Μελέτη του Προτύπου 2D-Potts σε Υπολογιστικό Περιβάλλον MATLAB

Παρουσίαση της Διπλωματικής Εργασίας του Γιάννη Ασσιώτη

Στατιστική Μηχανική

- Μέγεθος συστημάτων

↓

Στοχαστική αντιμετώπιση

- Σύστημα
  - Χαμιλτονιανή H
  - Διακριτό σύνολο καταστάσεων
  - Δεξαμενή θερμότητας
  - Δυναμική συστήματος

Στατιστική Μηχανική

- Δεσπόζουσα εξίσωση – master equation

$$\frac{dw_\mu}{dt} = \sum_\nu \left[w_\nu(t) R(\nu \to \mu) - w_\mu(t) R(\mu \to \nu) \right]$$

- Αναμενόμενη τιμή

$$\langle Q \rangle = \sum_\mu Q_\mu w_\mu(t)$$

Στατιστική Μηχανική

- Αν $\forall\ \mu, w_\nu(t)R(\nu \to \mu) = w_\mu(t)R(\mu \to \nu) \Longrightarrow \frac{dw_\mu}{dt} = 0$

↓

Ισορροπία

↓

$$p_\mu \equiv \lim_{t\to\infty} w_\mu(t) = \frac{1}{Z} e^{-E_\mu/kT} \text{ με } Z = \sum_\mu e^{-E_\mu/kT}$$

$$\langle Q \rangle = \sum_\mu Q_\mu p_\mu = \frac{1}{Z} \sum_\mu Q_\mu e^{-\beta E_\mu}$$

Προσομοίωση Monte Carlo

- Προσομοίωση τυχαίων θερμικών διακυμάνσεων
- Δυναμικά χαρακτηριστικά – $R(\mu \to \nu)$

↓ ισορροπία ↓

Κατανομή Boltzmann

Αρχές Προσομοίωση Monte Carlo

- Εκτιμητής

$$\langle Q \rangle = \frac{\sum_\mu Q_\mu e_\mu^{-\beta E_\mu}}{\sum_\mu e_\mu^{-\beta E_\mu}} \to Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} p_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}}{\sum_{j=1}^{M} p_{\mu_j}^{-1} e^{-\beta E_{\mu_j}}}$$

- Δειγματοληψία με κριτήριο σημαντικότητας (Importance sampling)

$$p_\mu = \frac{1}{Z} e^{-\beta E_\mu} \Longrightarrow Q_M = \frac{1}{M} \sum_{i=1}^{M} Q_{\mu_i}$$

Αρχές Προσομοίωση Monte Carlo

- Διαδικασία Markov
  - Πιθανότητα μετάβασης (transition probability) $P(\mu \to \nu)$
    - ανεξαρτησία από χρόνο
    - ανεξαρτησία από προηγούμενες καταστάσεις

↓

κατάσταση θερμικής ισορροπίας

Αρχές Προσομοίωση Monte Carlo

- Διαδικασία Markov
  - Κριτήριο εργοδικότητας (ergodicity)
  - Συνθήκη λεπτομερούς ισοζύγισης (detailed balance)

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{p_\nu}{p_\mu} = e^{-\beta(E_\nu - E_\mu)}$$

Αρχές Προσομοίωση Monte Carlo

- Λόγος Αποδοχής

$$P(\mu \to \nu) = g(\mu \to \nu)A(\mu \to \nu)$$

$g(\mu \to \nu)$ πιθανότητα επιλογής (selection probability)

$A(\mu \to \nu)$ λόγος αποδοχής (acceptance ratio)

$$\downarrow$$

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{g(\mu \to \nu)A(\mu \to \nu)}{g(\nu \to \mu)A(\nu \to \mu)}$$

Αλγόριθμος Metropolis

- Single-flip αλγόριθμος – μικρά q
- Πιθανότητα επιλογής

$$g(\mu \to \nu) = \frac{1}{N} \times \frac{1}{q-1}$$

- Λόγος Αποδοχής

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{g(\mu \to \nu)A(\mu \to \nu)}{g(\nu \to \mu)A(\nu \to \mu)} = \frac{A(\mu \to \nu)}{A(\nu \to \mu)} = e^{-\beta(E_\nu - E_\mu)}$$

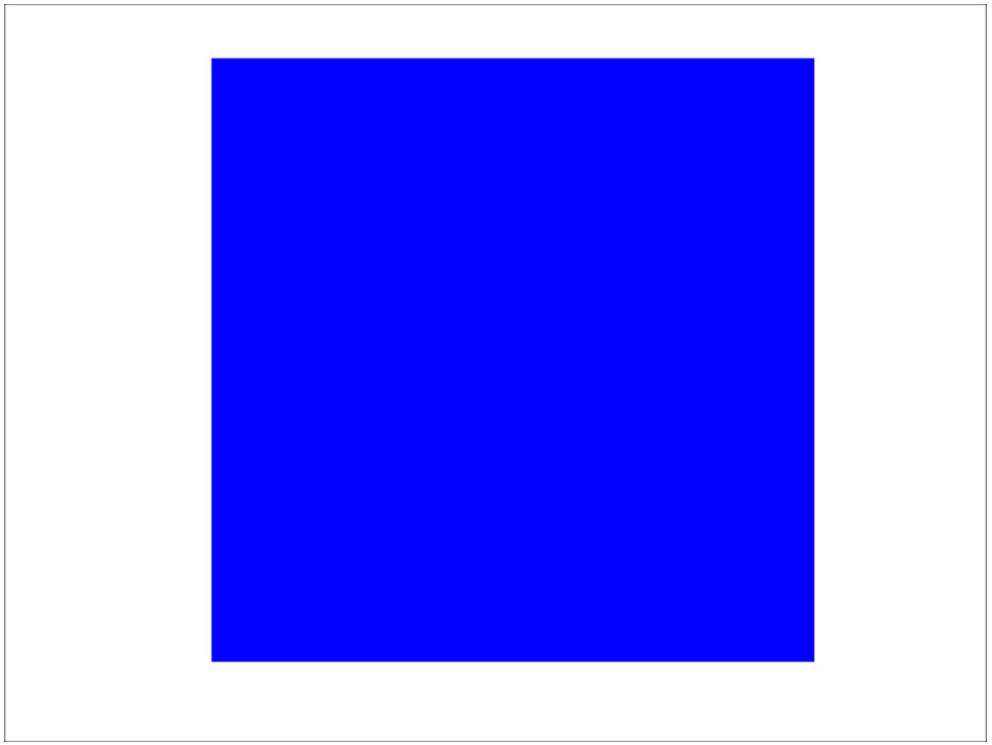
$$A(\mu \to \nu) = \begin{cases} e^{-\beta(E_\nu - E_\mu)} & \text{αν } E_\nu - E_\mu > 0 \\ 1 & \text{αλλιώς} \end{cases}$$

Αλγόριθμος Metropolis

Βήματα αλγορίθμου

1. Τυχαία επιλογή σπιν
2. Τυχαία επιλογή νέας τιμή σπιν
3. Αποδοχή/απόρριψη νέας κατάστασης

Αλγόριθμος Metropolis



Αλγόριθμος Heat-bath

- Single-flip αλγόριθμος – μεγάλα q
- Βάρος Boltzman

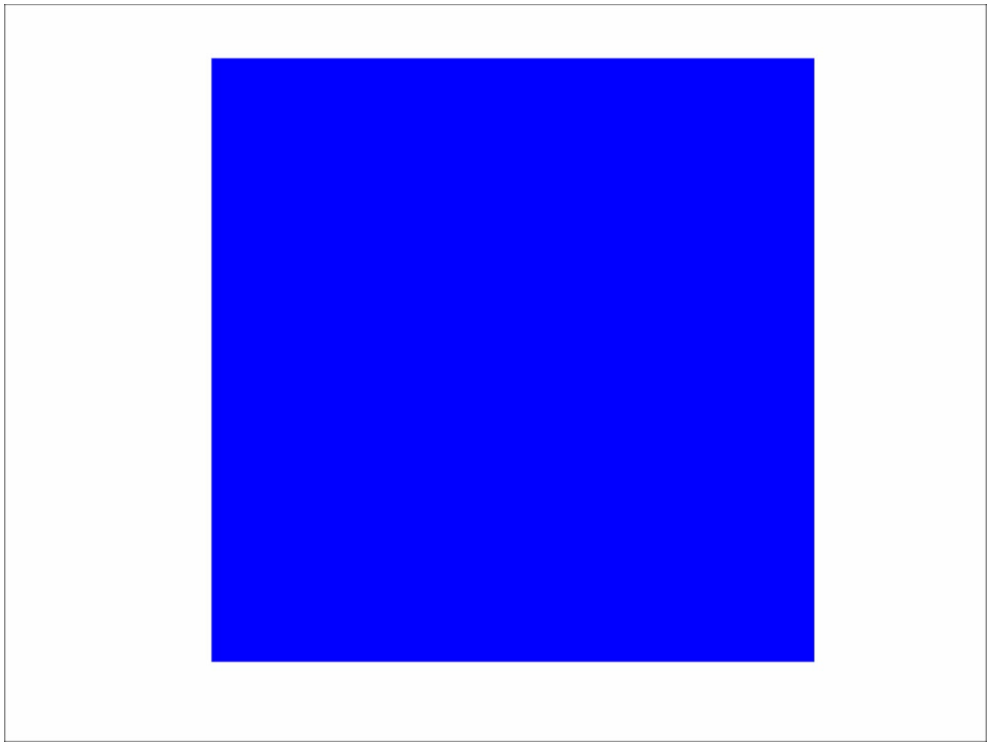
$$p_\nu = \frac{e^{-\beta E_\nu}}{\sum_{\mu=1}^{q} e^{-\beta E_\mu}}$$

- Συνθήκη λεπτομερούς ισοζύγισης

$$\frac{p(\nu \to \nu')}{p(\nu' \to \nu)} = \frac{p_{\nu'}}{p_\nu} = \frac{e^{-\beta E_{\nu'}}}{\sum_{\mu=1}^{q} e^{-\beta E_\mu}} \times \frac{\sum_{\mu=1}^{q} e^{-\beta E_\mu}}{e^{-\beta E_\nu}}$$

$$= e^{-\beta(E_{\nu'} - E_\nu)}$$

Αλγόριθμος Heat-bath



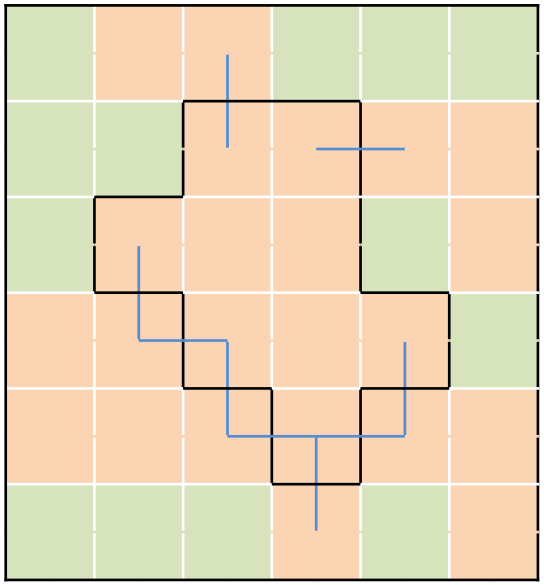
Αλγόριθμος Wolff

- Κρίσιμη επιβράδυνση – $\beta_c = \ln\left(1+\sqrt{q}\right)$
- Cluster αλγόριθμος
- Ανάπτυξη cluster - P_{add}
- Συνθήκη λεπτομερούς ισοζύγισης

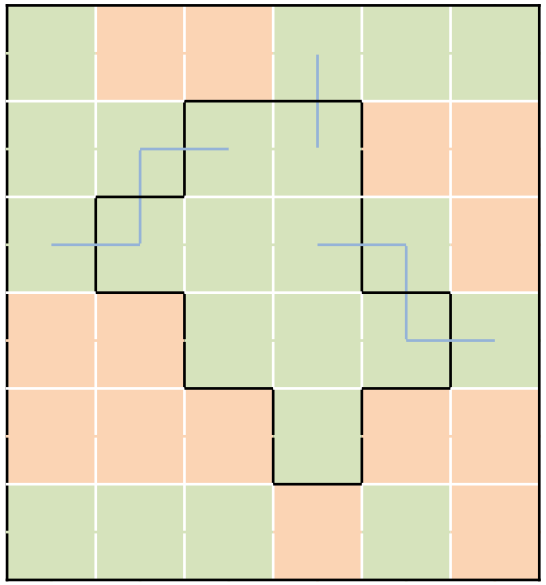
$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = e^{-\beta\left(E_\nu - E_\mu\right)}$$

Αλγόριθμος Wolff

$$E_\nu - E_\mu = (-n) - (-m) = m - n$$



$m = 9$



$n = 7$

Αλγόριθμος Wolff

$$g(\mu \to \nu) = p_{seed} \times p_{yes}^{int} \times p_{no}^{border}$$

$$p_{seed} = \frac{1}{N}$$

$$p_{yes}^{int}(\mu \to \nu) = p_{yes}^{int}(\nu \to \mu) = C_{\mu\nu}$$

$$p_{no}^{border}(\mu \to \nu) = (1 - P_{add})^{m}$$

$$p_{no}^{border}(\nu \to \mu) = (1 - P_{add})^{n}$$

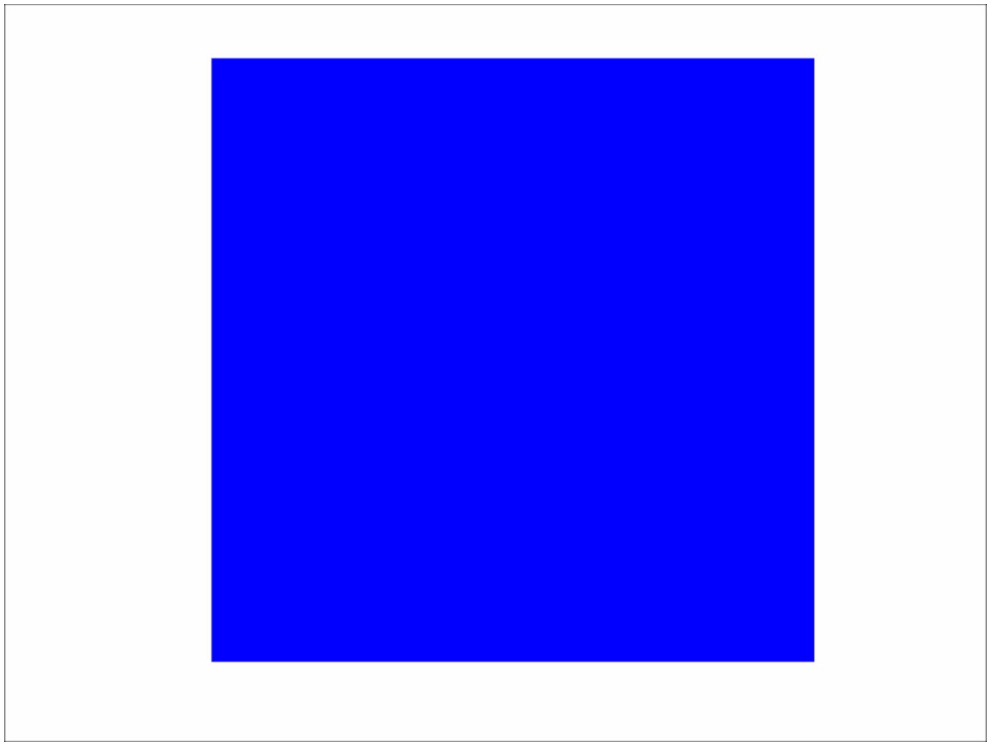
$$(1 - P_{add})^{m-n} \frac{A(\mu \to \nu)}{A(\nu \to \mu)} = e^{-\beta(m-n)} \to P_{add} = 1 - e^{-\beta}$$

Αλγόριθμος Wolff

Βήματα αλγορίθμου

1. Τυχαία επιλογή σπιν-γεννήτορα
2. Επαγωγικά – εξέταση πλησιέστερων γειτόνων $P_{add} = 1 - e^{-\beta}$.
3. Τερματισμός ανάπτυξης cluster
4. Νέα τιμή σπιν για το cluster

Αλγόριθμος Wolff



Δοκιμή 1 – Metropolis

- Απαίτηση υπολογισμού ενέργειας – ΔΕ

  $e^{-\beta(E_\nu - E_\mu)}$

- Single-flip → τοπικός υπολογισμός
- Αποθήκευση εκθετικών σε array
- met_naive.pdf

Δοκιμή 1 – Heat-bath

- Απαίτηση υπολογισμού ενέργειας

$$p_n = \frac{e^{-\beta E_n}}{\sum_{m=1}^{q} e^{-\beta E_m}}$$

- Single-flip → τοπικός υπολογισμός

$$p_n = \frac{e^{-\beta E_n}}{\sum_{m=1}^{q} e^{-\beta E_m}} \frac{e^{\beta E_\nu}}{e^{\beta E_\nu}}$$
$$= \frac{e^{-\beta (E_n - E_\nu)}}{\sum_{m=1}^{q} e^{-\beta (E_m - E_\nu)}}$$
$$= \frac{e^{-\beta \Delta E_n}}{\sum_{m=1}^{q} e^{-\beta \Delta E_m}}$$

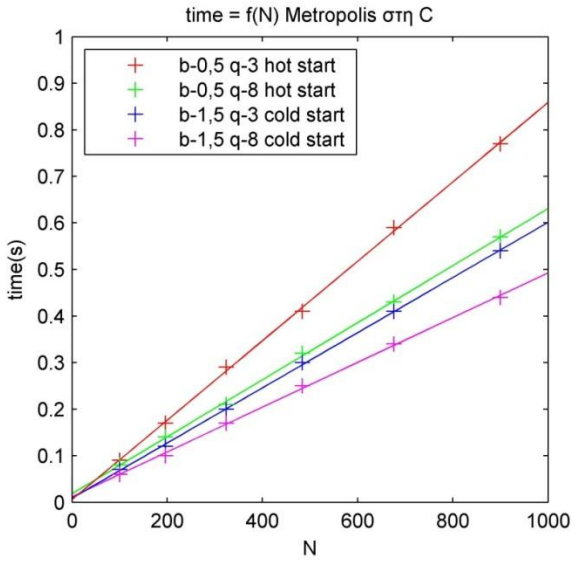
- heat_naive.pdf

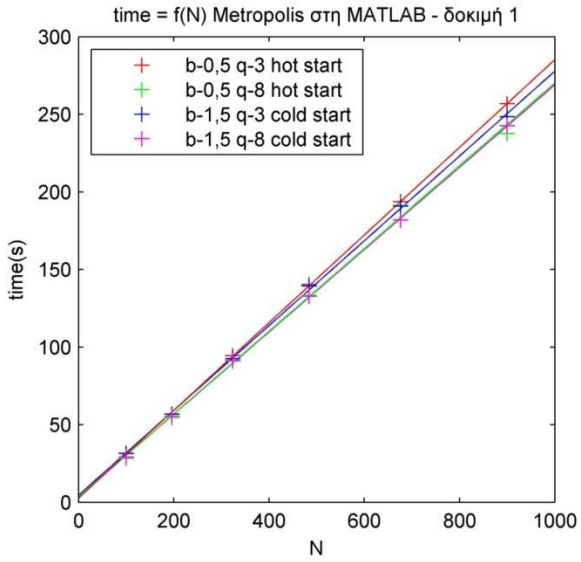
Δοκιμή 1 – Wolff

- Ανάπτυξη του cluster → seed spin
- cluster ↔ stack

  $P_{add} = 1 - e^{-\beta}$ ← υπολογισμένο εξαρχής
- wolff_naive.pdf

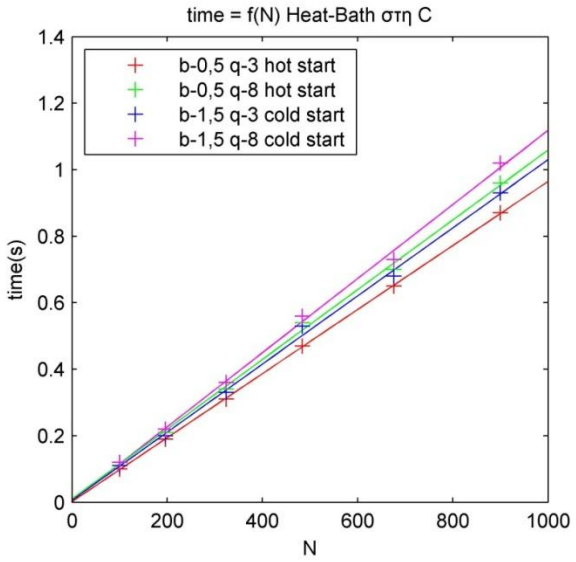
Δοκιμή 1 - Αποτελέσματα

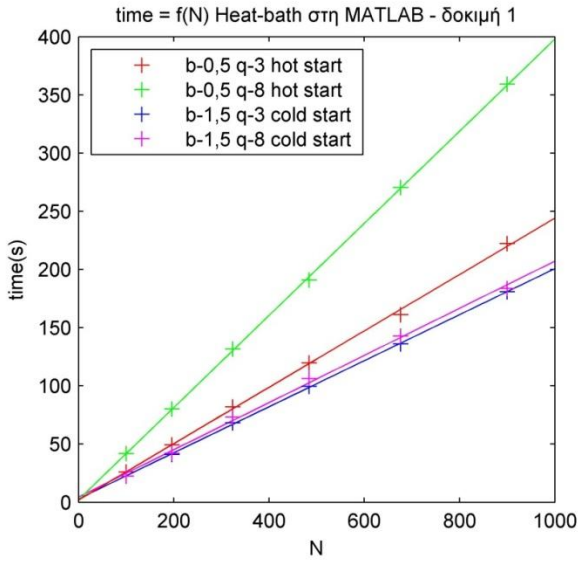




$$\frac{t_{MAT}}{t_C} \approx 450$$

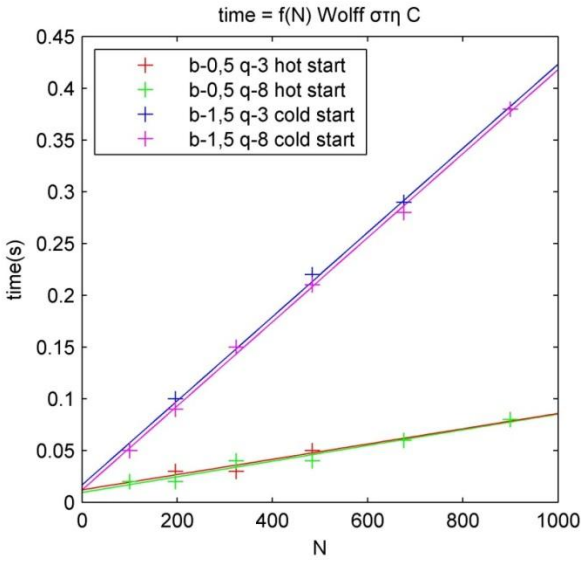
Δοκιμή 1 - Αποτελέσματα

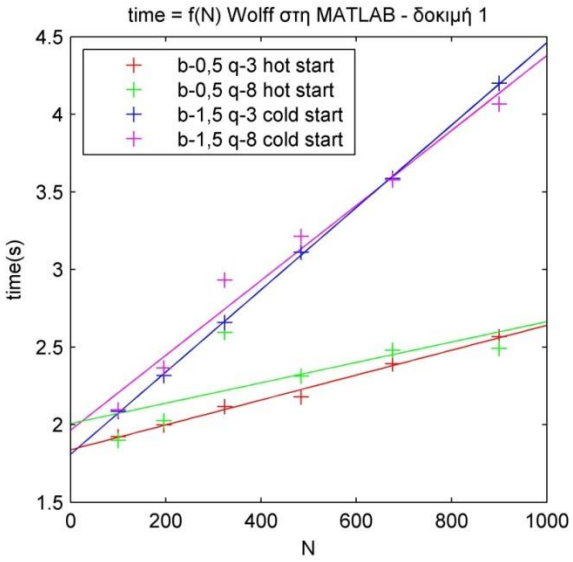




$$\frac{t_{MAT}}{t_C} \approx 250$$

Δοκιμή 1 - Αποτελέσματα





$$\frac{t_{MAT}}{t_C} \approx 8$$

Δοκιμή 2

- Κακή απόδοση – Ιδιαιτερότητες του MATLAB
  χρήση/διαμόρφωση πινάκων, logical arrays

	57	58	59	60	61	62	63	64	
8	1	2	3	4	5	6	7	8	1
16	9	10	11	12	13	14	15	16	9
24	17	18	19	20	21	22	23	24	17
32	25	26	27	28	29	30	31	32	25
40	33	34	35	36	37	38	39	40	33
48	41	42	43	44	45	46	47	48	41
56	49	50	51	52	53	54	55	56	49
64	57	58	59	60	61	62	63	64	57
	1	2	3	4	5	6	7	8	

Δοκιμή 2

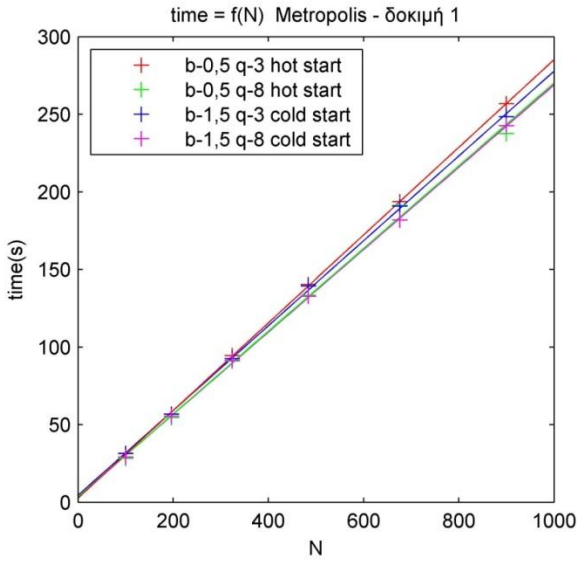
Προσθήκη νέων μεθόδων

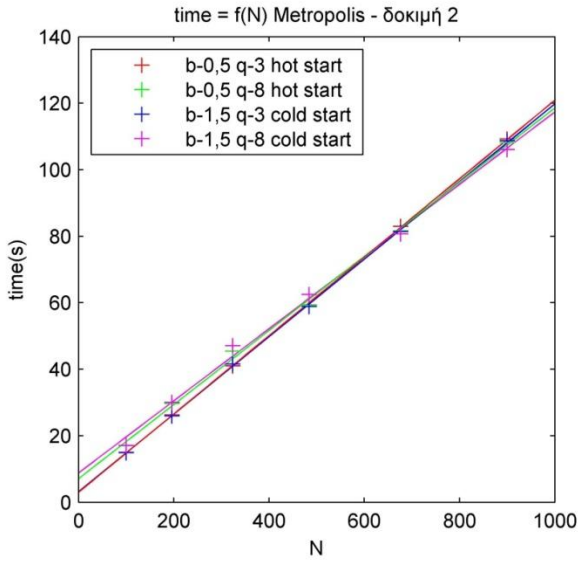
`nnfinder_per(L)`

`spliter(L)`

- Metropolis
- Heat-bath
- Wolff

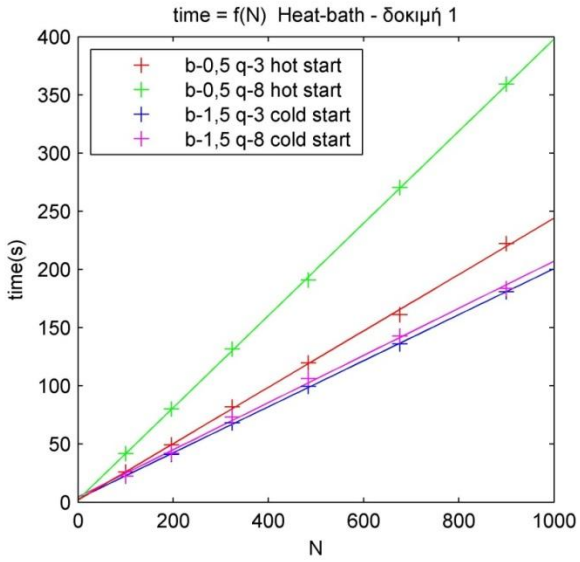
Δοκιμή 2 – Αποτελέσματα

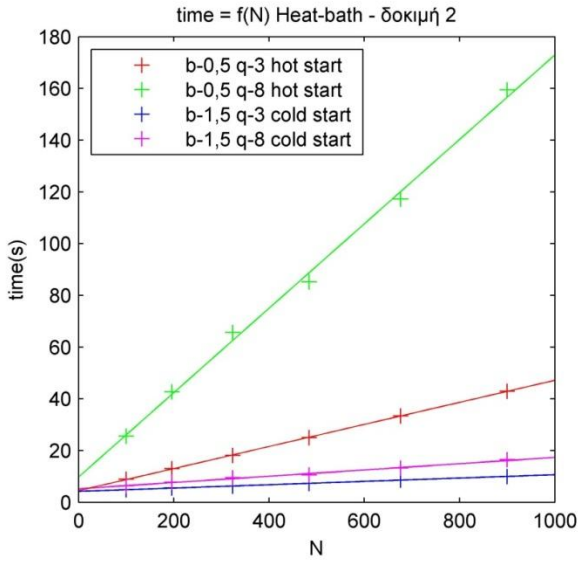




$$\frac{t_{MAT_1}}{t_{MAT_2}} \approx 2{,}5$$

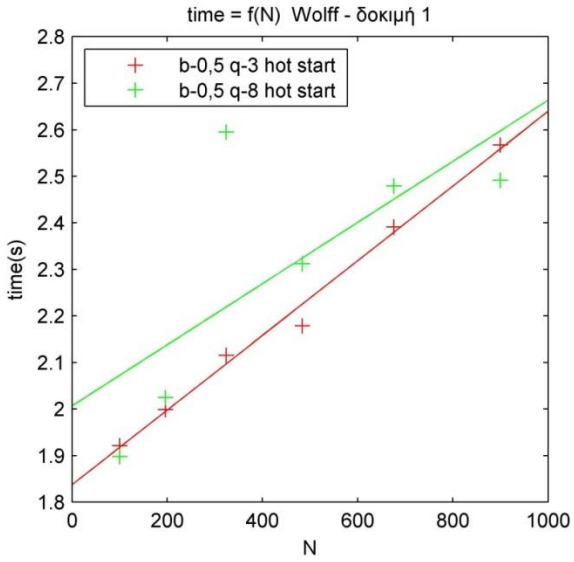
Δοκιμή 2 – Αποτελέσματα

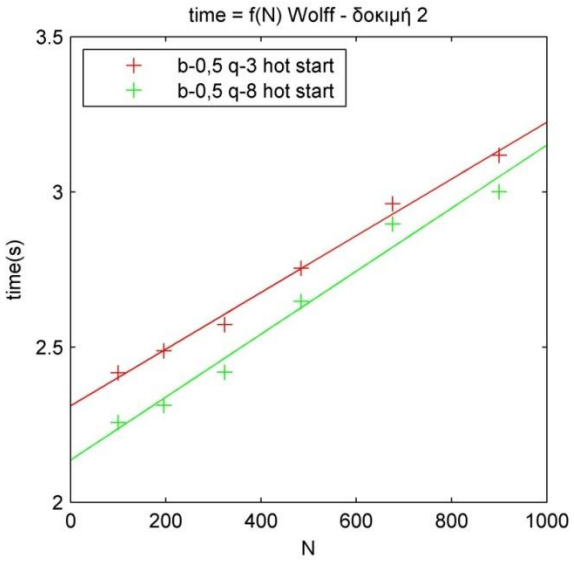




$$\frac{t_{MAT_1}}{t_{MAT_2}} \approx 2 - 7$$

Δοκιμή 2 – Αποτελέσματα





$$\frac{t_{MAT_1}}{t_{MAT_2}} \approx 1$$

Δοκιμή 3 - Metropolis

Profiler

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
23	`newstate(k) = avalstates(randi...`	1000000	10.243 s	57.0%	
22	`avalstates = find(ALLSTATES-ol...`	1000000	4.400 s	24.5%	
24	`end`	1000000	1.334 s	7.4%	
26	`Ei = sum((S(NN(n,:))==[oldstat...`	20000	0.383 s	2.1%	
15	`global Q N PROB S ALLSTATES NN`	20000	0.373 s	2.1%	
All other lines			1.232 s	6.9%	
Totals			17.964 s	100%	

Πρόβλημα στην επιλογή των νέων τιμών σπιν

Δοκιμή 3 - Metropolis

```
%% Find new spin values, different from the  initial ones
for k = 1:(N/2),
    avalstates = find(ALLSTATES-oldstate(k));
    newstate(k) = avalstates(randi(Q-1,1,1));
end
```

↓

```
%% Find new spin values, different from the initial ones
newstate  = randi(Q,N/2,1);
newstate = abs(newstate-((oldstate==newstate)*double(Q+1)));
```

Δοκιμή 3 - Metropolis

Profiler

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
15	`global Q N PROB S NN`	20000	0.303 s	15.5%	
23	`Ei = sum((S(NN(n,:))==[oldstat...`	20000	0.282 s	14.4%	
24	`Ef = sum((S(NN(n,:))==[newstat...`	20000	0.217 s	11.1%	
19	`newstate  = randi(Q,N/2,1);`	20000	0.212 s	10.8%	
27	`accept = (delta<=0) | (rand...`	20000	0.192 s	9.8%	
All other lines			0.752 s	38.4%	
Totals			1.958 s	100%	

Δοκιμή 3 – Heat-bath

Profiler

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
93	`qs = nonzeros(nonzeros(nonzero...`	429322	22.300 s	81.2%	
96	`S(nremain(ii)) = qs(bb(ii));`	429322	0.763 s	2.8%	
20	`global N Q S NN PROB ALLSTATES`	20000	0.359 s	1.3%	
33	`nnz = nnz + [(z12+z13+z14),(z1...`	20000	0.357 s	1.3%	
69	`logicarr = nonzeros(n .* tup ....`	20000	0.301 s	1.1%	
All other lines			3.377 s	12.3%	
Totals			27.457 s	100%	

Πρόβλημα στην επιλογή των νέων τιμών σπιν για τα μη ευθυγραμμισμένα με τους γείτονες τους σπιν

Δοκιμή 3 – Heat-bath

```
%% apply the new values to the lattice
for ii=1:length(bb),
    qs = nonzeros(nonzeros(nonzeros(nonzeros(ALLSTATES - ...
       nnsremain(ii,1)) + nnsremain(ii,1) - ...
       nnsremain(ii,2)) + nnsremain(ii,2) - ...
       nnsremain(ii,3)) + nnsremain(ii,3) - ...
       nnsremain(ii,4)) + nnsremain(ii,4);
    S(nremain(ii)) = qs(bb(ii));
end
```

Δοκιμή 3 – Heat-bath

```
%% apply the new values to the lattice
avalstates = repmat(1:Q,length(bb),1);
avalarray = logical((repmat(nnsremain(:,1),1,Q)==avalstates)+ ...
                    (repmat(nnsremain(:,2),1,Q)==avalstates)+ ...
                    (repmat(nnsremain(:,3),1,Q)==avalstates)+ ...
                    (repmat(nnsremain(:,4),1,Q)==avalstates));

avalstates(avalarray)=0;
avalstates = sort(avalstates,2,'descend');
indices_array = sub2ind(size(avalstates), 1:length(bb), bb')';

S(nremain) = avalstates(indices_array);
```

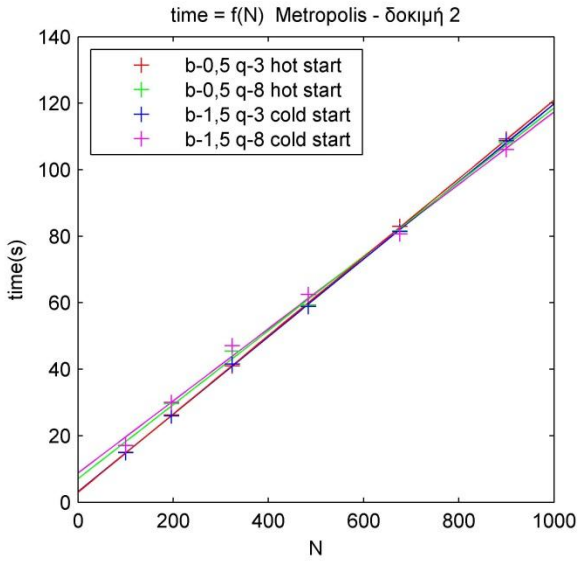
Δοκιμή 3 – Heat-bath

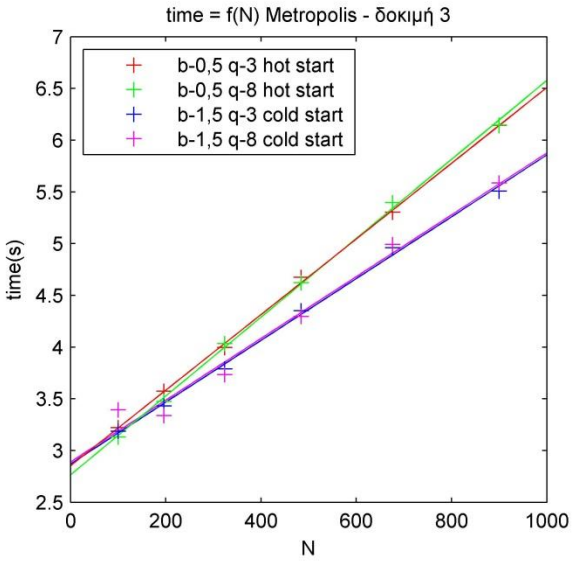
Profiler

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
92	avalarray = logical((repmat(nn...	20000	4.673 s	38.3%	
99	indices_array = sub2ind(size(a...	20000	1.491 s	12.2%	
91	avalstates = repmat(1:Q,length...	20000	1.061 s	8.7%	
20	global N Q S NN PROB ALLSTATES	20000	0.420 s	3.4%	
33	nnz = nnz + [(z12+z13+z14),(z1...	20000	0.368 s	3.0%	
All other lines			4.186 s	34.3%	
Totals			12.199 s	100%	

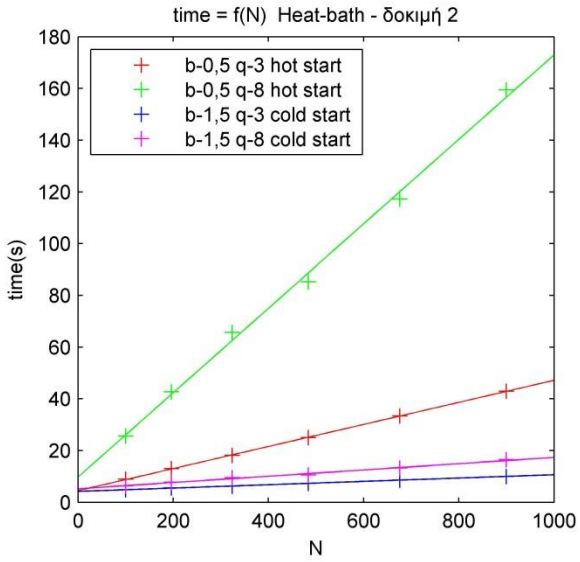
Δοκιμή 3 – Αποτελέσματα

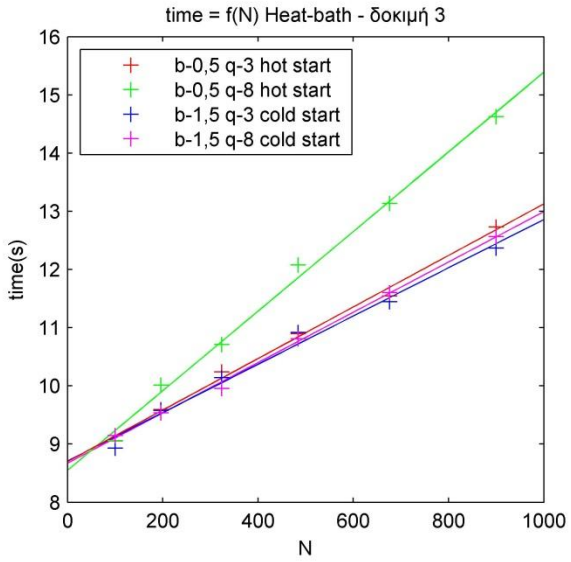




$$\frac{t_{MAT_2}}{t_{MAT_3}} \approx 20$$

Δοκιμή 3 – Αποτελέσματα





$$\frac{t_{MAT_2}}{t_{MAT_3}} \approx 1 - 10$$

Τελικά Αποτελέσματα

- Metropolis

$$\frac{t_{MAT_C}}{t_{MAT_3}} \approx 9 - 14$$

- Heat-bath

$$\frac{t_{MAT_C}}{t_{MAT_3}} \approx 26$$

- Wolff

$$\frac{t_{MAT_C}}{t_{MAT_3}} \approx 37$$

Συμπεράσματα

MATLAB

- ολοκληρωμένο προγραμματιστικό περιβάλλον
- ευκολίες στην γραφή του κώδικα
- εργαλεία ανάλυσης και παρουσίασης δεδομένων
- πλήρης, εύκολα προσβάσιμη βοήθεια offline και online

C

- ταχύτεροι υπολογισμοί
- πολλοί δωρεάν compilers
- ευρεία διάδοση/χρήση
- μεγάλη online κοινότητα