Given $f(\mathbf{x})$, with $\mathbf{x} = \{x_1, x_2, \dots x_n\}$
- Gradient $\nabla f(\mathbf{x}) \equiv \mathbf{g}(\mathbf{x}) = \{\partial_1 f, \partial_2 f, \dots \partial_n f\}$
- Second order partial derivatives: square symmetric matrix called the *Hessian matrix*:

$$\nabla^2 f(\mathbf{x}) \equiv H(\mathbf{x}) \equiv \begin{pmatrix} \partial_1\partial_1 f & \dots & \partial_1\partial_n f \\ \vdots & \ddots & \vdots \\ \partial_1\partial_n f & \dots & \partial_n\partial_n f \end{pmatrix}$$

General Gradient Algorithm

1. Test for convergence
2. Compute a search direction
3. Compute a step length
4. Update $\mathbf{x}$



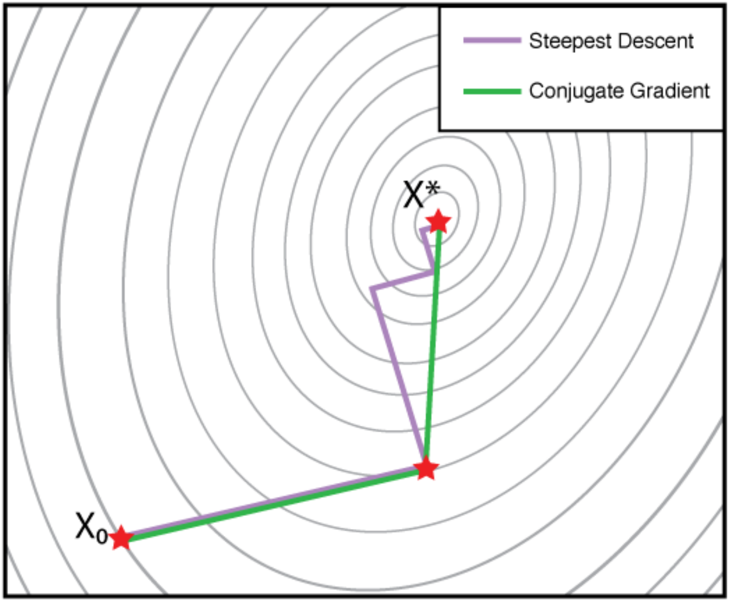
Steepest descent algorithm

1. Start from $\mathbf{x}_0$
2. Compute $\mathbf{g}(\mathbf{x}_k) \equiv \nabla f(\mathbf{x}_k)$. If $||\mathbf{g}(\mathbf{x}_k)|| \leq \varepsilon_g$ then stop, otherwise, compute normalized search direction $\mathbf{p}_k = -\mathbf{g}(\mathbf{x}_k)/||\mathbf{g}(\mathbf{x}_k)||$
3. Compute α_k such that $f(\mathbf{x}_k + \alpha \mathbf{p}_k)$ is minimized
4. New point: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}_k$
5. Test for $|f(\mathbf{x}_{k+1} - f(\mathbf{x}_k))| \leq \varepsilon_a + \varepsilon_r |f(\mathbf{x}_k)|$ and stop if fulfilled in two successive iterations, otherwise go to 2.

Conjugate Gradient Method

1. Start from $\mathbf{x}_0$
2. Compute $\mathbf{g}(\mathbf{x}_k) \equiv \nabla f(\mathbf{x}_k)$. If $||\mathbf{g}(\mathbf{x}_k)|| \leq \varepsilon_g$ then stop, otherwise Go to 5
3. Compute $\mathbf{g}(\mathbf{x}_k) \equiv \nabla f(\mathbf{x}_k)$. If $||\mathbf{g}(\mathbf{x}_k)|| \leq \varepsilon_g$ then stop, otherwise continue
4. Compute the new conjugate gradient direction $\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$, where
$$\beta = \left(\frac{||\mathbf{g}_k||}{||\mathbf{g}_{k-1}||} \right)^2$$
5. Compute α_k such that $f(\mathbf{x}_k + \alpha \mathbf{p}_k)$ is minimized
6. New point: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}_k$
7. Test for $|f(\mathbf{x}_{k+1} - f(\mathbf{x}_k))| \leq \varepsilon_a + \varepsilon_r |f(\mathbf{x}_k)|$ and stop if fulfilled in two successive iterations, otherwise go to 3.

Conjugate Gradient Algorithm



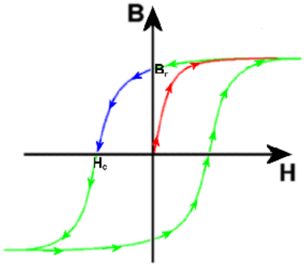
Modified Newton's method

Second order method

1. Start from $\mathbf{x}_0$
2. Compute $\mathbf{g}(\mathbf{x}_k) \equiv \nabla f(\mathbf{x}_k)$. If $||\mathbf{g}(\mathbf{x}_k)|| \leq \varepsilon_g$ then stop, otherwise, continue
3. Compute $H(\mathbf{x}_k) \equiv \nabla^2 f(\mathbf{x}_k)$ and the search direction $\mathbf{p}_k = -H^{-1}\mathbf{g}_k$
4. Compute α_k such that $f(\mathbf{x}_k + \alpha \mathbf{p}_k)$ is minimized
5. New point: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}_k$
6. Go to 2.

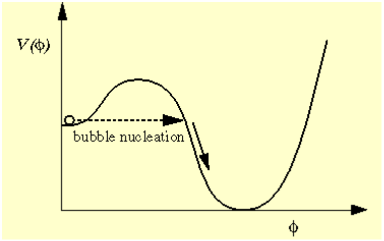
Metastability

- At first order transitions the correlation length remains finite.
- The mechanism of the first order transition is usually nucleation, which is related to metastability.
- Examples can be observed at hysteresis or undercooling, overheating, over-compessing etc.



Nucleation

- There is a competition between the bulk free energy of the droplet and its surface energy
- There is a critical nucleus size above which the transition is very rapid.
- However, such a critical nucleus has to be created by spontaneous fluctuations – which takes (sometimes enormously long) time.

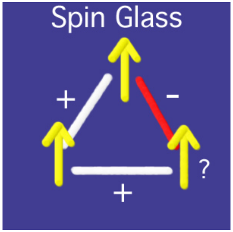


Glassy behavior, frustration

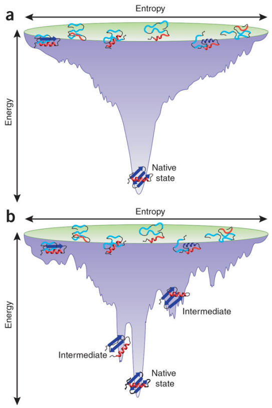
- Model glass: spin-glass:

$$H = -\frac{1}{2}\sum_{\langle i,j\rangle} J_{ij} S_i S_j$$

- where J_{ij} are random quenched variables with 0 mean (e.g. $\pm J$ with probability half)



Rugged energy landscape.


a
Entropy
Energy
Native state
b
Entropy
Energy
Intermediate
Intermediate
Native state