Advanced Monte Carlo algorithms

Bad Honnef Physics School

Computational Physics of Complex and Disordered Systemss

Werner Krauth

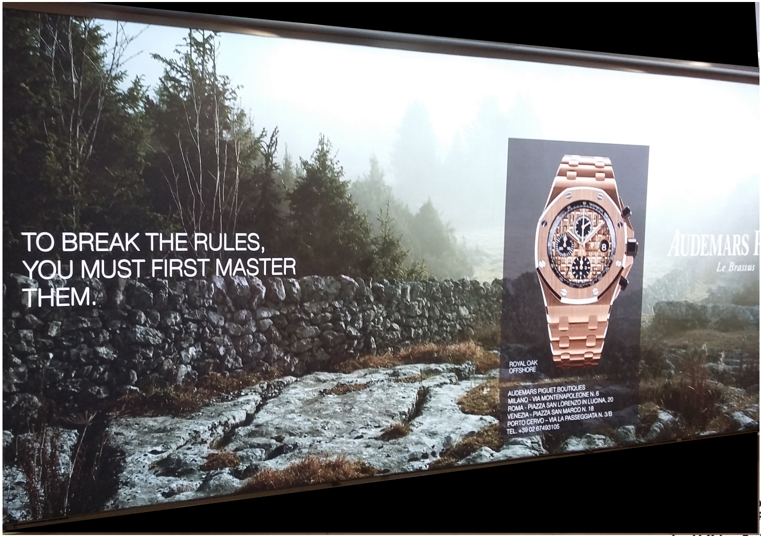
Département de Physique
Ecole Normale Supérieure, Paris, France

23 September 2015

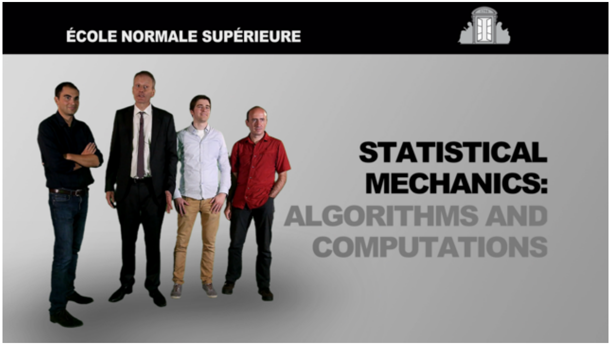
- Monte Carlo algorithms - basic notions.
- Hard disks: From detailed balance to global balance and to cluster algorithms.
- Integration and Sampling: From Gaussians to Maxwell and Boltzmann.
- Cluster algorithms for spin models: Ising, XY, Spin glasses.

Département de Physique
École normale supérieure

Breaking the rules


TO BREAK THE RULES, YOU MUST FIRST MASTER THEM.
AUDEMARS P
Le Brassus
ROYAL OAK OFFSHORE
AUDEMARS PIGUET BOUTIQUES
MILANO - VIA MONTENAPOLEONE N. 6
ROMA - PIAZZA SAN LORENZO IN LUCINA, 20
VENEZIA - PIAZZA SAN MARCO N. 18
PORTO CERVO – VIA LA PASSEGGIATA N. 3/B
TEL. +39 02 67493105

MOOC Statistical Mechanics: Algorithms and Computations



see WK Coming home from a MOOC (CSE 2015)

Département de Physique
École normale supérieure

Newton: "event-driven" molecular dynamics

- Next event at $t = 0.42813748291852523 12529\ldots$
- Molecular dynamics (Alder & Wainwright 1957)

Event-disks.py (1/2)

```
import math

def wall_time(pos_a, vel_a, sigma):
    if vel_a > 0: del_t = (1.0 - sigma - pos_a) / vel_a
    else: del_t = (pos_a - sigma) / abs(vel_a)
    return del_t

def pair_time(pos_a, vel_a, pos_b, vel_b, sigma):
    del_x = [pos_b[0] - pos_a[0], pos_b[1] - pos_a[1]]
    del_x_sq = del_x[0] ** 2 + del_x[1] ** 2
    del_v = [vel_b[0] - vel_a[0], vel_b[1] - vel_a[1]]
    del_v_sq = del_v[0] ** 2 + del_v[1] ** 2
    scal = del_v[0] * del_x[0] + del_v[1] * del_x[1]
    Upsilon = scal ** 2 - del_v_sq * ( del_x_sq - 4.0 * sigma **2)
    if Upsilon > 0 and scal < 0:
        del_t = - (scal + math.sqrt(Upsilon)) / del_v_sq
    else:
        del_t = float('inf')
    return del_t
```

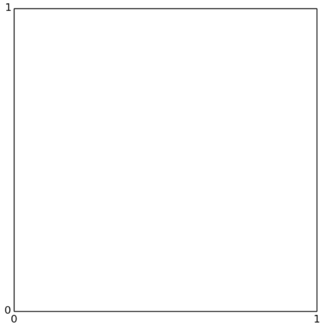
Département de Physique
École normale supérieure

Event-disks.py (2/2)

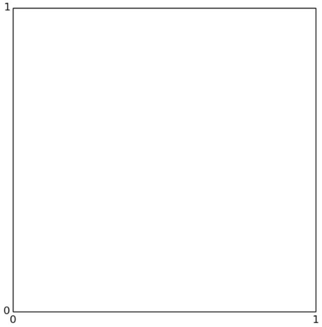
```
pos = [[0.25, 0.25], [0.75, 0.25], [0.25, 0.75], [0.75, 0.75]]
vel = [[0.21, 0.12], [0.71, 0.18], [-0.23, -0.79], [0.78, 0.1177]]
singles = [(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1), (3, 0), (3, 1)]
pairs = [(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)]
sigma = 0.2
t = 0.0
for n_event in range(4000000):
    wall_times = [wall_time(pos[k][l], vel[k][l], sigma) for k, l  in singles]
    pair_times = [pair_time(pos[k], vel[k], pos[l], vel[l], sigma) for k, l in pairs]
    next_event = min(wall_times + pair_times)
    t += next_event
    for k, l in singles: pos[k][l] += vel[k][l] * next_event
    if min(wall_times) < min(pair_times):
        collision_disk, direction = singles[wall_times.index(next_event)]
        vel[collision_disk][direction] *= -1.0
    else:
        a, b = pairs[pair_times.index(next_event)]
        del_x = [pos[b][0] - pos[a][0], pos[b][1] - pos[a][1]]
        abs_x = math.sqrt(del_x[0] ** 2 + del_x[1] ** 2)
        e_perp = [c / abs_x for c in del_x]
        del_v = [vel[b][0] - vel[a][0], vel[b][1] - vel[a][1]]
        scal = del_v[0] * e_perp[0] + del_v[1] * e_perp[1]
        for k in range(2):
            vel[a][k] += e_perp[k] * scal
            vel[b][k] -= e_perp[k] * scal
    print t, ' time'
    print pos, 'pos'
    print vel, 'vel'
```

Département de Physique
École normale supérieure

Boltzmann: Equiprobability



Boltzmann: Equiprobability

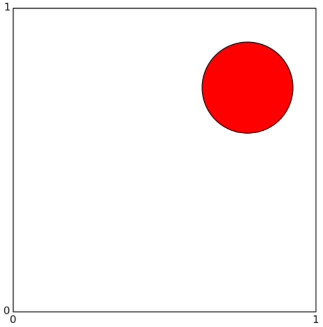


Random positions x_i, y_i $(\sigma < x_i, y_i < 1 - \sigma)$

- If illegal: Tabula rasa!
- Else: OK!

Département de Physique
École normale supérieure

Boltzmann: Equiprobability

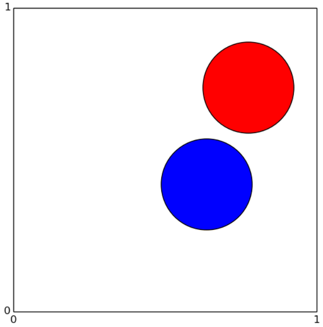


Random positions x_i, y_i ($\sigma < x_i, y_i < 1 - \sigma$)

- If illegal: Tabula rasa!
- Else: OK!

Département de Physique
École normale supérieure

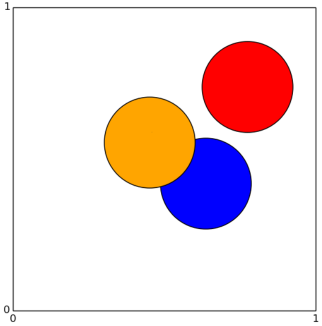
Boltzmann: Equiprobability



Random positions x_i, y_i $(\sigma < x_i, y_i < 1 - \sigma)$

- If illegal: Tabula rasa!
- Else: OK!

Boltzmann: Equiprobability

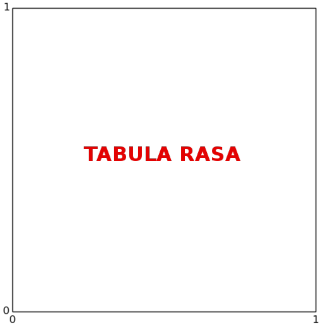


Random positions x_i, y_i $(\sigma < x_i, y_i < 1 - \sigma)$

- If illegal: Tabula rasa!
- Else: OK!

Département de Physique
École normale supérieure

Boltzmann: Equiprobability

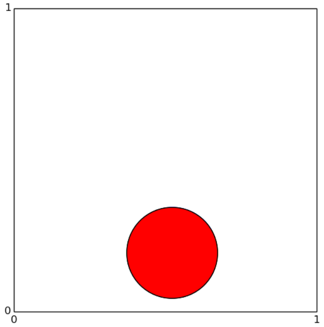


Random positions x_i, y_i $(\sigma < x_i, y_i < 1 - \sigma)$

- If illegal: Tabula rasa!
- Else: OK!

Département de Physique
École normale supérieure

Boltzmann: Equiprobability

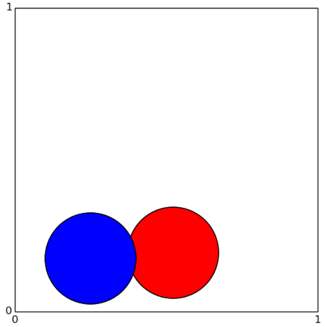


Random positions x_i, y_i $(\sigma < x_i, y_i < 1 - \sigma)$

- If illegal: Tabula rasa!
- Else: OK!

Département de Physique
École normale supérieure

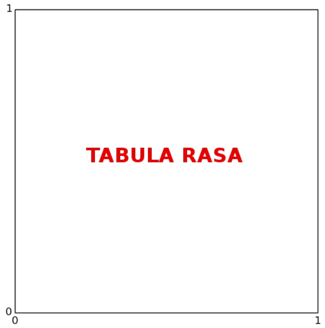
Boltzmann: Equiprobability



Random positions x_i, y_i $(\sigma < x_i, y_i < 1 - \sigma)$

- If illegal: Tabula rasa!
- Else: OK!

Boltzmann: Equiprobability

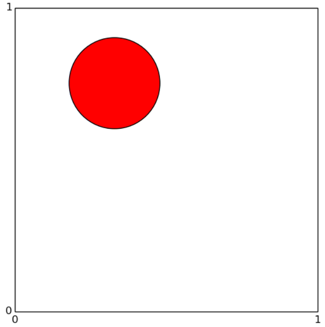


Random positions x_i, y_i $(\sigma < x_i, y_i < 1 - \sigma)$

- If illegal: Tabula rasa!
- Else: OK!

Département de Physique
École normale supérieure

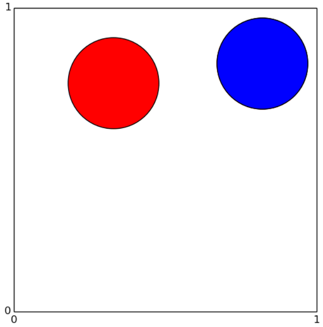
Boltzmann: Equiprobability



Random positions x_i, y_i ($\sigma < x_i, y_i < 1 - \sigma$)

- If illegal: Tabula rasa!
- Else: OK!

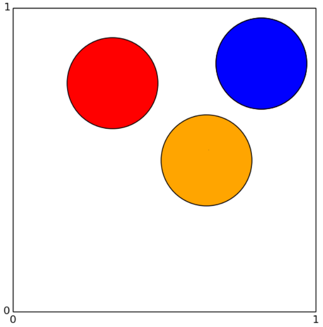
Boltzmann: Equiprobability



Random positions x_i, y_i ($\sigma < x_i, y_i < 1 - \sigma$)

- If illegal: Tabula rasa!
- Else: OK!

Boltzmann: Equiprobability

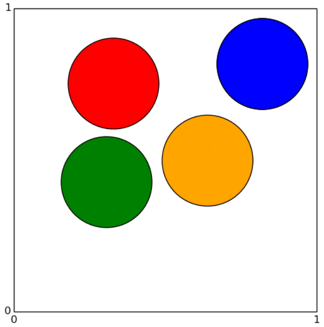


Random positions x_i, y_i ($\sigma < x_i, y_i < 1 - \sigma$)

- If illegal: Tabula rasa!
- Else: OK!

Département de Physique
École normale supérieure

Boltzmann: Equiprobability



Random positions x_i, y_i $(\sigma < x_i, y_i < 1 - \sigma)$

- If illegal: Tabula rasa!
- Else: OK!

Département de Physique
École normale supérieure

Direct-disks.py

```
import random, math

N = 4
sigma = 0.2
condition = False
while condition == False:
    L = [(random.uniform(sigma, 1.0 - sigma), random.uniform(sigma, 1.0 - sigma))]
    for k in range(1, N):
        a = (random.uniform(sigma, 1.0 - sigma), random.uniform(sigma, 1.0 - sigma))
        min_dist = min(math.sqrt((a[0] - b[0]) ** 2 + (a[1] - b[1]) ** 2) for b in L)
        if min_dist < 2.0 * sigma:
            condition = False
            break
    else:
        L.append(a)
        condition = True
print L
```

Newton vs. Boltzmann

- Equiprobability:

$$\pi^{\text{Newton}}(\mathcal{C}) = \pi^{\text{Boltzmann}}(\mathcal{C}) = \begin{cases} \text{constant} & \text{for legal } \mathcal{C} \\ 0 & \text{else} \end{cases}$$

- Rigorous proofs: Sinai (1970), Simanyi (2003).

Newton vs. Newton (Chaos)

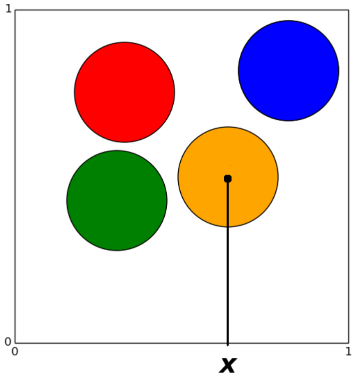
- initial position $t = 0$ of the green disks on the left

$$(x, y)_{\text{green}} = (0.25, 0.25)$$

- initial position $t = 0$ of the green disks on the right

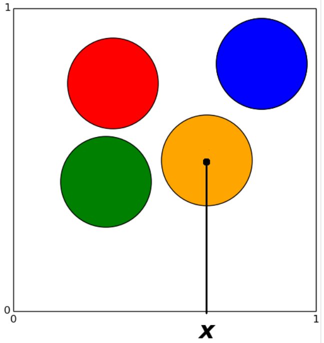
$$(x, y)_{\text{boule verte}} = (0.25000000000000006, 0.25)$$

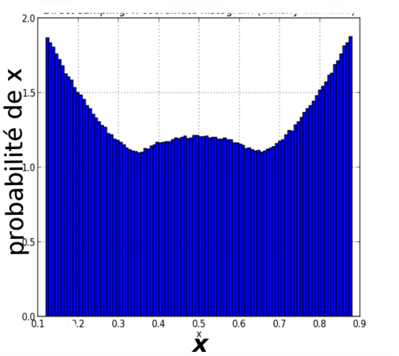
Asakura-Oosawa Depletion interaction (5th force in nature)



- The probability $\pi(x)$ is non-uniform (Asakura-Oosawa 1954).

Asakura-Oosawa Depletion interaction (5th force in nature)





- probability $\pi(x)$ non-uniform:
- Disks are attracted to the walls
- Disks are attracted to each other

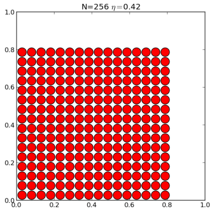
Département de Physique
École normale supérieure

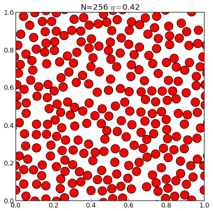
Asakura-Oosawa Depletion interaction (5th force in nature)

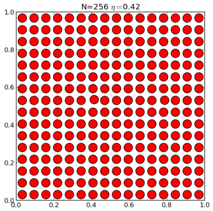
- Animations by Maxim Berman (for MOOC)

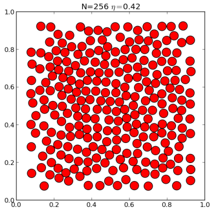
Département de Physique
École normale supérieure

Configurations at density $\eta = 0.42$





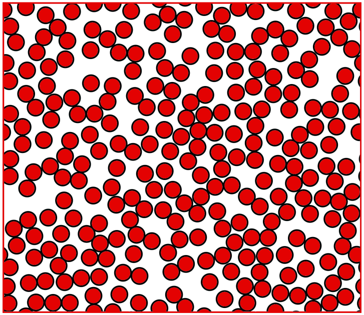


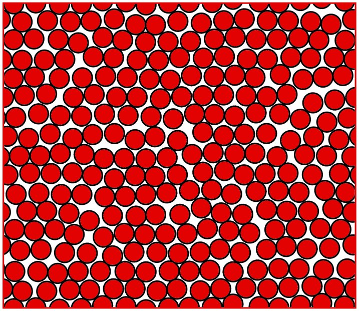


- All configurations equally likely

Département de Physique
École normale supérieure

Liquid-solid transition of hard disks


$\eta = 0.48$

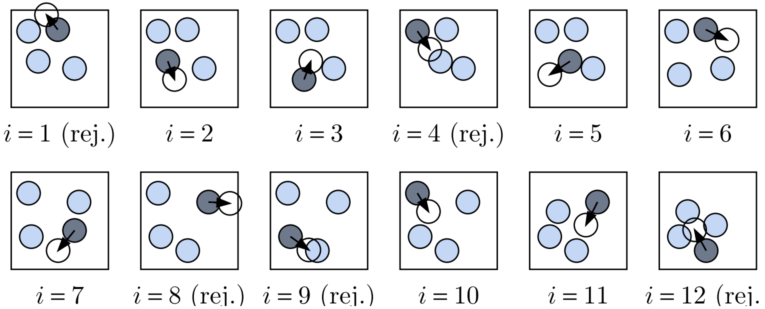

$\eta = 0.72$

- Described in 1964 using MD (Alder & Wainwright)
- Understood in 2011 (Bernard & Krauth)

Département de Physique
École normale supérieure

Markov-chain Monte Carlo (“Boltzmann”)

- Local Markov-chain Monte Carlo for hard disks:



- Metropolis et al. (1953) (see lecture 1).
- Exponential convergence (see lecture 1).

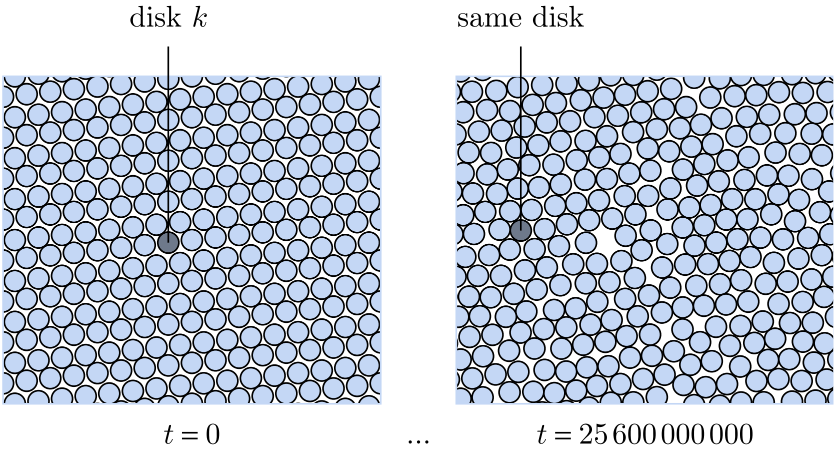
Markov-disks.py

```python
import random

L = [[0.25, 0.25], [0.75, 0.25], [0.25, 0.75], [0.75, 0.75]]
sigma = 0.15
sigma_sq = sigma ** 2
delta = 0.1
n_steps = 1000
for steps in range(n_steps):
    a = random.choice(L)
    b = [a[0] + random.uniform(-delta, delta), a[1] + random.uniform(-delta, delta)]
    min_dist = min((b[0] - c[0]) ** 2 + (b[1] - c[1]) ** 2 for c in L if c != a)
    box_cond = min(b[0], b[1]) < sigma or max(b[0], b[1]) > 1.0 - sigma
    if not (box_cond or min_dist < 4.0 * sigma ** 2):
        a[:] = b
print L
```

Metropolis et al. (1953).

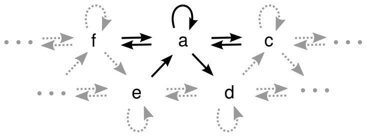
Correlation time in larger simulations



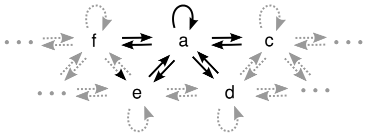
- τ exists, but it is large ($\tau \gg 25\,600\,000\,000$).

Département de Physique
École normale supérieure

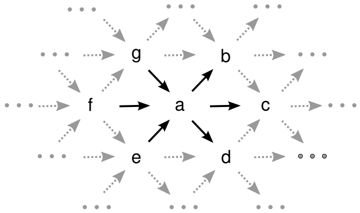
Detailed balance and global balance



global balance



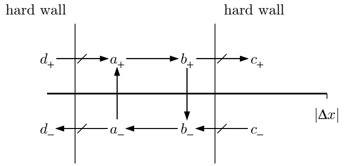
detailed balance

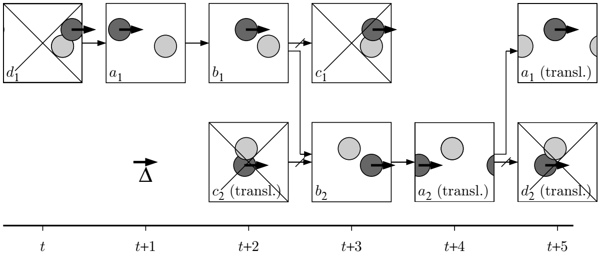


maximal global balance

Département de Physique
École normale supérieure

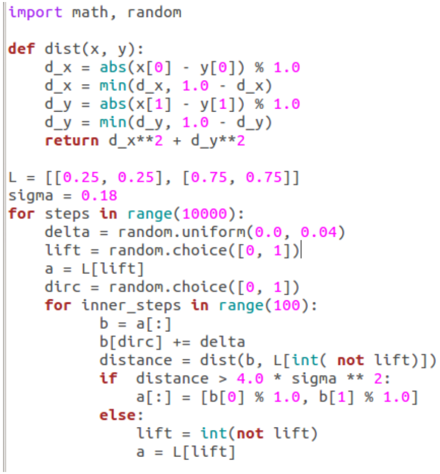
Lifting - two hard spheres





- Following: Diaconis, Holmes, Neal (2000)

Lift-two-disks-discrete.py



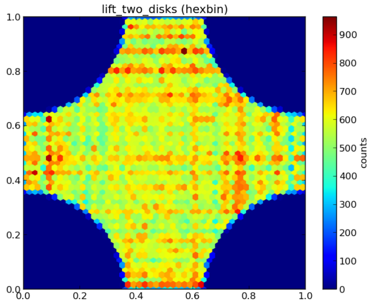
```python
import math, random

def dist(x, y):
    d_x = abs(x[0] - y[0]) % 1.0
    d_x = min(d_x, 1.0 - d_x)
    d_y = abs(x[1] - y[1]) % 1.0
    d_y = min(d_y, 1.0 - d_y)
    return d_x**2 + d_y**2

L = [[0.25, 0.25], [0.75, 0.75]]
sigma = 0.18
for steps in range(10000):
    delta = random.uniform(0.0, 0.04)
    lift = random.choice([0, 1])
    a = L[lift]
    dirc = random.choice([0, 1])
    for inner_steps in range(100):
          b = a[:]
          b[dirc] += delta
          distance = dist(b, L[int( not lift)])
          if  distance > 4.0 * sigma ** 2:
              a[:] = [b[0] % 1.0, b[1] % 1.0]
          else:
              lift = int(not lift)
              a = L[lift]
```

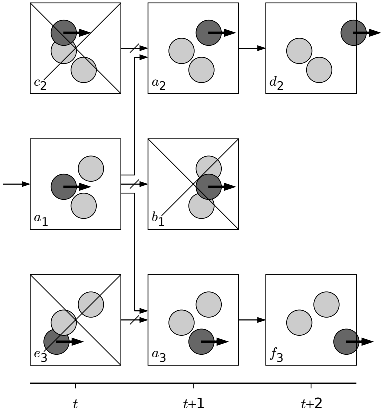
- Break detailed balance (move up or right)

Lift-two-disks-discrete.py (output)

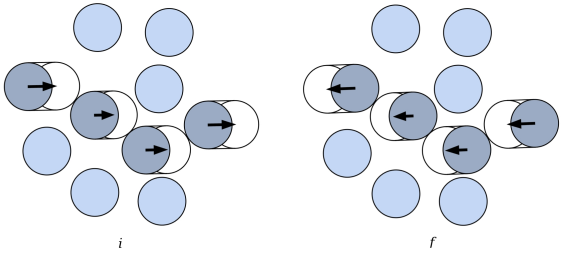


- Pair distances $(\Delta x, \Delta y)$ shown.
- Break detailed balance (move up or right)

Lifting impossible (?) - three hard spheres



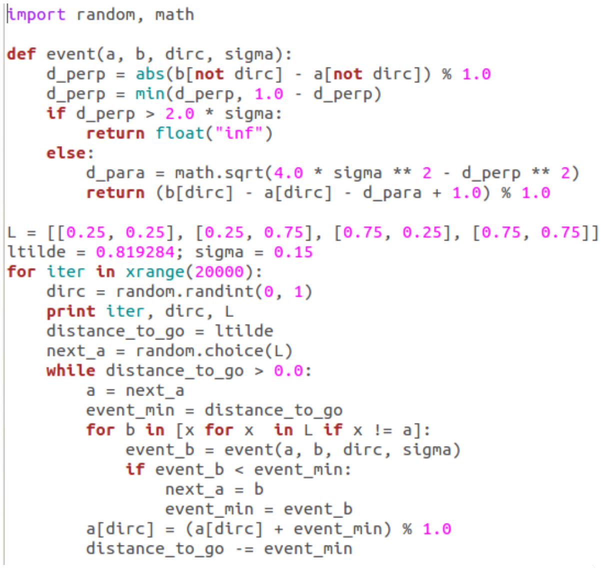
Event-chain algorithm



- Bernard, Krauth, Wilson PRE (2009) (hard-sphere potentials).
- Rejection-free, fixed total length .
- Global balance OK (!) (moving right and up).

Département de Physique
École normale supérieure

event-chain.py



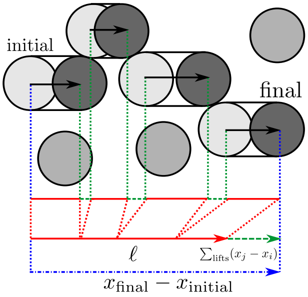
```python
import random, math

def event(a, b, dirc, sigma):
    d_perp = abs(b[not dirc] - a[not dirc]) % 1.0
    d_perp = min(d_perp, 1.0 - d_perp)
    if d_perp > 2.0 * sigma:
        return float("inf")
    else:
        d_para = math.sqrt(4.0 * sigma ** 2 - d_perp ** 2)
        return (b[dirc] - a[dirc] - d_para + 1.0) % 1.0

L = [[0.25, 0.25], [0.25, 0.75], [0.75, 0.25], [0.75, 0.75]]
ltilde = 0.819284; sigma = 0.15
for iter in xrange(20000):
    dirc = random.randint(0, 1)
    print iter, dirc, L
    distance_to_go = ltilde
    next_a = random.choice(L)
    while distance_to_go > 0.0:
        a = next_a
        event_min = distance_to_go
        for b in [x for x  in L if x != a]:
            event_b = event(a, b, dirc, sigma)
            if event_b < event_min:
                next_a = b
                event_min = event_b
        a[dirc] = (a[dirc] + event_min) % 1.0
        distance_to_go -= event_min
```

Département
de Physique
École normale
supérieure

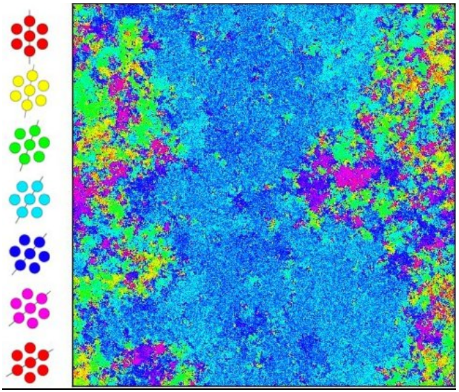
Pressure formula of Event-chain algorithm



initial
final
ℓ
$\sum_{\text{lifts}}(x_j - x_i)$
$x_{\text{final}} - x_{\text{initial}}$

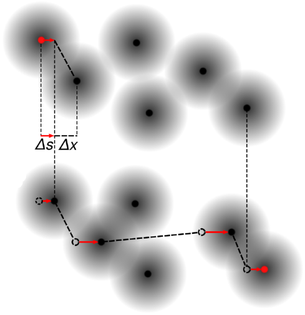
- Excess distance $\propto$ pressure.

Hard-disk configuration



- 1024^2 hard disks
- circular color code for orientational order
- Bernard, Krauth (PRL 2011)

Event-chain for continuous potentials

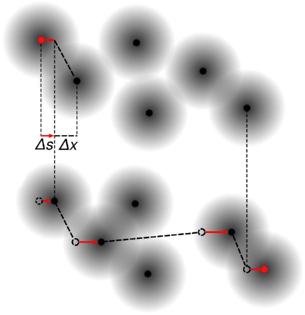


Event-chain algorithm for continuous potentials:

- Exact sampling of canonical partition function.
- Event-driven: ~~time-discretization~~, ~~energy drift~~, ~~thermostat~~, ~~potential cutoff~~ (!).

Département de Physique
École normale supérieure

Event-chain for continuous potentials



Event-chain algorithm for continuous potentials:

- Exact sampling of canonical partition function.
- Event-driven: ~~time-discretization~~, ~~energy drift~~, ~~thermostat~~, ~~potential cutoff~~ (!).

Département de Physique
École normale supérieure

Conclusions

Concepts considered:

- Molecular dynamics (Event-driven)
- Monte Carlo, equiprobability, “tabula rasa” algorithm
- Asakura-Oosawa depletion interaction
- Hard-disk transition
- Lifting

Algorithms considered:

- event-disks.py
- direct-disks.py
- markov-disks.py
- event-chain.py

Département de Physique
École normale supérieure